SAPIENZA
UNIVERSITÀ DI ROMA

PhD Thesis in Automatica, Bioingegneria e Ricerca Operativa

XXXII Ciclo

Multi-Sensor Coordination in Human-Robot Interaction

Maram Khatib

Advisor Prof. Alessandro De Luca

October 2019

Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG)
Sapienza Università di Roma

Dedicated to the memory of Fabrizio Flacco.

Abstract

In the framework of Human-Robot interaction, a robot and a human operator may need to move in close coordination within the same workspace. In this thesis, the contactless human-robot collaboration with coordinated motion tasks is considered. A control system based on multiple sensors is presented for safe and efficient collaboration. A contactless coordinated motion can be achieved using vision, mounting a camera either on the robot end-effector or on the human. We consider here a visual coordination task, with the robot end-effector that should maintain a prescribed position with respect to a moving RGB-D camera while pointing at it. For the 3D localization of the moving camera, we compare three different techniques and introduce some improvements to the best solution found for our application. Instead, an Oculus Rift HMD system can be used to track an operator moving in the workspace. In this case, a stereo camera is used to perform a mixed-reality interface that enables the user to choose between different collaboration modes. The robot should also avoid any collision with the operator and with nearby static or dynamic obstacles, based on distance computations performed in the depth space of a fixed Kinect sensor. To exploit effectively and efficiently the advantage of robot redundancy, different soft constraints for both the coordinated motion and collision avoidance tasks are proposed. Two relaxed versions of the pointing part of the task are introduced to achieve the desired task without exhausting the robot capabilities. Also, a relaxed formulation for collision avoidance task, that does not slack the avoidance performance, is used. Several control algorithms with different complexity are developed to suitably combine and organize the simultaneous control tasks with their priority. The proposed control system using different approaches is validated by V-REP and MATLAB simulations, and experiments with the 7-dof KUKA LWR manipulator.

Keywords: *Robotics, Human-Robot Collaboration, Relaxed Tasks, Motion Control, Redundant Robots, Mixed-Reality.*

Acknowledgements

Firstly, I would like to express my sincere gratitude to my teacher and advisor Prof. Alessandro De Luca for the sustained support of my Ph.D. study and related research, motivation, and immense knowledge.

In addition, I would like to thank all Robotics Lab members who shared with me this journey, and all my friends for their motivation and nice refresh breaks during the day.

A big thanks to my parents and brothers for their endless encouragement, support, and patience for being far from them.

My great gratitude and thanks to my soulmate Khaled AL Khudir for being my colleague, friend, and husband. I am so appreciative of his support and assistance in every step of my Ph.D. journey.

A very special thanks to my sweet little daughter Yafa for being such a good girl. She helped me in her calmness and amused me in her innocence.

Finally, I am grateful for the financial support of my work, provided by the Italian Ministry of Foreign Affairs and International Cooperation (Farnesina).

Contents

Introduction

The capability of handling tasks that involve interaction between humans and robots has become nowadays a highly desirable feature in service applications of robotics De Santis et al. [2008], as well as one of the enabling technologies of Industry 4.0 (Hägele et al. [2016], Lu [2017]). Robot co-workers should be able to share their workspace and collaborate safely with humans, combining and enhancing the skills of both parties. Safe handling of human-robot interaction tasks can be achieved within a hierarchical control architecture organized in three functional layers De Luca and Flacco [2012]. With reference to Fig. 1, the three nested layers are concerned with *safety*, *coexistence*, and *collaboration*, which easily map into the four forms of 'collaboration' modes of the ISO 10218 standard (enhanced by the technical specification TS 15066). Each layer contains specific methods related to its objectives, with their associated sensory requirements and control algorithms.

Safety is the most important feature of a robot that works close to human beings, and should always be enforced in any condition. The *safety* layer at the bottom is always active and deals with collision detection, specifying also how the robot should promptly react to undesired (and unavoidable) contacts. In (De Luca and Mattone [2005], De Luca et al. [2006]) a model-based method is introduced to detect collisions and isolate the link subject to the contact force which does not need exteroceptive or distributed tactile sensors. The method works for robots with rigid or with flexible joints, in the latter case both with and without joint torque sensing Haddadin et al. [2008]. Considering robot dynamics, the unknown torque due to a collision/contact force with the human/environment at a generic robot location, is approximated using the so-called *residual* vector. The critical aspects of friction and payload estimation in the robot dynamic model have been analyzed in Gaz and De Luca [2017]. The model-based terms in the residual can be efficiently computed via a Newton-Euler implementation following De Luca and Ferrajoli [2009]. The same holds true for robots with elastic joints in the absence of joint torque sensing Buondonno and De Luca [2015], and for VSA systems Buondonno and De Luca [2016]. When the robot dynamics is poorly known, an alternative signal-based method for detecting collisions (though without their isolation) relies on processing the motor currents in Geravand et al. [2013]. A logic based on high-pass and low-

(a)

Safety	Coexistence **Safety-rated monitored stop**	Collaboration **Hand guiding**	Coexistence **Speed and separation monitoring**	Collaboration **Power and force limiting**
Speed	Zero while operator in CWS	Safety-rated monitored speed	Safety-rated monitored speed	Max determined by RA to limit impact forces
Separation distance	Small or zero	Small or zero	Safety-rated monitored distacne	Small or zero
Torques	Gravity + load compensation only	As by direct operator input	As required to execute application and maintain min separation distance	Max determined by RA to limit static forces
Oprerator controls	None while operator in CWS	E-stop; Enabling device; Motion input	None while operatorin CWS	As required by application
Main risk reduction	No motion in presence of operator	Motion only by direct operator input	Contact between robot and operator prevented	By design or control, robot cannot impart excessive force

(b)

Figure 1: (a): The three nested layers of the hierarchical control architecture for pHRI proposed in De Luca and Flacco [2012]; (b): The mapping of the three control layers into the four modes of the ISO standards on robot safety.

pass filtering of motor currents has been used to distinguish the nature of the detected impact, i.e., a soft/slow contact (possibly signifying the start of an intentional collaboration) or a hard/fast collision (always undesired). The same can be performed with two residuals having low and high bandwidths. The various possible implementations and a comparison of these approaches have been described in Haddadin et al. [2017].

Coexistence occurs when the robot shares the workspace with humans, without requiring mutual contact. This is probably the most common situation in an Industry 4.0 environment, when a lightweight robot and human operator should work side-by-side. Safety requirements must still be guaranteed, obtaining thus a safe coexistence. Monitoring the workspace, e.g. with exteroceptive (camera, depth laser) sensors, and computing online relative distances between the full body of the robot in motion and the operator are fundamental to prevent collisions and to reduce the speed of the robot TCP in potentially critical situations. This should be obtained without giving up robot mobility and dexterity, by oversizing the safety areas.

For the detection of obstacles in the robot workspace, several sensors and methods have been proposed. Laser and sonar sensors can scan the workspace and detect obstacles intersecting a 2D plane (usually, parallel to the floor and at the calf height or at the torso), allowing the robot to avoid that part of the human body (Ulrich and Borenstein [1998], Brock and Khatib [1999]). Detection of the whole human body (or, simultaneously, of several of its parts) can be achieved by either attaching passive or active markers to the body, or by extracting its shape from RGB/depth images as a 'point cloud' in the Cartesian space Polverini et al. [2014]. However, the robot would avoid in this way only the human body and thus neglect other dangerous obstacles in the workspace. In Lacevic et al. [2013], a laser sensor was attached close to the robot end-effector to compute distances and danger zones from nearby obstacles. Unfortunately, repeating this arrangement for every robot link (which may also possibly collide) would be inefficient and too expensive. In (Flacco et al. [2012b], Flacco et al. [2015b]), an efficient robot-obstacle distance computation is introduced. The proposed approach works directly in the depth space of a RGB-D sensor, achieving collision avoidance with real time (300 Hz) performance. To avoid gray zones or sensor occlusion, the algorithm has been extended to the case of multiple depths sensor Fabrizio and De Luca [2016]. The optimal placement of multiple (depth and/or presence) sensors in the workspace is investigated in Flacco and De Luca [2010]. A GPU implementation of the algorithm Magrini and De Luca [2017] allows to preserve efficiency while computing distances with any moving object or any human part. More recently, this safe coexistence approach has been implemented in a full-size industrial cell with a large ABB robot, where the depth space method using two Kinects was combined/integrated with additional laser scanners for ISO compliance Magrini et al. [2020].

Based on this distance information, collisions can be avoided by any preferred variant of the artificial potential fields method Khatib [1986b]. In case of kinematic redundancy, it is preferred to preserve the desired tasks while avoiding any collision. This can be done with the common null space projection Siciliano and Slotine [1991]. Instead, the avoidance tasks are translated into hard inequality joint velocity constraints which should be respected by the other desired tasks Flacco et al. [2012a]. To keep the robot dexterity, a relaxed avoidance task is used with the so-called *Task Priority Matrix* in order to control the robot end-effector to achieve efficiently the prioritized desired Cartesian tasks while avoiding any collision Khatib et al. [2019b].

Finally, physical Magrini et al. [2015a] or contactless Khatib et al. [2017] human-robot *collaboration* is established in the top layer. Collaboration occurs when the robot performs complex tasks with direct human interaction and coordination, the most demanding and critical feature in safe pHRI. During a physical collaboration, there is an explicit and intentional contact with controlled exchange of forces between human and robot. Safety and coexistence should also be guaranteed during physical collaboration. For example, if the human is collaborating with the robot using his/her right hand, contact between the robot and the left hand or the rest of the human body is undesired, and therefore such accidental contacts are treated as potential collisions that must be avoided.

The estimation of contact forces at a generic point of the robot is proposed without using force/torque sensing Magrini et al. [2014]. This virtual force sensing is obtained determining first the localization of the contact point on

the robot body, by combining the information from an external RGB-D sensor (when the distance computed by the depth space algorithm is close to zero) with the residual-based contact detection algorithm (when at least one residual component crosses a small threshold). Once the contact point has been determined, it is possible to extract from the residual also an estimate of the contact force by pseudoinversion of the transpose of the $3 \times n$ Jacobian matrix related to the contact point. When the contact occurs far down the serial chain of the robot links (at a link 6) the action line of the contact force can be estimated also without the external sensor Haddadin et al. [2017]. At this stage, the knowledge of the estimate of the Cartesian contact force can be used for designing a number of classical control laws, regulating or tracking reference values for motion and/or force quantities. However, this occurs in a generalized whole-body sense, namely the desired behavior is not imposed at the joint or end-effector level, but directly at the level of the detected contact position in the robot structure. In Magrini et al. [2014], human-robot collaboration has been realized using the estimated contact force in an admittance scheme, with the robot controlled in position mode. Torque control laws based on generalized impedance and task-consistent force control Magrini et al. [2015b], as well as hybrid force-velocity control Magrini and De Luca [2016], have also been proposed. With a F/T sensor, it is possible to handle simultaneously and separately both intentional contacts at the end-effector and reaction to undesired collisions on the robot body Gaz et al. [2018]. To this end, kinematic redundancy of the robot can be exploited together with the relaxation of multiple tasks with priority Magrini and De Luca [2017]. A similar approach works also when a dynamic model is not available, using the motor currents as proxies of the residuals Mariotti et al. [2019].

Beside such a physical collaboration, contactless collaboration could also be established. A contactless collaborative task is typically realized by imposing a coordinated motion between the robot and a human operator under safety premises. For such a collaboration, localizing the human pose and detecting obstacles in the workspace should both be guaranteed (Khatib et al. [2017], Khatib et al. [2019a]). Spatial and temporal motion coordination can be obtained via direct and explicit communication, such as using gestures and/or voice commands Rogalla et al. [2002], or by indirect communication, such as recognizing intentions Nehaniv et al. [2005], raising the attention with legible action Mainprice et al. [2010], or passively following the human motion. Each type of collaboration raises different challenges. Indeed, as a common feature, the robot should always be aware of the existence of the human in the workspace, and specifically of the pose of his/her body parts. Use of laser range measurements Svenstrup et al. [2009] and of vision/depth cameras are the preferred choices for this purpose, followed by the extraction of the human pose from the sensor data Villani et al. [2009]. Based on this information, a coordinated motion task can be defined online by requiring the robot to track some human feature in a specific way. Another modality is to attach a compact RGB-D sensor on the human body, and then localizing with different techniques (Besl and McKay [1992], Davison [2003]). In Khatib et al. [2017], a comparison between three different localization methods introduced in Kato and Billinghurst [1999], Klein and Murray [2007], and Serafin and Grisetti [2015] was done. All these techniques suffer from inefficiency during fast human motion, in highly dynamic environments, or when markers/features are not present. Moreover, they need a

frequent and complex calibration phase. To overcome such problems, the Oculus Rift system (a HMD for Virtual Reality (VR) exploration) could be used Khatib et al. [2019a]. This sensor does not need markers or specific features, allows the human to look and move freely in the workspace, and provides a sufficiently accurate pose estimation both in static and dynamic environments, during fast human motion, and in bad lighting conditions. Furthermore, It can be used to perform a mixed-reality interface for end-user robot programming (Gadre et al. [2019], Khatib et al. [2019a]).

In this thesis, we address simultaneously a contactless human-robot collaboration supported with a mixed-reality interface and safe coexistence, based on a multi-sensor control system in Fig. 2. We consider a contactless collaborative scenario with direct communication and address the motion control problem for a coordination task in which a robot should track the motion of a human head (for instance, to show an item held by its end-effector) while avoiding any collision.

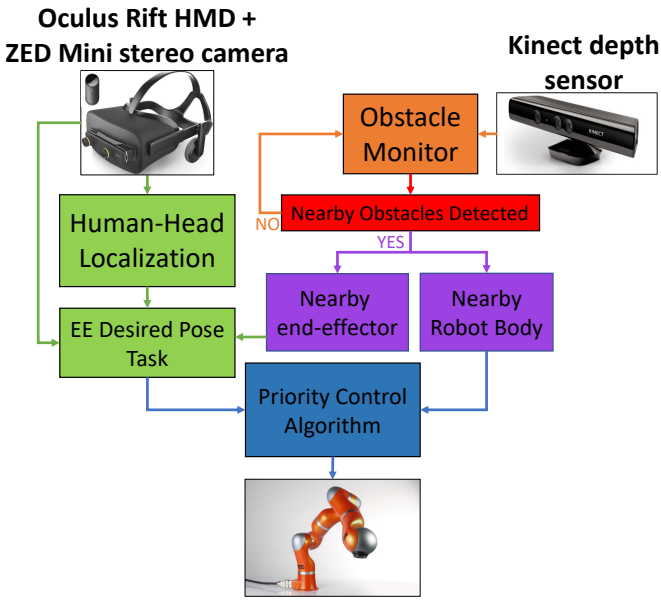


Figure 2: The proposed multi-sensor control scheme for safe coordinated human-robot collaboration.

At the beginning, the problem of Human-Head pose localization is investigated. First, for a RGB-D camera carried by the human (say, mounted on a helmet), different approaches for camera pose localization are compared experimentally and various improvements are proposed. As an alternative, an Oculus Rift is used to localize efficiently the human pose. In this case, a stereo camera is attached to perform a friendly human-robot interaction through a mixed-reality interface, that represents the current robot task and the other available collaboration modes which the user can switch between them. In this environment, both the robot and the operator will be aware about each other current activity.

Next, a coordinated motion task is defined in which the robot end-effector should maintain a desired relative position with respect to the head of a moving human operator while pointing at it. Different relaxed definitions for the pointing task are presented. Moreover, Cartesian task limits, w.r.t the robot work space, are specified to adjust the desired task accordingly. During the collaboration, the robot should avoid any collision with other body parts of the operator and with all nearby obstacles. This is done based on distance evaluations that use a Kinect depth sensor placed in the environment. For this, various definitions for the collision avoidance task, with different dimensions, are used and compared.

Subsequently, we suitably combine the previous tasks and organize them with a priority. Despite the many degrees of freedom of the chosen manipulator, the mobility resources of the robot should be carefully used when facing the multiple control tasks. For this, different kinematic control laws, that take advantage of the available robot redundancy to handle the desired tasks in a specific priority, are proposed and compared. The different presented approaches are implemented and tested in V-REP and MATLAB simulations and experiments with a KUKA LWR arm.

The present thesis is structured as follows. In Chapter 1, four methods for moving sensor pose localization are compared. Also, the proposed mixed-reality interface for friendly human-robot collaboration is presented. Chapter 2 contains the complete definitions of the different coordination tasks. The relaxed pointing tasks using equality and inequality constraints are shown. Also, the proposed task limits sphere is illustrated. Chapter 3 includes the distance computations and different definitions for the collision avoidance task. In Chapter 4, four different motion control approaches are illustrated to handle the coordination tasks with/without collision avoidance using different localization sensors. Also, comparative simulations and experiments are presented.

The ideas and methods presented in this thesis, have been published/are being submitted, as author's original work in:

- M. Khatib, K. Al Khudir, and A. De Luca, "Visual coordination task for human robot collaboration". In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3762–3768, 2017.
- M. Khatib, K. Al Khudir, and A. De Luca,"Multi-sensor control system for safe human-robot collaboration with mixed-reality interface", in preparation to the *Robotics and Computer Integrated Manufacturing.*
- M. Khatib, K. Al Khudir, and A. De Luca, "Robot collision avoidance using tasks priority matrix with soft constraints at the acceleration level", in preparation to the *IEEE Robotics and Automation Letters.*

1 Human-Robot Awareness

1.1 Introduction

To perform a friendly contactless collaboration experience, both the robot and the human should be aware about each other current action and location. In this chapter we propose to mount a sensor (i.e. RGB camera, depth camera, Oculus Rift) on the human head; by localizing the sensor; the robot will be aware of human head pose in order to collaborate with it.

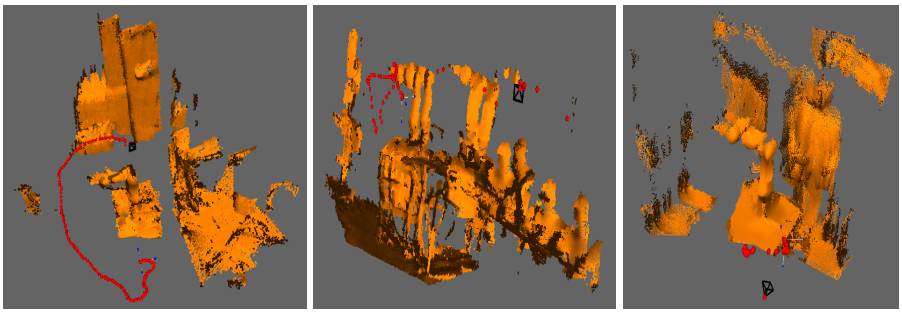
Sensor localization may be performed with vision-based or LiDAR-based methods. Vision-based methods either locate fiduciary markers using feature recognition algorithms from computer vision, or work in markerless fashion by estimating the RGB camera pose by features extraction and point correspondences in stereo images Davison [2003]. On the other hand, LiDAR-based methods localize a RGB-D sensor based on the Iterative Closest Point (ICP) algorithm Besl and McKay [1992], which compares the current Point Cloud with a reference cloud (typically, the initial one).

Oculus Rift device together with its tracking sensor could be used also for localization. In this case, we propose to design a Mixed Reality Head-Mounted Display (MR-HMD) interface that enables the human to know what the robot is currently doing. Moreover we allow the user to switch between different collaboration modes using the Oculus supplied controller.

In the following sections we have tested and compared different localization methods to find the best one that fits our dynamic sensor localization problem.

1.2 NICP

The first (markerless) method considered was the Normal Iterative Closest Point (NICP) Serafin and Grisetti [2015], which solves the Point Cloud Registration (PCR) problem, i.e., it finds the transformation that aligns at best the common parts of two point clouds. PCR is used in 2D- or 3D-surface reconstruction, in robot localization, path planning and many other applications. NICP considers



(a) (b) (c)

Figure 1.1: Outputs with the NICP method in three different operative conditions: (a) slow camera moving in static environment; (b) faster motion; (c) static camera in dynamic environment. The red dots represent the camera pose estimation and the yellow parts are the 3D reconstruction of the camera scenes.

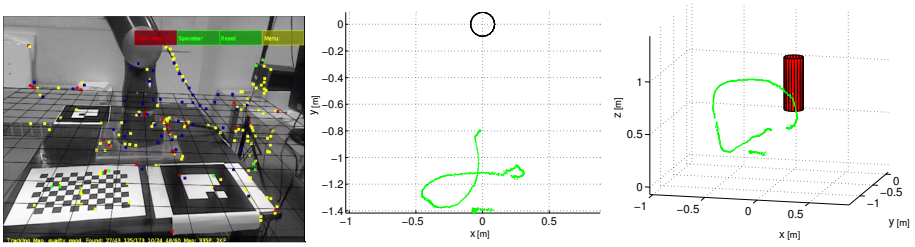
each point together with some local features of the surface, and takes advantage of the 3D structure around the points for guiding the data association between the two point clouds. Moreover, it is based on a least-squares formulation of the alignment problem, which minimizes an augmented error metric depending on point coordinates as well as on surface characteristics.

We have tested the NICP method for tracking the motion of a depth camera in different operative conditions. When the camera moves slowly (at about 0.14 m/s) in a static environment, Figure 1.1(a) shows that clearly NICP is able to track well the camera motion. When the camera moves slightly faster than before (at about 0.2 m/s, i.e., 40% faster) but again in a static environment, there is a duplication of some parts in the 3D map, as shown in Fig. 1.1(b), which implies an inaccurate camera pose estimation. In the last test, the camera was held at rest in a dynamic environment. Figure 1.1(c) shows that different pose estimations that do not reflect the static condition of the real camera. As a result, NICP is not suitable for our goal unless the user (carrying the camera) moves very slowly in an environment which is otherwise static.

1.3 PTAM

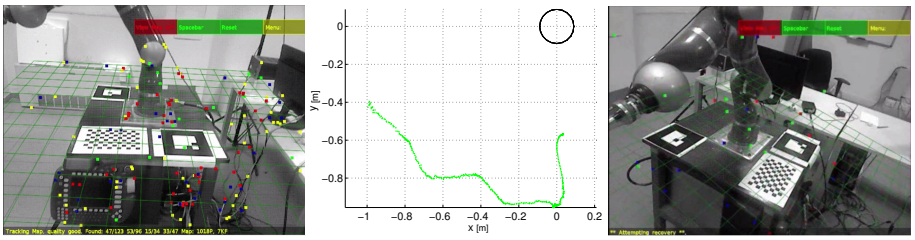
The second method tested was Parallel Tracking and Mapping (PTAM) Klein and Murray [2007]. Although PTAM was originally designed for Augmented Reality (AR), its parallel framework enables fast camera localization in a small, but otherwise unknown environment. This vision-based method does not require markers, pre-made maps, or known templates.

We have used PTAM to estimate the pose of a RGB camera moving around the desired robot workspace. An initialization phase, in which the camera must be translated between the first two key-frames, is mandatory before the tracking phase can start. In a first test (Fig. 1.2), the camera keeps moving in the same field of view seen in the initialization phase and the method is able to track the camera motion satisfactorily. In the second test (Fig. 1.3), starting from the same previous initialization view, the camera was moved around the whole



(a) (b) (c)

Figure 1.2: First PTAM test: (a) represents the tracking map and the features; in the top view (b) and 3D-view (c), the green dots represent the estimated path of the camera and the circle/cylinder denotes the robot position.



(a) (b) (c)

Figure 1.3: Second PTAM test: (a) initialization view; (b) path of the camera; (c) camera view when the method fails.

workspace. The method fails to estimate the camera pose as soon as its field of view exits from the one covered in the initialization phase (thus, the method strongly depends on this phase). As a result, PTAM is a good method for our tracking purposes only when the environment is relatively small and quasi-static.

1.4 ARToolKit

The Third camera tracking test was done with the ARToolKit library Kato and Billinghurst [1999], which is mainly used for developing AR applications. The algorithms in this computer vision library produce in fact a good solution to the problem of calculating in real time the user's viewpoint. Adding a simple calibration setup, is can be used aslo to determine the pose of a RGB camera relative to a physical marker, which can be chosen as a 2D black square card with a special pattern.

In our tests, three different markers have been added to the workspace, placed on the supporting table of the robot manipulator (see also Fig. 4.4). Each marker has its own features (location, size, and pattern). Using the associated homogeneous transformation matrices, it is then easy to obtain online the pose of a moving camera with respect to the world frame. During the multiple experiments done, the ARToolKit performance was extremely good, as long as at least one marker was found in the camera field of view. In Fig. 1.4, the green

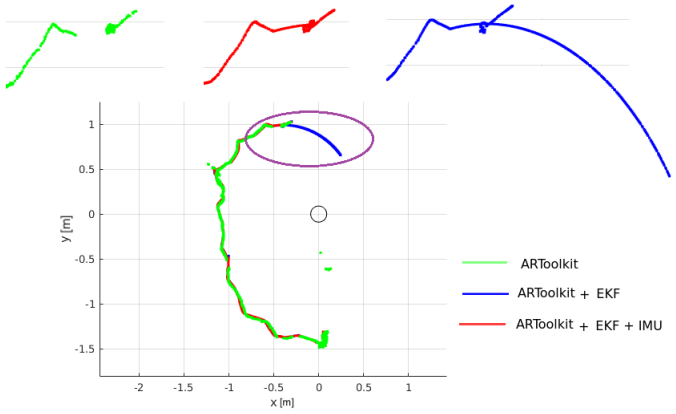


Figure 1.4: The estimated trajectory of a moving camera obtained using the ARToolkit method alone (green line), and with two additional enhancement techniques (blue line = with EKF; red line = with EKF and IMU). The three top figures are expanded views of the paths inside the violet ellipse. For clarity, just a 2D projection on a horizontal plane is shown, instead of the full 3D trajectories.

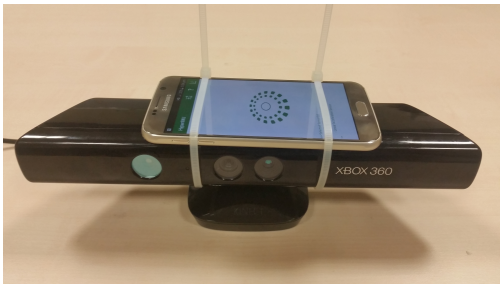


Figure 1.5: A smart phone attached to a Kinect sensor. For ARToolKit method, the RGB sensor of the Kinect, and the accelerometer of the smart phone are used to enhance the camera pose estimation.

line represents the path of the camera estimated with the ARToolKit method in one of the experiments. When there is no marker in the view or when this is not sufficiently clear, discontinuous tracts appear along the estimated motion path together with a number of outlier points. To address this problem, an Extended Kalman Filter (EKF) can be used in the processing of visual data, and a (cheap) Inertial Measurement Unit (IMU) can be added to the camera hardware, i.e. the accelerometer of a smart phone in Fig. 1.5. When the camera is in motion, the EKF eliminates discontinuities and outliers. However, when the camera stops and no marker is in the field of view, the EKF will return false camera pose estimations, i.e., the extra blue line in Fig. 1.4. This behavior is discarded when resorting to the IMU estimation, see the red line in Fig. 1.4.

The previous enhanced system has a simple setup and an easy initialization phase, works at the same camera frame acquisition rate (i.e; 30 Hz), and returns accurate camera pose estimations relative to the desired world reference

Figure 1.6: The Oculus Rift HMD and its tracking sensor.

frame (note that the NICP and PTAM methods provide instead pose estimates expressed with respect to their initial frame).

1.5 Oculus Rift

The Oculus system in Fig 1.6 is developed to provide a Virtual/Augmented Reality experience by synchronizing the user view in the screen of the Head Mounted Display (HMD) with his head motion in the real world. This is done by estimating on line the six degrees of freedom of the device, including position and orientation represented by roll-pitch-yaw angles, and their first and second derivatives, through a sensor fusion process. Data are combined coming from the micro-electrical-mechanical sensors (MEMS) on the Rift, that include gyroscope, accelerometer, and magnetometer, and from the IR on the tracking sensor. In our application the Head Mounted Display (HMD) together with its tracking sensor could be used also for human head pose localization. The tracking sensor should be located in a static place near to the human motion area, and a simple calibration procedure should be done each time the placement of the tracking sensor is changed. The tracking sensor is able to detect and localize the Rift in a distance range from 0.4 to 2.5 m. Multiple tracking sensors could be used to cover a larger area.

For our application we checked the Oculus localization experimentally through different scenarios. In first case, in Fig. 1.7(a), the Rift was mounted on a standing up human without moving for a duration of 60 s in dynamic environment. In second case, in Fig. 1.7(b), the human was moving during the experiment. Finally, we tested the localization during fast and long duration motions, in Fig. 1.7(c). In all previous experiments, the Rift pose estimation was stable, continues and deterministic. This system has a simple setup and an easy initialization phase, and returns accurate Rift pose estimation relative to the desired world reference frame.

From the obtained results and discussions, the head localization in our application can be done using either the ARToolKit method with both EKF and IMU or the Oculus system. The first option is cheaper, while with Oculus system, the mixed-reality user interface can be integrated.

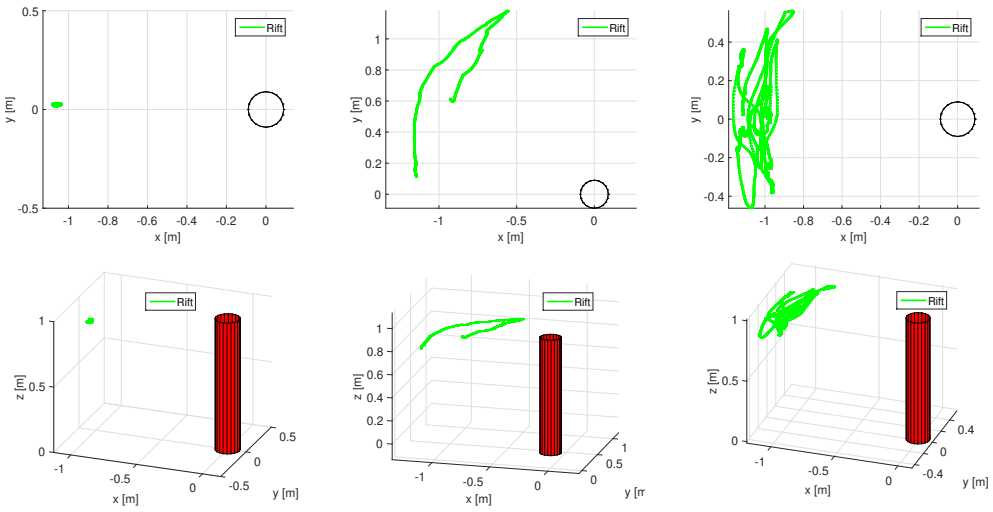


Figure 1.7: The estimated trajectory of a moving Oculus Rift during three human actions: (a) static; (b) slow motion; (c) fast and long motion. [top] The top view in 2D, and [bottom] for the 3D-view.

1.6 Mixed-Reality Interface

To let the operator aware about the active robot task, and give him the possibility to command the robot directly and efficiently, we propose to add a mixed-reality interface to the Oculus HMD. For this, a ZED-Mini stereo camera is mounted to the Rift as in Fig. 1.8. Using Unity cross-platform, the surrounding workspace of the operator can be rendered in the HMD screen and augmented with any useful information about the robot behavior and any desired optional commands. In figure 1.9, the whole hardware setup for the mixed-reality experience is shown.

For our proposed application, we designed a simple interface in Fig. 1.10 which consists of a static menu with four buttons represent the available collaboration modes. The user can switch between them using the Rift controller. The first mode is "Follow", where the robot should track a dynamic target position with respect to the human-head while pointing to it with a relaxed angle 5° or 90°. In the "Circle" mode, the robot should achieve a variable circle that centered on a dynamic position w.r.t human-head, and placed on a plane perpendicular to the line of sight of the human. Also, the user can determine the desired pointing angle. The option "Stop" will command the robot with the last computed target point reducing then the residual errors to zero, and finally remaining at rest. The last gray option is to choose between two pointing angles. After selecting the desired mode, the corresponding button is highlighted. More details about the desired robot tasks are in Chapter 2.

Figure 1.8: ZED-Mini stereo camera mounted on the Oculus Rift.

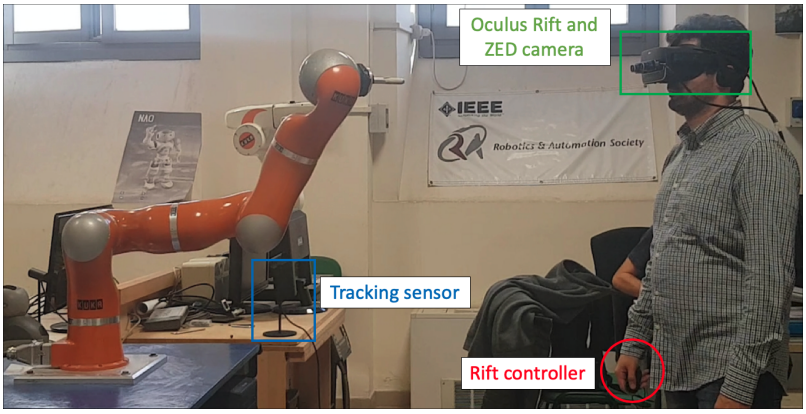


Figure 1.9: The hardware setup to achieve a mixed-reality experience during human-robot collaboration.

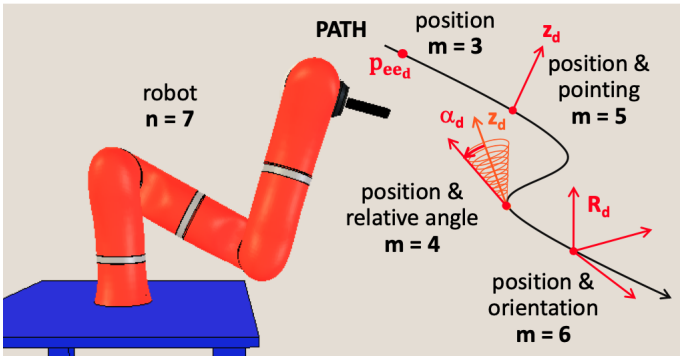
Figure 1.10: The mixed-reality user interface on the Rift HMD screen lenses.

2 Coordination Tasks

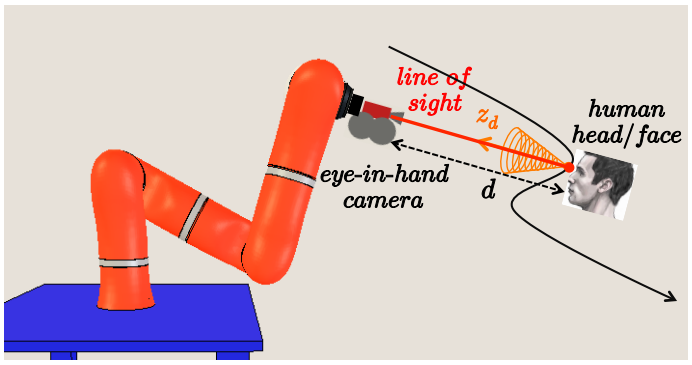
2.1 Introduction

For a robot with n joints, we can define a m-dimensional task to be executed. If $n > m$, the robot will be kinematically redundant for the given task. Figure 2.1(a) illustrates preliminarily some typical motion tasks, with their dimension m: following the position of a reference point $\boldsymbol{p}_{ee_d}$ along a path ($m = 3$), adding to this also a desired pointing direction defined by a unit vector $\boldsymbol{z}_d$ (in total $m = 5$), or specifying in addition a complete desired orientation through a moving reference frame ($m = 6$). Further, we introduce two possibilities that relax the pointing task, while still keeping some control on it: the actual pointing direction of the robot end effector (defined by a unit vector $\boldsymbol{z}_e$) is allowed to stay within a cone with apex at $\boldsymbol{p}_{ee_d}$, axis $\boldsymbol{z}_d$, and apex angle $\alpha > 0$. This is indeed an inequality constraint which increases the task dimension intermittently. However, inequalities are more cumbersome to be handled within the kinematic task formalism Chiaverini et al. [2008]. Another way is to consider a slightly stronger definition by imposing that the relative angle between $\boldsymbol{z}_e$ and $\boldsymbol{z}_d$ takes a (small) constant value α_d: pointing in a direction that belongs to the cone surface is then a one-dimensional task. Combining this with the positional task gives $m = 4$.

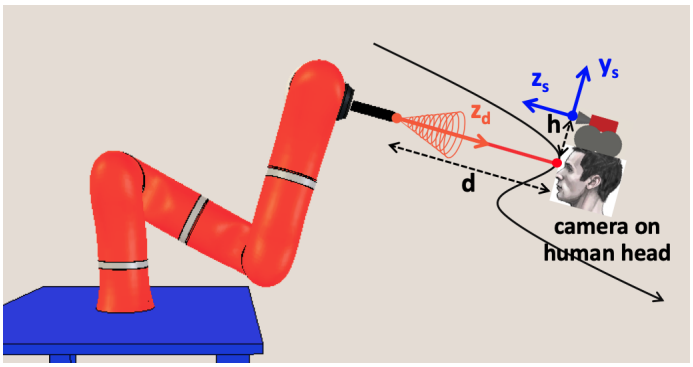
With the above in mind, a visual coordination task between the human (head/face) and the robot can be formulated in two almost symmetric ways. In Fig. 2.1(b), a camera is placed on the end effector (*eye-in-hand*) of the robot, whose motion needs to be controlled so as to search for the human face, point at it (with the above defined relative angle relaxation), and follow the human motion at a desired distance d along the camera line of sight. In this case, the pose of the camera is known from the robot direct kinematics. Alternatively, a camera/sensor can be mounted on the human head (say, on a helmet at height h above the eyes) as in Fig. 2.1(c). In this case, the moving (*eye-to-hand*) camera needs to be continuously localized with respect to the world frame as described in Chapter 1. The robot will receive this information and move accordingly in



(a)



(b)



(c)

Figure 2.1: (a) Different types of Cartesian motion tasks, with their dimension m. (b) Visual coordination task with the camera on the robot, pointing at the moving human head/face. (c) Camera on the moving human head, with the robot end-effector pointing at it. The cones represent the relaxation of the pointing task by some relative angle α_d.

order to achieve coordination. This is the situation considered in this thesis, with the coordination task being of dimension $m = 4$ in case of equality constraint for relaxed pointing task and $m = 3$ or $m = 4$ in case of inequality constraint,

while the robot joint space is of dimension $n = 7$ for a KUKA LWR robot in Appendix A.

In the following sections, the desired coordination tasks of different collaboration modes for our application are defined. Also, a suitable workspace task limits are proposed with its corresponding effects.

2.2 Positional Task

Consider a desired task $\boldsymbol{p}_{ee_d} \in \mathbb{R}^3$ defined for the robot end-effector position,

$$\boldsymbol{p}_{ee} = \boldsymbol{k}(\boldsymbol{q}) \quad \Rightarrow \quad \dot{\boldsymbol{p}}_{ee} = \boldsymbol{J}_{p_{ee}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the robot configuration, $\boldsymbol{k}(.)$ is the direct kinematics, and $\boldsymbol{J}_{p_{ee}} = \partial \boldsymbol{k}/\partial \boldsymbol{q}$ is the $3 \times n$ Jacobian matrix for this task. In this case, the positional error can be defined as $\boldsymbol{e}_{p_{ee}} = \boldsymbol{p}_{ee_d} - \boldsymbol{k}(\boldsymbol{q}) \in \mathbb{R}^3$. For our contactless collaboration, three different positional tasks are defined as follows.

2.2.1 Human-Head Following

As shown in Fig. 2.2, the tip of the robot should follow a desired Cartesian point defined as

$$\boldsymbol{p}_{ee_d} = \boldsymbol{p}_{coord}(t) = \boldsymbol{p}_s(t) + {}^r\boldsymbol{R}_s \begin{pmatrix} 0 \\ h \\ d \end{pmatrix}, \tag{2.2}$$

which is attached to the moving camera/sensor, displaced by $h \geq 0$ along the camera frame axis $\boldsymbol{y}_s$, and located at a distance $d > 0$ along its $\boldsymbol{z}_s$ axis, where ${}^r\boldsymbol{R}_s$ is the rotation matrix between the sensor frame and the world reference frame. In case the localization sensor is worn on the human head (e.g. Oculus Rift) then $h = 0$, as in Fig. 2.3. This task is corresponding to the command "Follow" of the mixed-reality interface.

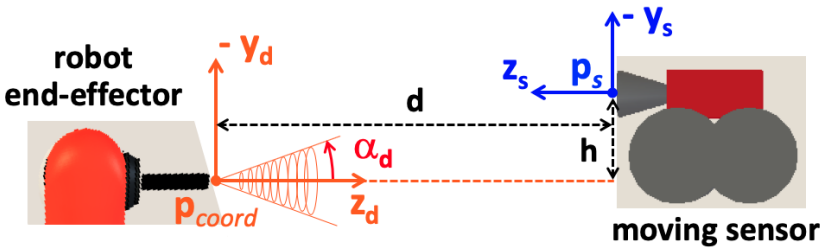


Figure 2.2: Frames and parameter definitions for the coordination task. Here, $\boldsymbol{z}_e = \boldsymbol{z}_d$ yielding $\alpha = 0$.

2.2.2 Variable Circular Task

The second positional task, in Fig. 2.3, is to follow a circular path by the robot *end-effector*. The circle radius is $r = 0.2$ m and its center is at the point $\boldsymbol{p}_{coord}(t)$

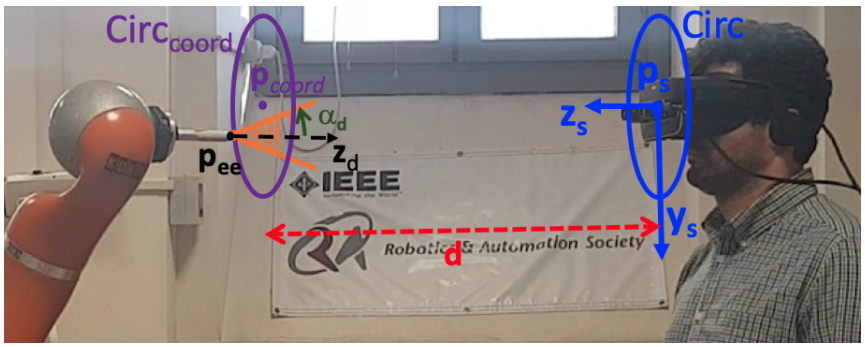


Figure 2.3: Frames and parameter definitions for the desired circular coordination task.

that is also attached to the moving sensor in the same way of the previous task. In this case, the unit vector $\boldsymbol{z}_d$ (i.e. always parallel to $\boldsymbol{z}_s$) should be orthogonal on the desired circle

$$\boldsymbol{p}_{ee_d} = Circ_{coord}(s(t)) = Circ(s(t)) + {}^{r}\boldsymbol{R}_s \begin{pmatrix} 0 \\ h \\ d \end{pmatrix}, \tag{2.3}$$

where

$$Circ(s(t)) = \boldsymbol{p}_s(t) + r\ (cos(s(t))\ \boldsymbol{u} + sin(s(t))\ \boldsymbol{n}), \tag{2.4}$$

where $\boldsymbol{n}$ and $\boldsymbol{u}$ are any two orthonormal vectors to $\boldsymbol{z}_s$, and $s(t)$ is the path parameter. This task is corresponding to the command "Circle" of the mixed-reality interface. In (2.2) and (2.3) the $\boldsymbol{p}_{ee_d}$ is always updated according to the human-head motion localized as in Chapter 1.

The last positional task is corresponding to the command "Stop". The robot should regulate to the last computed desired point $\boldsymbol{p}_{ee_d}$ and remains at rest.

2.3 Pointing Task

Adding a classical 3D pointing task ($m = 2$) or a complete orientation task ($m = 3$) to the positional one, the task dimension would reach $m = 5$ or $m = 6$, respectively. For a standard industrial manipulator with $n = 6$ joints, this would imply that just one or no additional dof is left to the robot in order to achieve other tasks, i.e. collision avoidance. To milden this situation, in our framework we have proposed the use of a relaxed pointing task which requires one additional dof ($m = 3 + 1 = 4$) continuously using an equality constraint Khatib et al. [2017], or transversely in the inequality case Khatib et al. [2019b].

Denoting by $\boldsymbol{q} \in \mathbb{R}^n$ the joint coordinates of the robot. The relaxed pointing task is specified by a constant relative angle $\alpha_d \in \mathbb{R}$ (e.g., $\alpha_d = 5°$) between a unit vector $\boldsymbol{z}_d(t)$ and the $\boldsymbol{z}_e(\boldsymbol{q})$ axis of the frame attached to the robot *end-effector* (the third column of the rotation matrix $\boldsymbol{R}_e(\boldsymbol{q})$ relative to the world frame). In this case, the pointing task can be represented as a cone has its apex located at $\boldsymbol{p}_{ee_d}(t)$, with an apex angle α_d and an unit axis $\boldsymbol{z}_d(t)$ with a desired orientation. For our application, $\boldsymbol{p}_{ee_d}(t)$ is determined from the estimated head

pose, and $\boldsymbol{z}_d(t)$ is ideally pointing at the human eyes as shown in Figs. 2.2 and 2.3 where

$$\begin{aligned} \boldsymbol{z}_d(t) &= \boldsymbol{R}_{e_d}(t) \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T, \\ \boldsymbol{R}_{e_d}(t) &= \boldsymbol{R}_s(t) \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}. \end{aligned} \tag{2.5}$$

The *end-effector* (EE) unit axis may point only approximately toward the human head and, in fact, should only belong to the surface of the pointing cone. In this case, the EE pointing can be expressed as

$$p_{rp}(\boldsymbol{q}) = \boldsymbol{z}_d^T \boldsymbol{z}_e(\boldsymbol{q}) = \cos(\alpha), \tag{2.6}$$

where for a constant desired relative angle $\alpha_d > 0$,

$$p_{rp_d} = \cos(\alpha_d) \tag{2.7}$$

and the error is computed as

$$e_{rp} = p_{rp_d} - p_{rp} \in \mathbb{R}. \tag{2.8}$$

Beside using the equality constraint in (2.7), we propose further relaxation by defining the desired pointing angle using the following inequality

$$0 \leq \alpha \leq \alpha_d. \tag{2.9}$$

In this case, the desired task (2.7) can be written as

$$\begin{aligned} 1 \geq cos(\alpha) \geq cos(\alpha_d), \\ 1 \geq \boldsymbol{z}_d^T \boldsymbol{z}_e(\boldsymbol{q}) \geq cos(\alpha_d), \end{aligned} \tag{2.10}$$

where the unit axis $\boldsymbol{z}_e(\boldsymbol{q})$ is allowed to point toward any Cartesian point belongs to the surface of the coin or any point inside it. Through our mixed-reality interface in Sec.1.6, the user can determine the desired α_d to be 5° or 90°.

To derive the Jacobian associated to the relaxed pointing task, assume that α is constant. Differentiating eq. (2.6) yields

$$\frac{dp_{rp}(\boldsymbol{q})}{dt} = \frac{d\boldsymbol{z}_d}{dt}^T \boldsymbol{z}_e(\boldsymbol{q}) + \boldsymbol{z}_d^T \frac{d\boldsymbol{z}_e(\boldsymbol{q})}{dt} = 0. \tag{2.11}$$

Being the time derivative of $\boldsymbol{R}_e(\boldsymbol{q})$ Siciliano et al. [2010]

$$\frac{d\boldsymbol{R}_e(\boldsymbol{q})}{dt} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}_e(\boldsymbol{q}),$$

where $\boldsymbol{S}$ is the 3×3 skew-symmetric matrix representing the vector ($\times$) product and $\boldsymbol{\omega}$ denotes the angular velocity of frame $\boldsymbol{R}_e(\boldsymbol{q})$, we have

$$\frac{d\boldsymbol{z}_e(\boldsymbol{q})}{dt} = -\boldsymbol{S}(\boldsymbol{z}_e(\boldsymbol{q}))\boldsymbol{\omega} = \boldsymbol{S}^T(\boldsymbol{z}_e(\boldsymbol{q}))\, \boldsymbol{J}_{o_{ee}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.12}$$

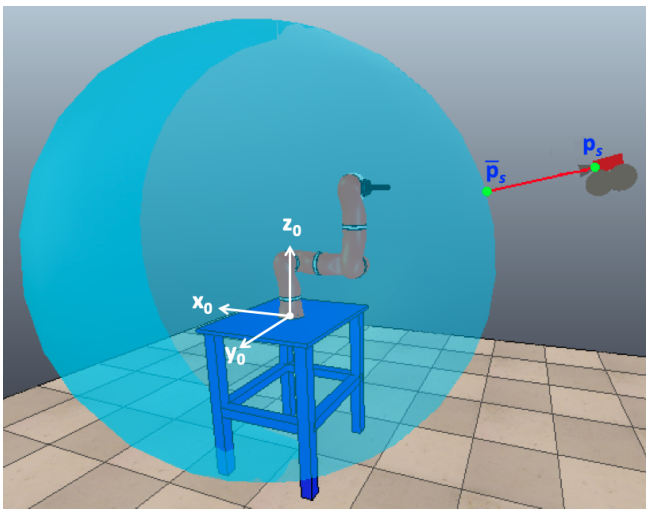


Figure 2.4: The blue sphere represents a limit for the coordination task. When the sensor is outside the sphere in the position $\boldsymbol{p}_s$, the projected position $\overline{\boldsymbol{p}}_s$ on the boundary of the sphere will be used as reference to compute the desired following task.

where $\boldsymbol{J}_{o_{ee}}(\boldsymbol{q})$ is the $3 \times n$ Jacobian that maps the joint velocity $\dot{\boldsymbol{q}}$ to the *end-effector* angular velocity $\boldsymbol{\omega}$. Substituting (2.12) in (2.11), we obtain

$$\frac{d\boldsymbol{z}_d}{dt}^T \boldsymbol{z}_e(\boldsymbol{q}) + \boldsymbol{J}_{rp}(\boldsymbol{q})\dot{\boldsymbol{q}} = 0, \tag{2.13}$$

where

$$\boldsymbol{J}_{rp}(\boldsymbol{q}) = \boldsymbol{z}_d^T \, \boldsymbol{S}^T(\boldsymbol{z}_e(\boldsymbol{q})) \, \boldsymbol{J}_{o_{ee}}(\boldsymbol{q}) \tag{2.14}$$

is the $1 \times n$ Jacobian matrix associated to the task defined through (2.6). From (2.13), our relaxed pointing task is executed in nominal conditions by choosing a $\dot{\boldsymbol{q}}$ that satisfies

$$\boldsymbol{J}_{rp}(\boldsymbol{q})\dot{\boldsymbol{q}} = -\dot{\boldsymbol{z}}_d^T(t) \, \boldsymbol{z}_e(\boldsymbol{q}) = \dot{p}_{rp_d}. \tag{2.15}$$

2.4 Task Limit Sphere

During the execution of a complete coordination task, some limits may be reached as well as task singularities encountered. While singularities can be handled properly by the DLS technique Chiaverini et al. [2008] for Jacobian computation, the occurrence of task limits deserve a special treatment.

For this situation, we propose in algorithm 1, a special treatment to avoid any operational task limit. A Cartesian boundary is defined by a virtual sphere $\boldsymbol{S}$ (of radius $r_s = 1$ [m] and center $\boldsymbol{c}_s$ at the second robot joint in the case of our 7-dof KUKA), as shown in Fig. 2.4. If the sensor position is outside the sphere and an intersection exists between its line of sight and the sphere, then the desired EE position for the following task will be relocated accordingly at the intersection point. Otherwise, if there is no intersection, the robot will be

Algorithm 1 Coordinated task limit check

$\boldsymbol{l} = \frac{\boldsymbol{z}_s}{\|\boldsymbol{z}_s\|}$, $a = \| \boldsymbol{l} \|^2$, $b = 2\boldsymbol{l}(\boldsymbol{p}_s - \boldsymbol{c}_s)$, $c = \| \boldsymbol{p}_s - \boldsymbol{c}_s \|^2 - r_s^2$
incidence $= b^2 - 4ac$
if incidence ≤ 0 **then**
  use the last estimate pose
else
  $d_1 = \frac{-2b+\sqrt{\text{incidence}}}{2a}$, $d_2 = \frac{-2b-\sqrt{\text{incidence}}}{2a}$
  $\overline{\boldsymbol{p}}_1 = \boldsymbol{p}_s + d_1\boldsymbol{l}$, $\overline{\boldsymbol{p}}_2 = \boldsymbol{p}_s + d_2\boldsymbol{l}$
  $\boldsymbol{l}_1 = \frac{\overline{\boldsymbol{p}}_1 - \boldsymbol{p}_s}{\|\overline{\boldsymbol{p}}_1 - \boldsymbol{p}_s\|}$, $\boldsymbol{l}_2 = \frac{\overline{\boldsymbol{p}}_2 - \boldsymbol{p}_s}{\|\overline{\boldsymbol{p}}_2 - \boldsymbol{p}_s\|}$
  if $\| \boldsymbol{p}_s - \boldsymbol{c}_s \|^2 < r_s^2$ **then**
    if $\{\boldsymbol{l}_1 == \boldsymbol{l}$ and $\| \boldsymbol{p}_1 - \boldsymbol{p}_s \|^2 > r_s^2\}$ or $\{\boldsymbol{l}_2 == \boldsymbol{l}$ and $\| \boldsymbol{p}_2 - \boldsymbol{p}_s \|^2 > r_s^2\}$ **then**
      update the task with the current pose estimation
    else
      use the last estimate pose
    end if
  else if $\| \overline{\boldsymbol{p}}_1 - \boldsymbol{p}_s \|^2 > \| \overline{\boldsymbol{p}}_2 - \boldsymbol{p}_s \|^2$ and $\boldsymbol{l}_2 == \boldsymbol{l}$ **then**
    update the task according to $\overline{\boldsymbol{p}}_2$
  else if $\| \overline{\boldsymbol{p}}_2 - \boldsymbol{p}_s \|^2 > \| \overline{\boldsymbol{p}}_1 - \boldsymbol{p}_s \|^2$ and $\boldsymbol{l}_1 == \boldsymbol{l}$ **then**
    update the task according to $\overline{\boldsymbol{p}}_1$
  else
    use the last estimate pose
  end if
end if

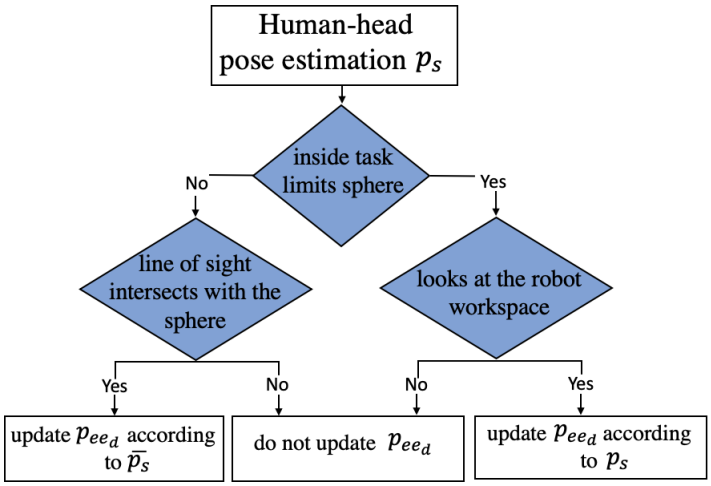


Figure 2.5: Different situations for the human-head pose, and how it get modified according to the task limits sphere.

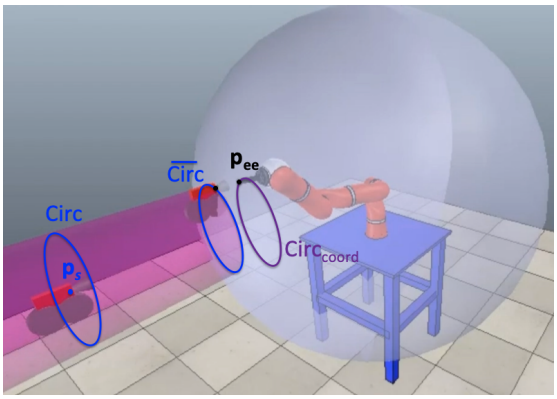


Figure 2.6: The desired circular task is defined according to the projected circle on the task limits sphere.

commanded to regulate for the last computed desired task reducing the residual errors to zero, and finally remaining at rest.

Another situation which is beyond the task limits occurs when the sensor is inside the sphere but looking to the outside of the robot workspace. Also in this case the robot will be commanded for a regulation as before. As soon as the position and pointing direction of the sensor become again feasible for the task, motion control is resumed with the updated desired tasks. A scheme that illustrates how we deal with the task limits in different situations is presented in Fig. 2.5.

The same procedure is done in case of the desired circular task in Fig. 2.3, where each point of the circle outside the limits, will be projected on the sphere, as in Figure 2.6. Then the desired EE task will be computed according to the projected point. The different coordinations tasks are illustrated, using V-REP simulations, in a video at this link.

3 Collision Avoidance Tasks

3.1 Introduction

In our contactless collaboration, the human operator is supposed to work close to a robot that may live in an environment cluttered with obstacles. Therefore, both human safety and robot integrity should always be guaranteed. Collision avoidance problem can be separated into two sub-problems. The first one is the definition of the avoidance task in the Cartesian space (which will be introduced in this chapter), and the other one is how to control this task together with other robot main tasks (in Chapter 4). The main process for avoiding any obstacle is the related distance computations. In the following sections, we summarized the depth space approach Flacco et al. [2015b] which we used for distance computations in our application. Also, different definitions for the collision avoidance task are introduced.

3.2 Distance Computation

The critical operation for any collision avoidance approach is to compute the distances between all interest objects efficiently in the real-time. For this, we resort to the depth space approach in Flacco et al. [2015b] to evaluate the distances between a number of control points (including the EE) selected along the robot arm and any obstacle (including the human operator) in the workspace. In this work, we considered nine control points along the robot body, four of them located between the third and fourth joints. While, the next four points are located between the fourth and sixth joints. The last control point is located on the robot TCP. The distance computations are done using a single RGB-D camera only, where the point-to-object distances are evaluated directly in the depth space of the sensor, allowing large saving in the computation time, Fig. 3.1.

Consider an obstacle point $\boldsymbol{o}$ and a generic control point $\boldsymbol{c}$, which are represented in the depth space as ${}^{D}\boldsymbol{o} = (o_x \ o_y \ d_o)^T$ and ${}^{D}\boldsymbol{c} = (c_x \ c_y \ d_c)^T$, where the first two coordinates represent the position of the projected point in the 2D image plane of the sensor, and the third coordinate represents the depth of this point as seen from the sensor. To compute the Cartesian distance $D(\boldsymbol{c}, \boldsymbol{o})$

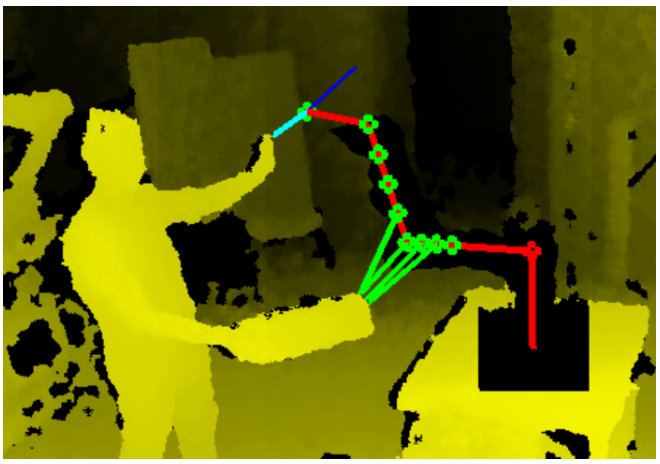


Figure 3.1: Snapshot of a depth image superposed with the computed distances between control points (green circles) on the robot arm and close objects located in the surveillance area. The picture shows also the computed distance between the end-effector and the human hand obstacle (cyan line), as well as its associated repulsive vector (blue line).

between points $\boldsymbol{c}$ and $\boldsymbol{o}$, two different cases are considered.

- If $d_o > d_c$, then

$$\begin{aligned} D(\boldsymbol{c},\boldsymbol{o}) \backsimeq D_D({}^D\boldsymbol{c},{}^D\boldsymbol{o}) &= \sqrt{a_x^2+a_y^2+a_z^2}, \\ a_x &= \frac{(o_x-\rho_x)d_o-(c_x-\rho_x)d_c}{l\ s_x}, \\ a_y &= \frac{(o_y-\rho_y)d_o-(c_y-\rho_y)d_c}{l\ s_y}, \\ a_z &= d_o-d_c, \end{aligned} \tag{3.1}$$

  where D_D is the distance in the depth space, l is the focal length of the depth camera, (s_x, s_y) are the pixel sizes and (ρ_x, ρ_y) are the pixel coordinates of the image plane center.

- If $d_o \leq d_c$, the depth of the obstacle is assumed conservatively to be equal to the depth of the control point ($d_o = d_c$), and the distance is then computed using (3.1).

The generic unit vector between $\boldsymbol{c}$ and $\boldsymbol{o}$ is given by

$$\boldsymbol{u}({}^D\boldsymbol{c},{}^D\boldsymbol{o}) = \frac{(a_x\ a_y\ a_z)^T}{D_D({}^D\boldsymbol{c},{}^D\boldsymbol{o})}. \tag{3.2}$$

To evaluate distances between $\boldsymbol{c}$ and all obstacle points sufficiently close to it, the distance evaluation is applied only to pixels in the depth image plane within a desired region of surveillance.

3.3 Task Definition for Collision Avoidance

Potential field approach is a common way to define the collision avoidance task Khatib [1986a], where the key tool is how the repulsive vectors between the interest robot points and obstacles will be computed. The desired reaction magnitude is usually based on a reverse relation with the nearest object. While the preferred direction can be considered as the mean/minimum for the direction vectors of all objects in the surveillance area of each interest point, see for examples Flacco et al. [2015a] and Lacevic et al. [2013].

In our case, the magnitude of the desired reaction takes into account the *nearest* object to each control point as

$$u(\boldsymbol{c}) = \frac{u_{max}}{1 + e^{(D_{min}(\boldsymbol{c})(2/\rho)-1)\gamma}}, \tag{3.3}$$

where u_{max} is the maximum admissible magnitude, the factor $\gamma > 0$ shapes the exponential decay rate, and

$$D_{min}(\boldsymbol{c}) = \min_{i=1}^{h} D(\boldsymbol{c}, \boldsymbol{o}_i), \tag{3.4}$$

where h is the number of obstacle points in the surveillance area of $\boldsymbol{c}$. In practice, for a large value of γ, the magnitude u of the repulsive vector approximately equals u_{max} when $D_{min}(\boldsymbol{c}) \approx 0$, and approximately vanishes when the distance reaches the ρ value, where beyonds ρ the $u(\boldsymbol{c})$ is not defined.

The direction of the desired reaction is evaluated either as the *mean* vector ${}^{c}\boldsymbol{u}_{mean}(\boldsymbol{c})$, or *minimum* vector ${}^{c}\boldsymbol{u}_{min}(\boldsymbol{c})$ of the unit distance vectors between each control point $\boldsymbol{c}$ and all points of objects in the surveillance area where

$${}^{c}\boldsymbol{u}_{mean}(\boldsymbol{c}) = \frac{1}{h}\sum_{i=1}^{h} \boldsymbol{u}({}^{D}\boldsymbol{c}, {}^{D}\boldsymbol{o}_i), \tag{3.5}$$

and

$${}^{c}\boldsymbol{u}_{min}(\boldsymbol{c}) = \boldsymbol{u}({}^{D}\boldsymbol{c}, {}^{D}\boldsymbol{o}_{min}), \tag{3.6}$$

where $\boldsymbol{o}_{min}$ is the nearest obstacle to $\boldsymbol{c}$. In this case, the repulsive vector associated to each control point can be defined in the base reference frame as

$$\boldsymbol{u}(\boldsymbol{c}) = {}^{r}\boldsymbol{R}_c \, u(\boldsymbol{c}) \, {}^{c}\boldsymbol{u}_{mean|min}(\boldsymbol{c}), \tag{3.7}$$

where ${}^{r}\boldsymbol{R}_c$ is the rotation matrix between the depth camera frame and the base reference frame.

Using (3.7), each control point along the robot has its corresponding repulsive vector to be used for collision avoidance task. These repulsive vectors can be considered as desired Cartesian forces, accelerations or velocities according to the applied motion control. In Flacco et al. [2015a], all repulsive vectors for the robot body control points, but the EE, are treated as Cartesian constraints with artificial forces and transformed approximately into hard joint velocity or acceleration constraints which used during the control of the desired tasks. Instead, the authors in Lacevic et al. [2013] considered the collision avoidance

task as a joint space desired velocity. All repulsive vectors are transformed approximately to the joint space and then accumulated algebraically to finally be projected it in the null space of the robot main task as follows

$$\dot{\boldsymbol{q}} = \delta \boldsymbol{J}^{\#} \dot{\boldsymbol{p}} + (\boldsymbol{I} - \delta \boldsymbol{J}^{\#} \boldsymbol{J}) \dot{\boldsymbol{q}}_c, \tag{3.8}$$

where

$$\dot{\boldsymbol{q}}_c = \sum_{k=1}^{y} \boldsymbol{J}_{c_k}^T \boldsymbol{u}(\boldsymbol{c}_k), \tag{3.9}$$

where y is the total number of robot control points, and $\boldsymbol{J}_c$ is the analytic Jacobian of the direct kinematics for the position of the control point $\boldsymbol{c}$. In (3.8) the parameter δ is used to switch off/on the main task smoothly according to a predefined danger field. Using (3.9), if there are multiple obstacles moving oppositely to the same interest point, the corresponding repulsive vectors will cancel/reduce the effect of each other and the collision could be unavoidable.

Another way is to consider each repulsive vector as a desired Cartesian acceleration

$$\ddot{\boldsymbol{p}}_c = \boldsymbol{u}(\boldsymbol{c}). \tag{3.10}$$

In (3.10), the definition of the collision avoidance task is over deterministic and requires at least three dofs to be handled. Instead, similar to Zlajpah and Nemec [2002], we propose to relax the avoidance task constraints to be $m = 1$. This is done by projecting the repulsive vector in its corresponding unit direction as

$$\begin{aligned} \ddot{p}_n &= \frac{\boldsymbol{u}(\boldsymbol{c})^T}{||\boldsymbol{u}(\boldsymbol{c})||} \ddot{\boldsymbol{p}}_c, \\ \boldsymbol{J}_n &= \frac{\boldsymbol{u}(\boldsymbol{c})^T}{||\boldsymbol{u}(\boldsymbol{c})||} \boldsymbol{J}_c, \end{aligned} \tag{3.11}$$

where $\ddot{p}_n \in \mathbb{R}$ and $\boldsymbol{J}_n \in \mathbb{R}^{1 \times n}$. In this form, the task in (3.10) is modified so that not the direction but only the speed of moving away from the obstacle is specified thus creating only one constraint. According to the motion control algorithm, one could consider single or several relaxed collision avoidance tasks, e.g. the most critical one.

4 Motion Control

4.1 Introduction

In this chapter different motion control algorithm will be introduced and compared to perform the proposed collaboration tasks using different sensors. This will be presented in a progressive way.

In the first two sections, we present the design of kinematic control laws that handle the coordination task, $\boldsymbol{p}_{ee_d}$, introduced in Chapter 2, including the relaxed definition of the pointing task, p_{rp_d}. Two strategies are pursued: the first one is by augmentation of the positional task ($m_{p_{ee}} = 3$) with the relaxed pointing task ($m_{rp} = 1$), as detailed in Sec. 4.2; the second one is through a suitable projection of the relaxed pointing task into the null space of the positional one (in Sec. 4.3). Comparative results of V-REP simulations and of an experiment with a KUKA LWR arm are reported.

In the second stage, the collision avoidance task, $\boldsymbol{p}_{c_k}$, for y different control points along the robot body is included, where $k \in [1, 2, .., y]$. For this, a special consideration is done for the control point of *end-effector*, $\boldsymbol{p}_{c_{ee}}$. The robot desired tasks, i.e. $(\boldsymbol{p}_{ee_d}, p_{rp_d}, \boldsymbol{p}_{c_{ee}}, \boldsymbol{p}_{c_1}, .., , \boldsymbol{p}_{c_y})$, are handled using two different task priority control algorithms. Namely, the SNS algorithm in Sec. 4.5, and the Task Priority Matrix (TPM) in Sec. 4.6. Various simulations using MATLAB, and several experiments are presented.

4.2 Task Augmentation

Let a set of l desired Cartesian tasks, to be achieved in a specific desired priority, be defined as

$$\boldsymbol{p}_i = \boldsymbol{p}_i(t), \quad 0 < i \leq l, \tag{4.1}$$

where $\boldsymbol{p}_i \in \mathbb{R}^{m_i}$. The i-th task has higher priority than the j-th task if $i < j$. In this case, the simplest solution, that does not consider the priority order, can be obtained using the first-order differential inverse kinematics given by Chiaverini

et al. [2008]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{p}}, \tag{4.2}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ represents the joint velocities,

$$\boldsymbol{J} = [\boldsymbol{J}_1^T \quad \dots \quad \boldsymbol{J}_i^T \quad \dots \quad \boldsymbol{J}_l^T]^T, \tag{4.3}$$

is the augmented matrix that contains a Jacobian $\boldsymbol{J}_i \in \mathbb{R}^{m_i \times n}$ related to each desired task, where $\sum_{i=1}^{l} m_i = m \leq n$, and

$$\dot{\boldsymbol{p}} = [\dot{\boldsymbol{p}}_1^T \quad \dots \quad \dot{\boldsymbol{p}}_i^T \quad \dots \quad \dot{\boldsymbol{p}}_l^T]^T. \tag{4.4}$$

Using (4.2), the accuracy of each task execution is according to the dependency between all the tasks. For example, if there is no tasks conflict, the robot can achieve all of them, otherwise the final solution will be the sum of each task contribution. Indeed, solution (4.2) has a good shaping of the inverse kinematic solution where the contribution and the null space for each task can be deduced easily. On the other hand, singularities may appear when there are linear dependencies between the tasks.

The complete coordination task can be realized using Task Augmentation (TA), namely by stacking the Jacobians $\boldsymbol{J}_{p_{ee}}$ from (2.1) and $\boldsymbol{J}_{rp}$ from (2.15) in (4.3). In nominal conditions, a solution can be obtained by choosing a joint velocity $\dot{\boldsymbol{q}}$ that satisfies

$$\dot{\boldsymbol{p}}_{A,d} = \begin{pmatrix} \dot{\boldsymbol{p}}_{ee_d} \\ \dot{p}_{rp_d} \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_{p_{ee}}(\boldsymbol{q}) \\ \boldsymbol{J}_{rp}(\boldsymbol{q}) \end{pmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4.5}$$

where $\boldsymbol{J}_A(\boldsymbol{q})$ is the $4 \times n$ Augmented Jacobian. Assuming that $n > 4 = m$, i.e., the robot is redundant for the augmented task, and that we need to react by feedback to possible positioning error $\boldsymbol{e}_{p_{ee}}$ and/or relaxed pointing error e_{rp} that may arise during task execution, the final control law is defined as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{\#}(\boldsymbol{q}) \left(\begin{pmatrix} \dot{\boldsymbol{p}}_{ee_d} \\ \dot{p}_{rp_d} \end{pmatrix} + \begin{pmatrix} \boldsymbol{K}_P(\boldsymbol{p}_{ee_d} - \boldsymbol{k}(\boldsymbol{q})) \\ k_{rp}(p_{rp_d} - p_{rp}(\boldsymbol{q})) \end{pmatrix} \right), \tag{4.6}$$

where $\boldsymbol{J}_A^{\#}$ is the pseudoinverse of the Jacobian $\boldsymbol{J}_A$ in (4.5), $\boldsymbol{K}_P > 0$ is a 3×3 diagonal gain matrix, and $k_{rp} > 0$ is a scalar gain. Typically, the 3D motion of the human/sensor is not known in advance, and so $\dot{\boldsymbol{p}}_{ee_d}$ and $\dot{p}_{rp_d}$ in (4.6) will not be available to the controller. In practice, these reference velocities are set to zero and robot motion will be driven only by the two errors $\boldsymbol{e}_{p_{ee}}$ and e_{rp}. Indeed, it is also possible to obtain an online prediction of the camera motion (e.g., based on an EKF method, as done in Milighetti et al. [2011]), and include this as the nominal feedforward term in the control law (4.6).

To obtain the pseudoinverse in (4.6), one needs to compute the Singular Value Decomposition (SVD) Golub and Van Loan [1996] of the Jacobian, $\boldsymbol{J}_A = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U} = \text{col}\{\boldsymbol{u}_i\}$ and $\boldsymbol{V} = \text{col}\{\boldsymbol{v}_j\}$ are, respectively, $m \times m$ and $n \times n$ unitary matrices, and $\boldsymbol{\Sigma}$ is a $m \times n$ block matrix, with a leading $m \times m$ diagonal block containing the singular values $\sigma_i \geq 0$ $(i = 1, \dots, m)$ of $\boldsymbol{J}_A$ in decreasing order ($\sigma_h \geq \sigma_k$ for $h < k$), followed by $n - m$ zero columns. It is

$$\boldsymbol{J}_A^{\#} = \boldsymbol{V}\boldsymbol{\Sigma}^{\#}\boldsymbol{U}^T = \sum_{i=1}^{\rho} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T, \tag{4.7}$$

where $\rho \leq m = 4$ is the rank of $\boldsymbol{J}_A$. When the smallest singular value(s) becomes too small (i.e., close to singularities), large joint velocities are being generated. This drawback was addressed in our implementation by the Damped Least Squares (DLS) technique Chiaverini et al. [2008], replacing in (4.7)

$$\frac{1}{\sigma_i} \rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \quad \Rightarrow \quad \boldsymbol{J}_A^{\#} \rightarrow \boldsymbol{J}_A^{\text{DLS}}. \tag{4.8}$$

with

$$\lambda^2 = \begin{cases} 0, & \text{when } \sigma_i \geq \epsilon, \\ \left(1 - (\sigma_i/\epsilon)^2\right)\lambda_{max}^2, & \text{otherwise.} \end{cases}$$

The small parameter $\epsilon \geq 0$ monitors the singular values σ_i and defines the range of the damping action, while $\lambda_{max}^2 > 0$ is the largest damping factor used at singularities.

4.3 Null-Space Projection

To take into account the task priority strictly, while executing the Stack of Task (SoT) in (4.4), the most common way is to use the recursive solution with null space projection Siciliano and Slotine [1991] as follows

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + (\boldsymbol{J}_i \boldsymbol{N}_{i-1})^{\#}(\dot{\boldsymbol{p}}_i - \boldsymbol{J}_i \dot{\boldsymbol{q}}_{i-1}), \tag{4.9}$$

where $\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$. The null space projector is given by

$$\boldsymbol{N}_i = \boldsymbol{N}_{i-1} - (\boldsymbol{J}_i \boldsymbol{N}_{i-1})^{\#} \boldsymbol{J}_i \boldsymbol{N}_{i-1}, \tag{4.10}$$

where $\boldsymbol{N}_0 = \boldsymbol{I}$ is the identity matrix. Using (4.9) the highest priority task, i.e. $i = 1$, is ideally no longer affected by other tasks. On the other hand, when some of the tasks are linearly dependent, the contribution of a lower priority task and the null space of the higher task can not be inferred. This missing information is important to perform a smooth transition during a desired addition, deletion or reordering operation on the tasks.

As a second control strategy, we pursue the null-space design method to consider the positional tracking task defined through (2.1) as the one with highest priority, we accommodate the relaxed point task into a joint velocity $\dot{\boldsymbol{q}}_2 \in \mathbb{R}^n$ which is then projected in the null space of the high-priority task, so as not to affect its execution in case of conflicts. Thus, we define

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_1 = \boldsymbol{J}_{p_{ee}}^{\#}(\boldsymbol{q}) \left(\dot{\boldsymbol{p}}_{ee_d} + \boldsymbol{K}_P \boldsymbol{e}_{p_{ee}}\right) + \boldsymbol{N}_1(\boldsymbol{q}) \dot{\boldsymbol{q}}_2 \tag{4.11}$$

where $\boldsymbol{N}_1 = \boldsymbol{I} - \boldsymbol{J}_{p_{ee}}^{\#} \boldsymbol{J}_{p_{ee}}$. To execute the relaxed pointing task, a Projected Gradient (PG) technique Chiaverini et al. [2008] is followed, choosing vector $\dot{\boldsymbol{q}}_2$ as

$$\dot{\boldsymbol{q}}_2 = -k_g \nabla_{\boldsymbol{q}} H(\boldsymbol{q}), \tag{4.12}$$

with

$$H(\boldsymbol{q}) = \frac{1}{2} e_{rp}^2 = \frac{1}{2} \left(\cos \alpha_d - \boldsymbol{z}_d^T \boldsymbol{z}_e(\boldsymbol{q})\right)^2, \tag{4.13}$$

namely along the negative gradient of the squared norm of the error e_{rp}, taken as objective function $H(\boldsymbol{q}) \geq 0$. The scalar gain $k_g > 0$ in (4.12) is used to shape the convergence rate, and plays a very similar role as k_{rp} in (4.6).

4.4 Task Augmentation vs. Projected Gradient

4.4.1 Comparative Simulations

Several simulations have been run in a ROS environment, integrated with the robot simulator V-REP, using the motion control law (4.6) or (4.11). Figure 4.1 shows in the V-REP scenario some typical robot configurations obtained for different camera poses.

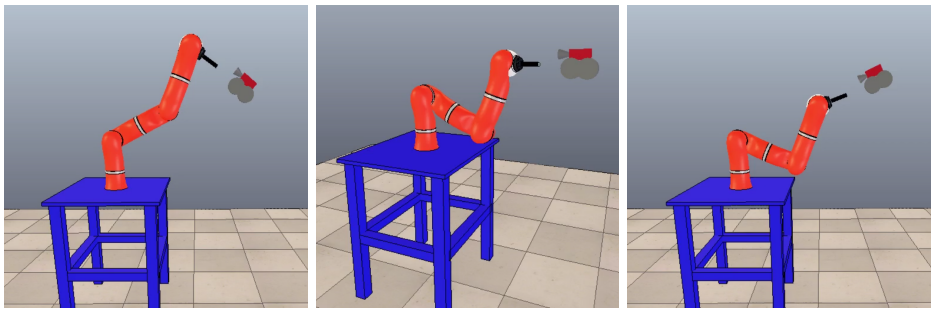


Figure 4.1: Representative robot behaviors achieved during the visual coordination task.

In all the numerical tests, we assumed an ideal localization. The two offsets used to determine the desired position $\boldsymbol{p}_{ee_d}$ and the constant relative angle for the relaxed pointing task (see Fig. 2.2) were chosen as

$$h = 0.05 \text{ [m]}, \qquad d = 0.2 \text{ [m]}, \qquad \alpha_d = 5^\circ. \tag{4.14}$$

The gains in the control law (4.6) were chosen as

$$\boldsymbol{K}_P = \text{diag}\{20, 30, 30\}, \qquad k_{rp} = 20.$$

Moreover, the feedforward terms in (4.6) were absent ($\dot{\boldsymbol{p}}_{ee_d} = \boldsymbol{0}$, $\dot{p}_{rp_d} = 0$), since the camera motion is assumed to be unknown.

In Figs. 4.2–4.3, we report the results obtained for two simple camera motions. In both cases, the robot initial configuration is matched with the initial desired task, i.e., at time $t = 0$, the errors are $\boldsymbol{e}_{p_{ee}}(0) = \boldsymbol{0}$ and $e_{rp}(0) = 0$. In the first simulation, the camera position changes continuously, tracing a circle in the vertical plane at $\boldsymbol{x}_0 = -0.8$ m (parallel to $(\boldsymbol{y}_0, \boldsymbol{z}_0)$, see Fig. 2.4), while the pointing direction is kept constant and horizontal at the value $\boldsymbol{z}_d = (-1, 0, 0)$. Since the motion is relatively slow, both the maximum of the pointing error ($e_{rp,max} = 5.4 \cdot 10^{-4}$) and the maximum norm of the position error ($\|\boldsymbol{e}_{p_{ee}}\|_{max} = 2 \cdot 10^{-3}$) in Fig. 4.2 are very small. In the second simulation, the camera center is kept fixed while $\boldsymbol{z}_d$ oscillates periodically in a horizontal plane around the vertical axis. As a result, the reference position of the displaced target point $\boldsymbol{p}_{ee_d}$ is changing, as shown in Fig. 4.3. The peaks or discontinuities in the errors occur when the x and y coordinates of $\boldsymbol{p}_{ee_d}$ are inverting motion, but the errors remain always small. The results obtained using the PG control method are very similar to those shown with TA control, and are thus not reported.

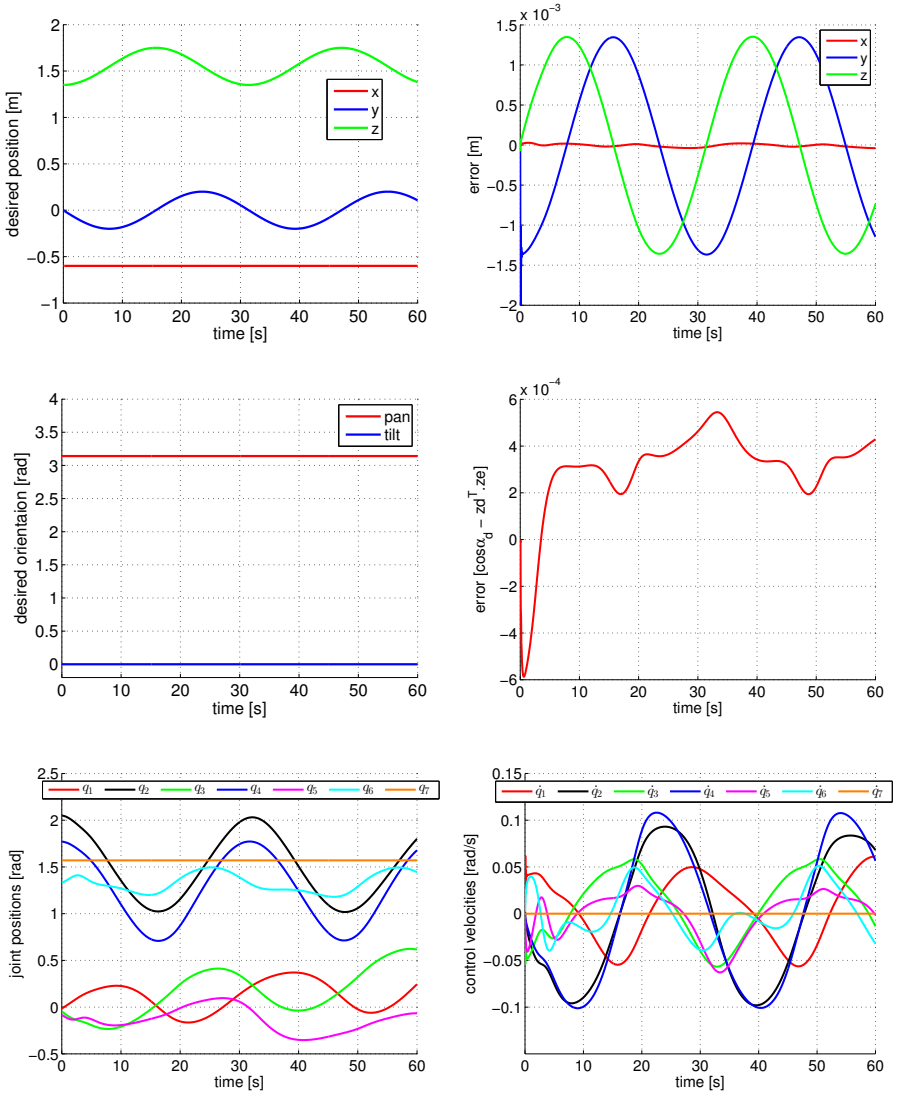


Figure 4.2: Motion control with the TA method in the first simulated task: (top) desired end-effector position $\boldsymbol{p}_{ee_d}$ and position error $\boldsymbol{e}_{p_{ee}}$; (center) desired pointing direction $\boldsymbol{z}_d$, represented by its pan and tilt angles, and relaxed pointing error e_{rp}; (bottom) joint positions and commanded velocities.

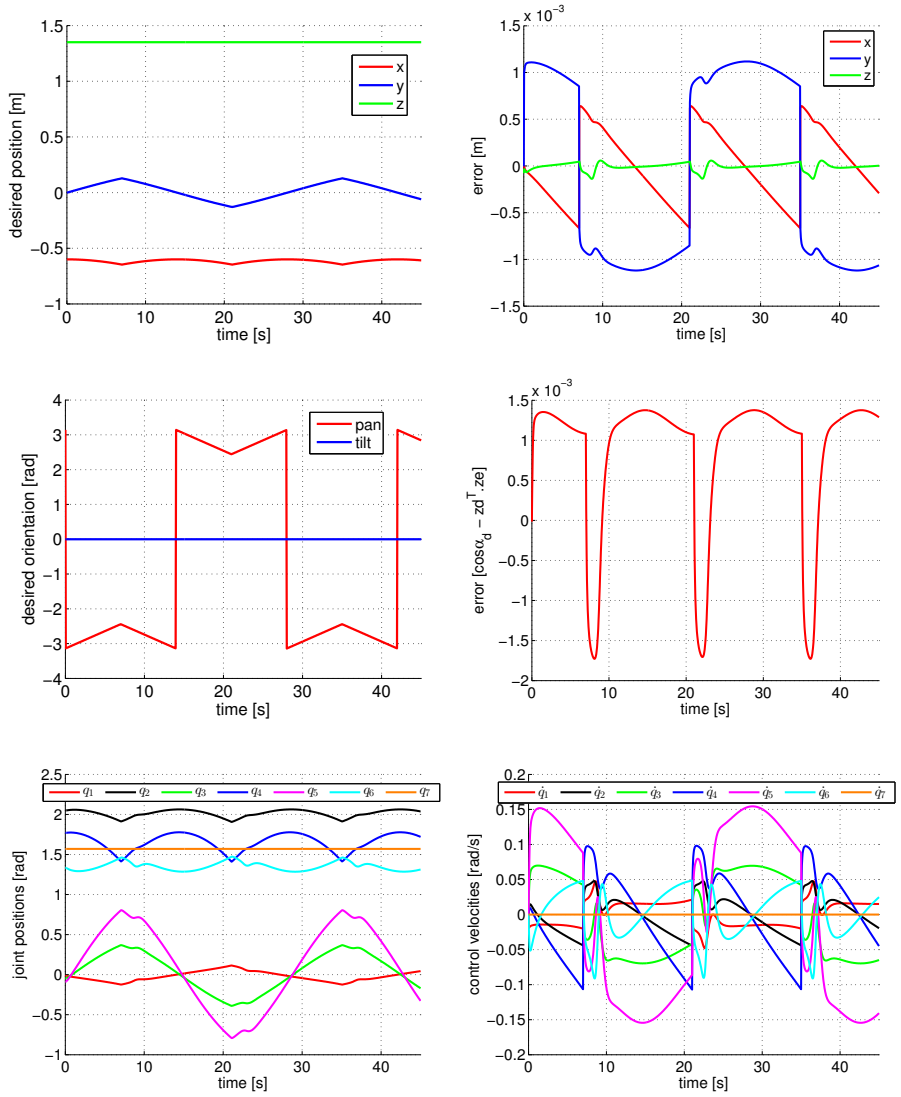


Figure 4.3: Motion control with the TA method in the second simulated task. Quantities are the same reported in Fig. 4.2.

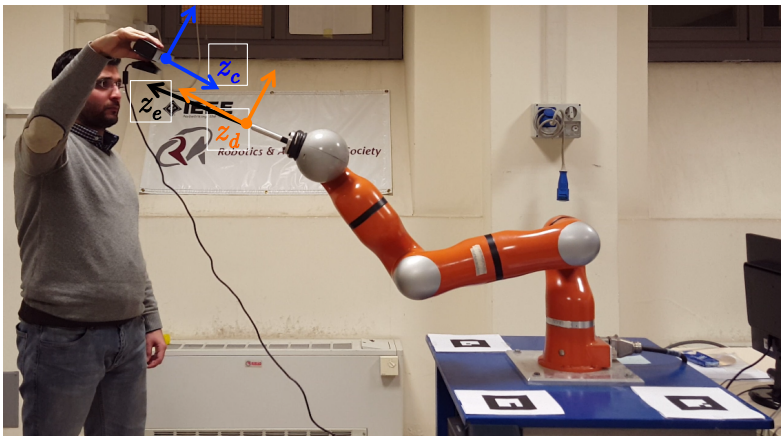


Figure 4.4: Snapshot of the experiment. The camera is simply held by hand and moved around, but frames and offsets mimic the situation of a camera mounted on the human head. Three markers are used for continuous camera localization. Depends on the camera pose, the marker with the highest confidence factor being used.

4.4.2 Experimental Evaluation with Task Augmentation

Experiments were conducted with a KUKA LWR IV manipulator in a ROS environment, using the joint position control mode of the FRI with a sampling time $t_s = 5$ ms. An Intel core i5 @2.5GHz×4 laptop was used under 64-bit Ubuntu. For the online localization of the human, in this experiment we used the ARToolKit library with a RGB camera (i.e. Kinect camera [1]) as shown in Sec.1.4. Three markers were placed on the robot supporting table and an additional EKF was used. Nonetheless, we found in practice that performance in the visual coordination tasks was already good without the further addition of the IMU. The camera is being held by an operator who is moving around in the workspace tracing an arbitrary, a priori unknown path, see Fig. 4.4. The offset data h and d, and the relative angle α_d were chosen as in (4.14). Task augmentation was used to control the robot motion, with (cautious) gains $\boldsymbol{K}_P = 1.5 \cdot \boldsymbol{I}$ and $k_{rp} = 1.5$, and no feedforward.

A video for one representative experiment can be watched here, where the obtained results are reported in Fig. 4.5. The error peaks on position and pointing angle are related to fast transients in the camera motion, leading to larger actual differences with respect to the desired visual coordination task. Nonetheless, once the initial mismatching error is recovered, the maximum of the (non-dimensional and normalized) pointing error was $e_{rp,max} \simeq 0.04$. Similarly, $\|\boldsymbol{e}_{p_{ee}}\|_{max} \simeq 0.14$ m.

[1] We used the libfreenect-based ROS driver for the Microsoft Kinect, where the camera parameters can be found and edited.

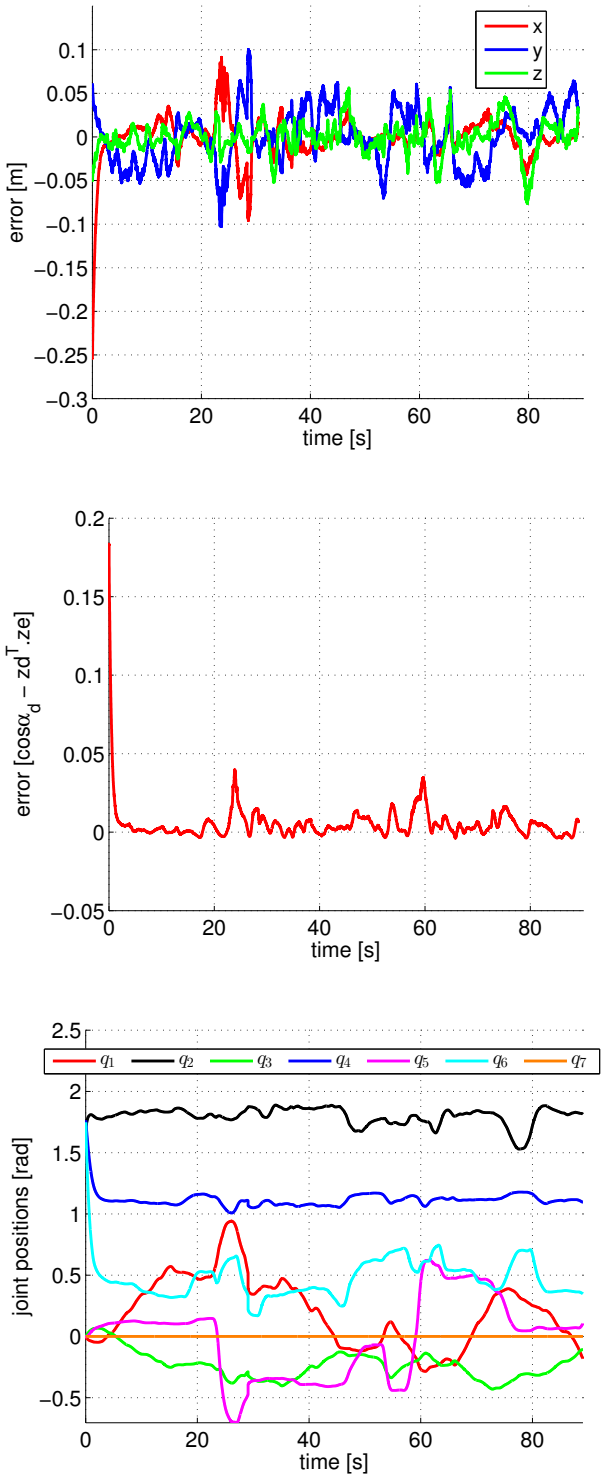


Figure 4.5: Experimental results with the KUKA LWR IV: (top) position error; (center) relaxed pointing error; (bottom) evolution of the joints.

4.5 Saturation in the Null Space Algorithm

In this section we will consider the collision avoidance task in addition to the EE coordination tasks in Chap. 2. For this, the saturation in the null-space algorithm (SNS) of Flacco et al. [2015a] will be used to handle all desired tasks in a specific priority. The EE collision avoidance task will be considered as a desired Cartesian velocity, while the collision avoidance tasks of the other control points along the robot body will be transformed to hard joint space limits. In this case, the SNS algorithm will be applied at the acceleration level to avoid any joint velocity discontinuity due to the switching of saturated joints Flacco et al. [2015a]. The distance computations required for collision avoidance are done using the depth space approach in Sec. 3.2.

Repulsive vector of the robot EE

For EE collision avoidance, the repulsive vector in (3.7) is considered as a repulsive velocity $\dot{\boldsymbol{p}}_{c_{ee}}$ that directly modifies the EE original desired velocity $\dot{\boldsymbol{p}}_{ee_d} \in \mathbb{R}^3$ for the coordinated positional task as

$$\dot{\boldsymbol{p}}'_{ee_d} = \dot{\boldsymbol{p}}_{ee_d} + \dot{\boldsymbol{p}}_{c_{ee}}, \tag{4.15}$$

where

$$\dot{\boldsymbol{p}}_{ee_d} = v \, \frac{\boldsymbol{p}_{ee_d} - \boldsymbol{p}_{ee}}{\|\boldsymbol{p}_{ee_d} - \boldsymbol{p}_{ee}\|}, \tag{4.16}$$

and

$$\dot{\boldsymbol{p}}_{c_{ee}} = \boldsymbol{u}(c_{ee}) = u(\boldsymbol{c}_{ee}) \, \boldsymbol{u}_{mean}(\boldsymbol{c}_{ee}). \tag{4.17}$$

The EE repulsive vector direction $\boldsymbol{u}_{mean}(\boldsymbol{c}_{ee})$ is computed as in (3.5), to prevent the EE to stuck between any obstacles located in symmetric positions. The desired EE position $\boldsymbol{p}_{ee_d}$ in (4.16) is computed according to the desired collaboration using (2.2) or (2.3).

To have a smooth trajectory, the velocity magnitude v in (4.16) is evaluated at discrete instants $t_k = kT$, being $T > 0$ the sampling time, as

$$v_k = v(t_k) = k_p \|\boldsymbol{p}_{ee_d,k} - \boldsymbol{p}_{ee,k}\| - k_d v_{k-1}, \tag{4.18}$$

where $\boldsymbol{p}_{ee} = \boldsymbol{k}(\boldsymbol{q})$ is the robot direct kinematics, $k_p > 0$, $k_d > 0$, and v_{k-1} is the previous sample of the Cartesian speed. From (4.15), the desired task acceleration $\ddot{\boldsymbol{p}}'_{ee_d}$ is obtained as

$$\ddot{\boldsymbol{p}}'_{ee_d} = \frac{\dot{\boldsymbol{p}}'_{ee_d} - \dot{\boldsymbol{p}}_{ee}}{T}, \tag{4.19}$$

where $\dot{\boldsymbol{p}}_{ee} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$ is the EE velocity and $\boldsymbol{J}$ is the $3 \times n$ analytic Jacobian related to the positional task.

Repulsive vectors along the robot body

Following Flacco et al. [2015a], the repulsive vectors associated to control points along the robot body are treated as Cartesian constraints with artificial forces that are translated into joint velocity and acceleration inequality constraints. Let $D_{min}(\boldsymbol{c})$ and $\boldsymbol{u}_{min}(\boldsymbol{c})$ be the minimum distance and the minimum vector

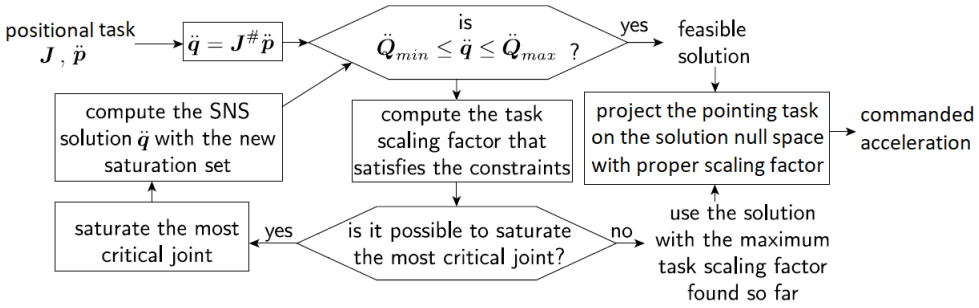


Figure 4.6: The block diagram of the SNS algorithm.

between a control point $\boldsymbol{c}$ and all objects in its associated surveillance region. In this case, a 'risk of collision' can be defined through the function similar to (3.3) as

$$u(\boldsymbol{c}) = \frac{1}{1 + e^{(D_{min}(\boldsymbol{c})(2/\rho)-1)\gamma}}. \tag{4.20}$$

Accordingly, a Cartesian constraint force can be defined as $u(\boldsymbol{c})\boldsymbol{u}_{min}(\boldsymbol{c})$ and converted to the joint space by

$$\boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{J}_c^T(\boldsymbol{q})\big[u(\boldsymbol{c})\,\boldsymbol{u}_{min}(\boldsymbol{c})\big], \tag{4.21}$$

where $\boldsymbol{J}_c$ is the analytic Jacobian of the direct kinematics for the position of the control point $\boldsymbol{c}$. Each component of the n-dimensional vector $\boldsymbol{h}$ represents the 'degree of influence' of the Cartesian constraint on the homologous joint. Next, the admissible velocity limits of each joint will be reshaped using the risk of collision function (4.20) according to the rule

$$\begin{aligned} \text{if } h_i > 0, \quad & \dot{q}_{max,i} = \dot{Q}_{max,i}\,(1 - u(\boldsymbol{c})) \\ \text{else}, \quad & \dot{q}_{min,i} = -\dot{Q}_{max,i}\,(1 - u(\boldsymbol{c})), \end{aligned} \tag{4.22}$$

where $\pm\dot{Q}_{max,i}$ are the (symmetric) original bounds on the ith joint velocity, i.e., $|\dot{q}_i| \leq \dot{Q}_{max,i}$, for $i = 1, \ldots, n$. As a result, the modified velocity limits will be converted as bounds on the actual acceleration commands, namely

$$\ddot{Q}_{min,i} = \frac{\dot{q}_{min,i} + \dot{q}_i}{T} \leq \ddot{q}_i \leq \frac{\dot{q}_{max,i} + \dot{q}_i}{T} = \ddot{Q}_{max,i}, \tag{4.23}$$

for $i = 1, \ldots, n$. Multiple Cartesian constraints, arising from different obstacles, can be taken into account by considering, for each joint i, the maximum degree of influence of all these virtual constraints.

SNS algorithm for the first priority task

At this stage, we can apply the simplest version of the SNS algorithm at the acceleration level Flacco et al. [2012a], considering the joint acceleration limits (4.23), which already embed collision avoidance for the robot body, and exploiting robot redundancy to realize the other desired tasks as much as possible (see Fig. 4.6). In our framework, the first priority task for the robot is to follow with its end-effector EE a specific position trajectory $\boldsymbol{p}_{ee_d}(t)$ that is coordinated

with the motion of the head of the human operator, as defined in Sec. 2.2. This will be the case, unless the acceleration command $\ddot{\boldsymbol{p}}'_{ee_d}$ in (4.19) includes the velocity modification resulting from the EE collision avoidance scheme (4.17) and (4.15).

In the SNS algorithm, the joint acceleration satisfying the first task is computed through some iterations (at the current sampling instant) based on Jacobian pseudoinversion as

$$\ddot{\boldsymbol{q}}_1 = \ddot{\boldsymbol{q}}_{N,1} + (\boldsymbol{J}_1\boldsymbol{W}_1)^{\#}(s_1\ddot{\boldsymbol{p}}'_{ee_d} - \dot{\boldsymbol{J}}_1\dot{\boldsymbol{q}} - \boldsymbol{J}_1\ddot{\boldsymbol{q}}_{N,1}), \tag{4.24}$$

with the first iteration being initialized with $\boldsymbol{W}_1 = \boldsymbol{I}$, $\ddot{\boldsymbol{q}}_{N,1} = 0$, and $s_1 = 1$. If the joint acceleration in (4.24) exceeds any of the limits (4.23) related to the robot body collision avoidance, it will be modified by bringing back to its saturated value the most violating command and projecting it into the null space of the task Jacobian of the enabled (non-saturated) joints (i.e., by suitably modifying the values of $\boldsymbol{W}_1$, and $\ddot{\boldsymbol{q}}_{N,1}$). This is repeated until all acceleration limits are satisfied or when the rank of $J_1W_1 < m$. In the latter case, a proper scaling factor $s_1 \in (0, 1)$ is necessarily used to reduce the original $\ddot{\boldsymbol{p}}'_{ee_d}$ and obtain feasibility. In practice, joint motions that are in contrast with the Cartesian constraints will be scaled down. When a constraint is too close, all joint motions that are not compatible with it will be denied.

SNS algorithm for the second priority task

The relaxed pointing task defined in section 2.3 can be realized by minimizing the cost function $H(\boldsymbol{q})$ in (4.13). This will be considered as our second (lower) priority task, thus preserving the higher priority positional task and still without violating the constraints on the joint acceleration commands associated to the robot body collision avoidance requirement. Therefore, the negative gradient of the cost function (4.13) with a step $k_g > 0$ is projected in the auxiliary null-space projector $\boldsymbol{P}$ given by

$$\boldsymbol{P} = (\boldsymbol{I} - ((\boldsymbol{I} - \boldsymbol{W}_2)(\boldsymbol{I} - \boldsymbol{J}_1^{\#}\boldsymbol{J}_1))^{\#})(\boldsymbol{I} - \boldsymbol{J}_1^{\#}\boldsymbol{J}_1), \tag{4.25}$$

where initially $\boldsymbol{W}_2 = \boldsymbol{I}$. For each saturated $\ddot{\boldsymbol{q}}_{1i}$, we shall set $\boldsymbol{W}_{2ii} = 0$. Iterations proceed then as for the first task. The final commanded joint acceleration will take the form

$$\ddot{\boldsymbol{q}}_{com} = \ddot{\boldsymbol{q}}_1 + s_2\boldsymbol{P}(-\boldsymbol{D}\dot{\boldsymbol{q}} - k_g\nabla H(\boldsymbol{q})), \tag{4.26}$$

where $s_2 \in (0, 1)$ is a proper scale factor introduced only if feasibility with respect to the hard inequality constraints cannot be recovered. The addition of a (diagonal) damping matrix $\boldsymbol{D} > 0$ in the null space is strictly recommended when working with acceleration commands, in order to eliminate any uncontrolled self-motion velocity. The complete multi-task SNS algorithm at the acceleration level can be found in Flacco et al. [2012a].

4.5.1 Experimental Evaluation

In the experimental setup we have considered again the KUKA LWR4 manipulator. The robot should execute the coordination positional and pointing tasks while avoiding collision with the human operator or any static or dynamic

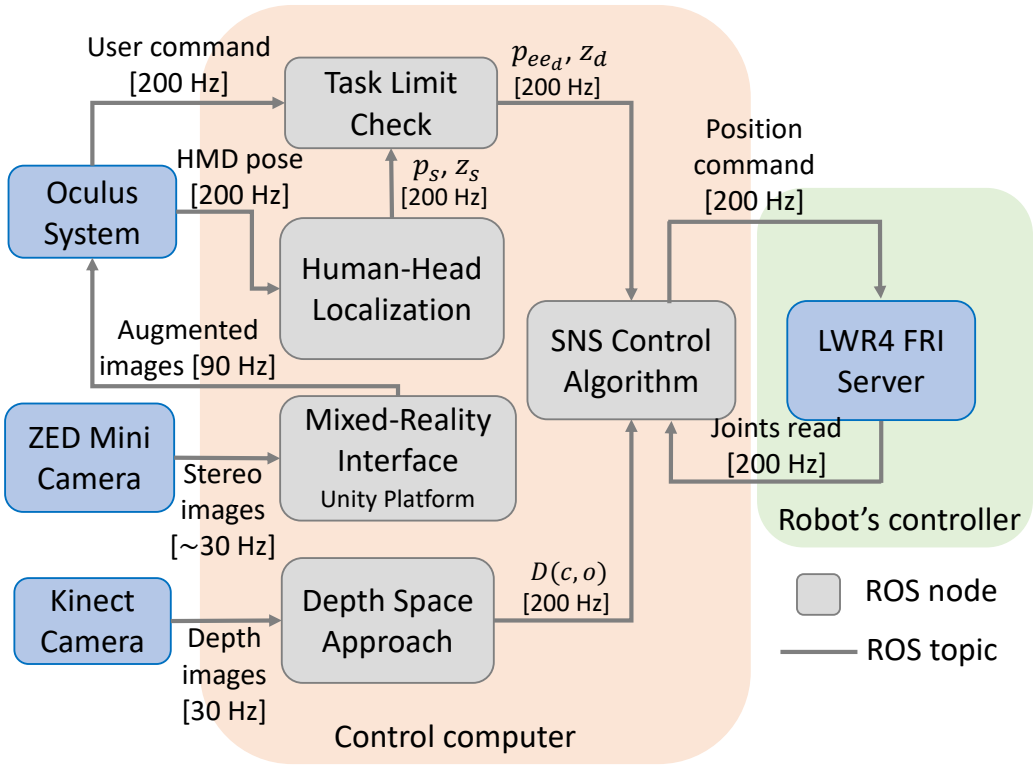


Figure 4.7: Data flow diagram for the proposed control system.

obstacle in the environment. The proposed control algorithm in Fig. 4.6 is implemented using C++ through ROS 2 middle-ware. The control framework is implemented according to the data flow diagram in Fig. 4.7. The KUKA robot is controlled using the position control mode through the Fast Research Interface (FRI) library KUK [2011], with a control cycle of $T = 5$ [ms]. For collision avoidance, the workspace is monitored by a Microsoft Kinect depth sensor that captures 640×480 depth images at a frequency of 30 Hz. For human head localization, a combination of Oculus Rift and single tracking sensor is used as illustrated in Sec. 1.5. The Rift is attached to the operator helmet while the tracking sensor is located in a static place, directed toward the robot workspace, see Fig. 4.12. The control scheme in these experiments can be represented using the general proposed one in Fig. 2, where in this case there is no mixed-reality interface and the involved priority control algorithm is the SNS. The system runs on core i7-6700k CPU @4GHz, with 4-core and 8-logical processors.

The parameters values for EE collision avoidance in (3.3) and for the other control points in (4.20) have been set to $\rho = 0.3$ [m], $u_{max} = 1.5$ [m/s] and $\gamma = 5$. The used motion control gains are $k_p = 0.5$, $k_d = 0.05$ in (4.18), and $k_g = 10$ in (4.26). Finally, for the coordinated motion tasks we have used the parameters $h = d = 0.15$ [m] for the following task in (2.2) and $\alpha_d = 5°$ for the pointing task in (4.13). The results of two typical experiments are presented in the following paragraphs. More experiments are shown in a video available here.

First experiment

In the first experiment the human operator is standing still in front of the robot during the whole experiment (for about 60 s), keeping his head at rest and frequently moving his right hand toward the robot (see Fig. 4.8). At the beginning, the robot EE moves to comply with the desired coordination task, minimizing the position and orientation errors w.r.t. the constant reference specified by the human. Every time an obstacle (in this case, the operator hand) is coming closer to the robot, the EE will try to achieve the task while primarily avoiding collision. If this is impossible, the robot will move away from the obstacle increasing thus the coordination error. When the operator moves back the hand away from the robot end-effector, the EE starts again being able to achieve the coordinated task. In Fig. 4.8, since the head position is always out of the robot predefined task limits, the corresponding projection is used to define the desired task. The difference between the actual EE position and the desired one is due instead to obstacle avoidance.

The positional tracking error in Fig. 4.9(a) increases every time the robot is not able to achieve the task because of the need of avoiding obstacles. The robot reaction magnitude (3.3) in Fig. 4.9(c) indicates in fact how close is the nearest obstacle to the robot EE. Nonetheless, the EE keeps approximately the desired orientation during the whole experiment, since this is not in conflict with the higher priority task (Fig. 4.9(b)). The scaling factor $(1 - u(\boldsymbol{c}))$ on the velocity limits (4.22) is shown in Fig. 4.9(d). Only the limits corresponding to joints that are more influenced by the presence of the obstacle (joints 1 to 3) are reduced.

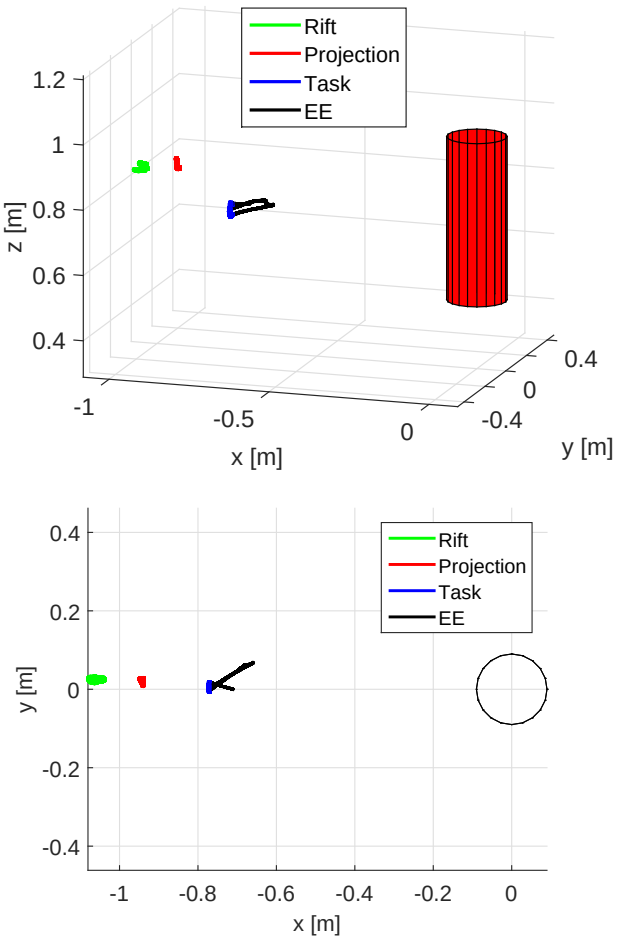
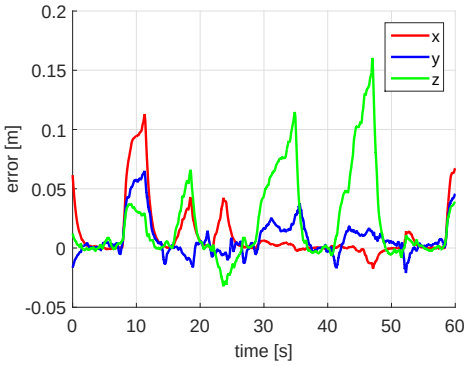
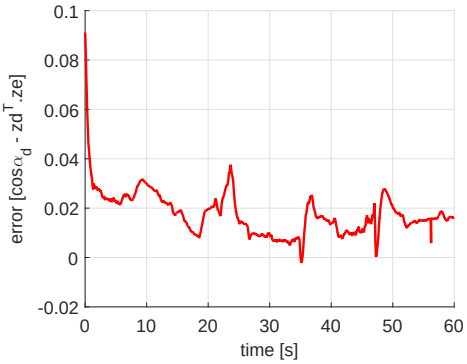


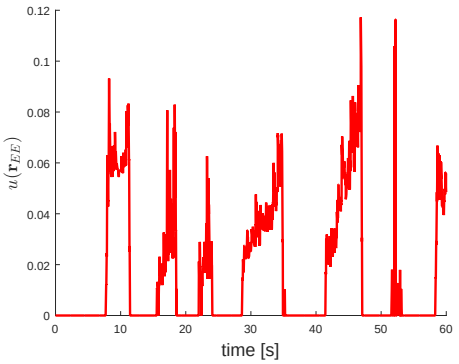
Figure 4.8: 3D (above) and top (below) views of the coordination task during the first experiment. Green points: Estimated position of the human head. Red points: Head pose projection on the task limit sphere. Blue points: Desired positional task. Black points: EE position. The cylinder/circle denote the robot base location.



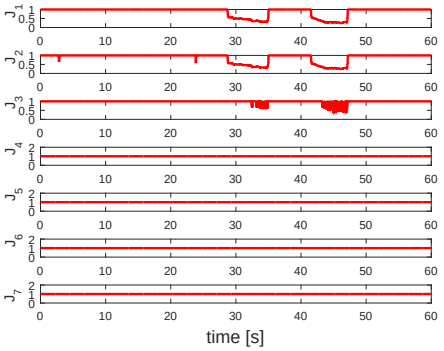
(a)



(b)



(c)



(d)

Figure 4.9: First experiment: The operator is standing still in front of the robot and uses his hand as a dynamic obstacle. (a-b) EE position and orientation errors. (c) EE desired reaction magnitude. (d) Joint limit scaling factors.

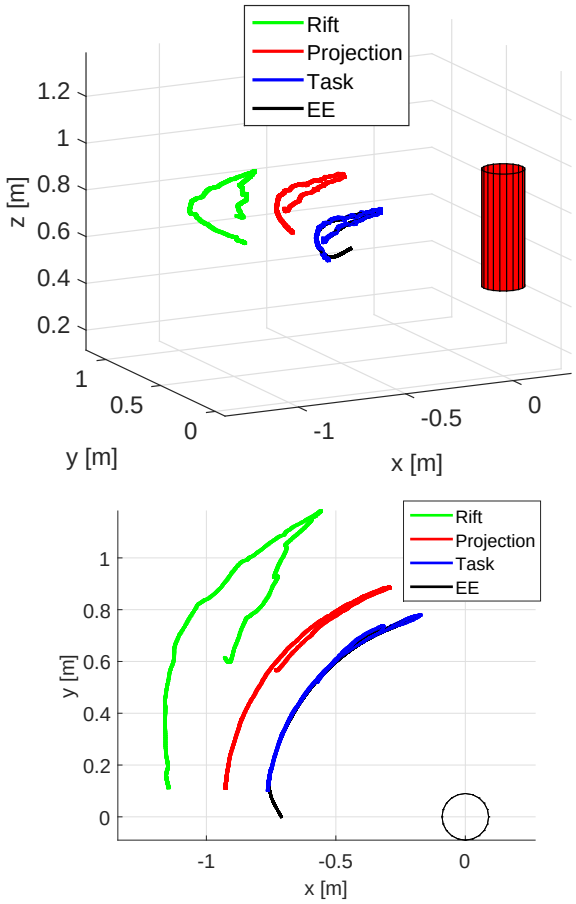
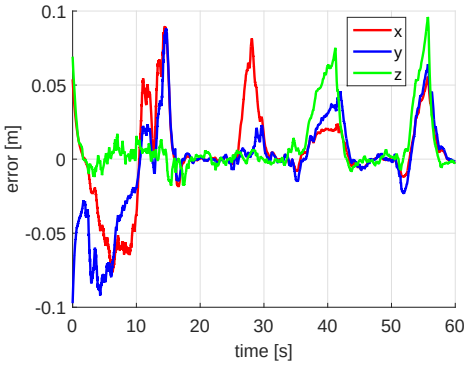
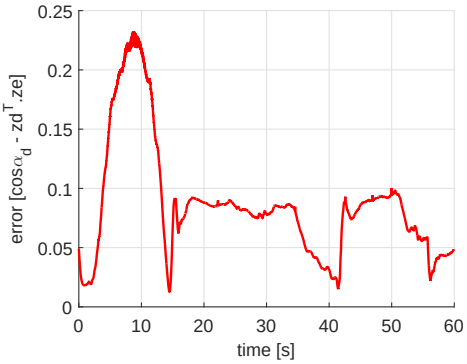


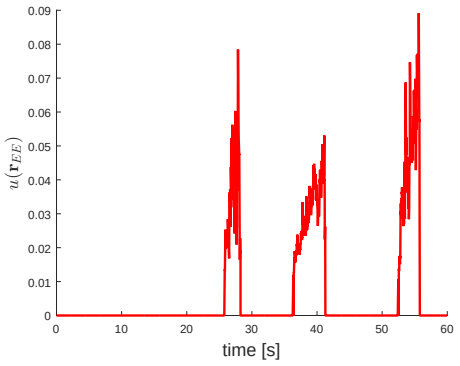
Figure 4.10: 3D (above) and top (below) views of the coordination task during the second experiment. Legend is the same as in in Fig. 4.8.



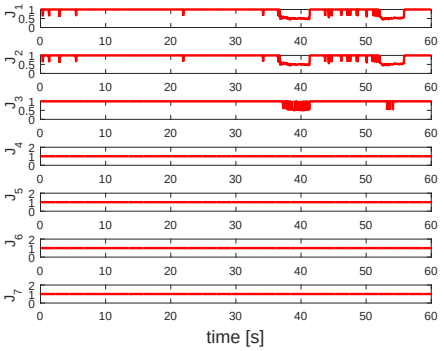
(a)



(b)



(c)



(d)

Figure 4.11: Second experiment: The operator is moving and uses his hand as a dynamic obstacle. Quantities are the same reported in Fig. 4.9.

Second experiment

In the second experiment, the human operator moves initially and the robot follows his head for the first 15 s approximately. Then, the human stops and moves his hand toward the robot as in the first experiment. Figure 4.10 shows how the human head pose estimation using the Oculus Rift is continuous and smooth without any outliers. During some intervals, the EE does not follows the desired behavior, again due to the need of obstacle avoidance.

In Fig. 4.11(a-b), the large initial values of the position tracking error are due to the relatively fast motion of the human operator in this case. A similar behavior is illustrared also in the mentioned video clip. The successive increases in the position error are due instead to obstacle avoidance, as indicated also by the three peaks in Fig. 4.11(c). At about $t = 43$ s, the robot is keeping the desired EE position while an obstacle is getting nearer the first and second joints. However, since the desired orientation has lower priority, its corresponding error will be relatively high in response to the obstacle avoidance.

It must be remarked that multiple obstacles may affect at the same time the motion of the robot, while acting on different parts of the structure. One example is shown in Fig. 4.12, which is a snapshot from the video of another experiment. In this case, the robot end-effector is pushed away from its coordinated motion (with the human head) and is moving downward in response to the hand motion, which is seen as a dynamic obstacle. However, in this way the robot elbow is dangerously approaching the table supporting the robot base, a static obstacle in the workspace, and the robotic system may also get stuck in the limit. The proposed control scheme handles these situations as well, with the SNS method smoothly stopping the joint motion.

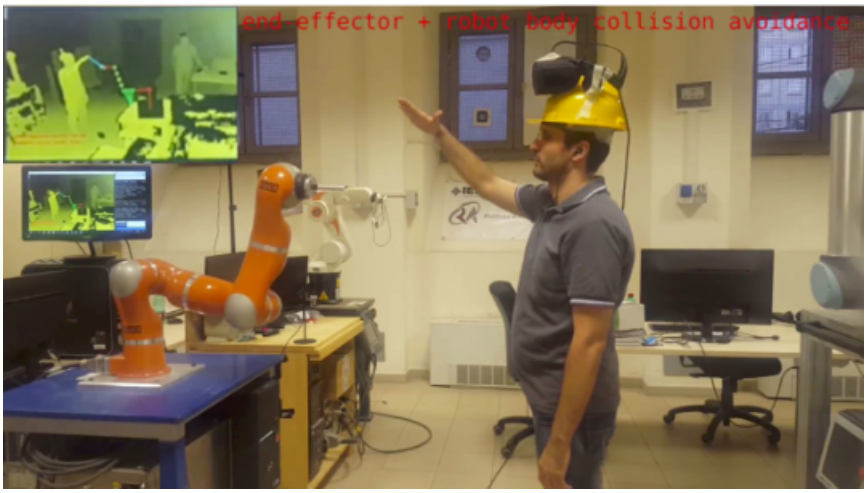
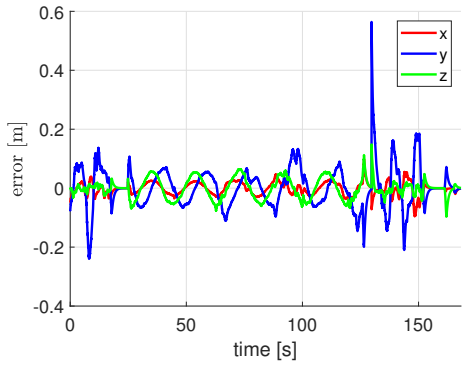
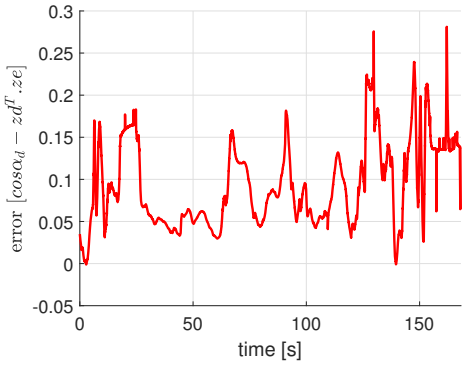


Figure 4.12: A situation in which the simultaneous avoidance of two obstacles affects the original human-robot coordinated motion: the dynamic obstacle constituted by the human hand moving toward the robot end-effector and the supporting table as a static obstacle acting repulsively from the bottom on the robot elbow. Minimum distances to control points are highlighted in the depth image on the top left.



(a)



(b)

Figure 4.13: Experiment with mixed-reality interface: (a) The EE position and (b) orientation errors. The operator is switching between the available tasks and moving his hand as a dynamic obstacle.

4.5.2 Including Mixed-Reality Interface

The last experiment using the SNS algorithm is done including the proposed mixed-reality interface in Sec. 1.6, where the whole proposed general scheme in Fig. 2 is considered. The robot should execute the desired coordination task selected by the operator using the Oculus controller. The robot starts with the human-head following task as a default task, and then the operator can change it to "circle" or "stop" tasks as defined in the Sec. 2.2.2. Also, the desired α_d for the pointing task can be chosen (i.e. 5° or 90°). In this case the ZED-Mini camera is attached to the Oculus Rift as shown in Fig. 1.9. The parameters values for motion control and collision avoidance are the same as previous, but the h parameter for the coordination task (2.2) is set to zero. The results of the experiment is presented in the following paragraph, and its corresponding video is available at this link.

In Fig. 4.13, the error in the desired coordination tasks is shown for the whole experiment. From beginning, the "follow" task is activated for 16 s where at the 15th second the human hand act as a moving obstacle near the EE. Then, the operator activates the "stop" task from the second 17 until the second 23.

Between the seconds 24 and 59, the "circle" is activated and the user keeps standing. Next, the user moves till the second 125. After that the "stop" task is selected for two seconds only. From the second 129 to 150, the "follow" task is re-activated. After that, the user selects the "stop" mode. At the seconds 141 and 160 the human hand act again as a moving obstacle near the robot. The high error peaks in the EE desired position happened whenever the human hand moves nearly (i.e. at the seconds 16 and 141).

4.6 Task Priority Matrix

The solution using the task augmentation in (4.2) has a good inverse kinematic solution shape, but it does not consider the tasks priority and singularities may appear when there are linear dependencies between the tasks. The null-space projection approach in (4.9) takes into account the task priority strictly. On the other hand, some information for the contribution of lower priority tasks can not be inferred. This missing information is important to perform a smooth transition during any change in the tasks priority. In the SNS algorithm, the modified joint limits are considered hard although they are determined approximately in (4.21) and (4.22). Also, it is more complicated than the previous two solutions.

In order to have a simple solution that combines the advantages of both (4.2) and (4.9) solutions, a new approach that has a similar shape of (4.2) and returns exactly the solution obtained by (4.9), keeping the whole knowledge of the tasks, has been presented by Flacco [2016], where redundancy resolution is independent from enforcing desired task priority. This is done using the so-called Tasks Priority Matrix (TPM) as follows

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}\dot{\boldsymbol{p}}, \tag{4.27}$$

where $\boldsymbol{F} \in \mathbb{R}^{m\times m}$ is a square matrix to be computed in a specific way to control the desired task priority strictly. The key tool of TPM calculation is to use the Gauss-Jordan elimination method, which is commonly used to compute the reduced row echelon form (rref), considering a pivot square matrix with dimension $(m_i \times m_i)$ for each corresponding desired task instead of pivot elements as in rref.

Let the QR decomposition for the transpose of the augmented Jacobian (4.3) given by

$$\boldsymbol{J}^T = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{O}\end{bmatrix}, \tag{4.28}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is an upper triangular matrix. The diagonal block matrices $\boldsymbol{R}_{ii} \in \mathbb{R}^{m_i\times m_i}$ correspond to the desired tasks will be used as pivots matrices during the TPM calculations. Note that, the pivot matrix $\boldsymbol{R}_{ii}$ is nonsingular if the task i is nonsingular and linearly independent from all tasks with higher priority.

The algorithm to compute the TPM is as follows

Step 1: Initialize the TPM as

$$\bar{\boldsymbol{F}} = \boldsymbol{R}. \tag{4.29}$$

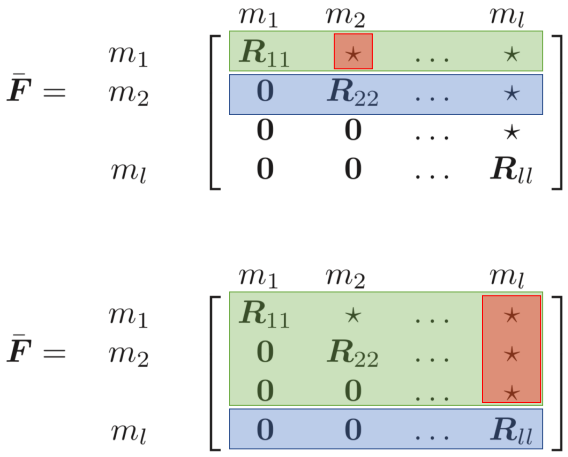


Figure 4.14: The structure of the temporary matrix $\bar{\boldsymbol{F}}$. The highlighted components refer to the involved block matrices during the operations of computing the TPM for $i = 2$ [top], and $i = l$ [bottom].

Step 2: Let $m_{h_i} = \sum_{k=1}^{i-1} m_k$ and $m_{h_1} = 0$. Multiply the block row $\bar{\boldsymbol{F}}(m_{h_i} + 1 : m_{h_i} + m_i, 1 : m)$, i.e. the blue blocks in Fig. 4.14, with the pseudoinverse of its pivot matrix, i.e. $\boldsymbol{R}_{ii}^{\#}$. As a result, the i-th diagonal block will be Identity if $\boldsymbol{R}_{ii}$ is nonsingular.

Step 3: For $i > 1$, multiply the block $\bar{\boldsymbol{F}}(1 : m_{h_i}, m_{h_i} + 1 : m_{h_i} + m_i)$, i.e. the red blocks in Fig. 4.14, by the row block $\bar{\boldsymbol{F}}(m_{h_i} + 1 : m_{h_i} + m_i, 1 : m)$, i.e. the blue blocks in Fig. 4.14, and subtract it from the block row $\bar{\boldsymbol{F}}(1 : m_{h_i}, 1 : m)$, i.e. the green blocks in Fig. 4.14,.

Step 4: Repeat the steps 2 and 3 for all blocks row related to all tasks, i.e. for $0 < i \leq l$.

Step 5: Get the transpose of the modified matrix where

$$\boldsymbol{F} = \bar{\boldsymbol{F}}^{T}_{\text{modified}}. \tag{4.30}$$

Note that, the QR decomposition in (4.28) can be also used to compute the $\boldsymbol{J}^{\#}$ in (4.27) as

$$\boldsymbol{J}^{\#} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{O} \end{bmatrix}^{-T}. \tag{4.31}$$

In Flacco [2016], the solution using TPM (4.27) is proofed to be exactly equal to the standard null-space projection (4.9). Note that, using (4.27), the redundancy resolution is totally separated from controlling the SoT in a desired priority order. This gives more flexibility for applying any desired change in the task priority on-line. In this case, the only part to be recomputed in (4.27) is the $\boldsymbol{F}$ matrix according to the new tasks order. Also, to boost the computational

efficiency, the QR decomposition in (4.28) can be parallelized. While using (4.9), all computations should be repeated sequentially since the low priority tasks control has to wait for the outcome of all higher priority tasks. On the other hand, using TPM, it is not clarified in Flacco [2016] if and how the tasks defined in the joint space can be included (e.g., joint damping). Furthermore, both solutions in (4.9) and (4.27) suffer from joints velocity discontinuity within any change in either the desired SoT or in the tasks dependencies between each others. A scale factor can be used in (4.9) to achieve soft transition as proposed in Lee et al. [2012]. Also, since the solutions are defined in the velocity level, the advantage of including dynamics is missed.

In this thesis, we are proposing two main changes to the solution in (4.27) in order to get the following improvements

- Eliminating any undesired behaviour in the joints velocity.
- Allowing to include the consideration of dynamics and any desired joint space task.

This is done by defining a control law, similar to (4.27), at the acceleration level as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}(\ddot{\boldsymbol{p}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\ddot{\boldsymbol{q}}_0, \tag{4.32}$$

where $\ddot{\boldsymbol{q}}_0$ is used to achieve any desired joint space task. For example, in order to get stable and smooth joint trajectories, the joint acceleration $\ddot{\boldsymbol{q}}_0$ can be computed as

$$\ddot{\boldsymbol{q}}_0 = -\boldsymbol{D}\dot{\boldsymbol{q}}, \tag{4.33}$$

where $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is a positive semi-definite matrix, e.g., the robot mass matrix. Note that, by projecting $\ddot{\boldsymbol{q}}_0$ in the null space of the augmented Jacobian (4.3), the desired joint task/behaviour will be always in the lowest control priority without any influence on the higher priority Cartesian tasks.

4.6.1 The Complete Approach

In order to control the EE to achieve the prioritized desired Cartesian tasks while avoiding any collision efficiently, we propose:

- Using the soft pointing task in (2.7) or in (2.10) to control the desired orientation.
- Using the relaxed definition (3.11), consider one collision avoidance task for the most critical control point $\boldsymbol{c}_k$, i.e. $\mathbf{argmin}_k\, D_{min}(\boldsymbol{c}_k)$, when

$$D_{minimal} = \min_{k=1}^{y} D_{min}(\boldsymbol{c}_k) < \epsilon, \tag{4.34}$$

  where $D_{minimal}$ is the minimal distance between all obstacles and all control points, y is the number of the robot body control points, and $\epsilon > 0$ is a danger threshold distance.
- Similar to (4.15), adding the repulsive vector $\ddot{\boldsymbol{p}}_{c_{ee}}$ to the main EE positional task when $D_{min}(\boldsymbol{c}_{ee}) < \epsilon$, where $\ddot{\boldsymbol{p}}_{c_{ee}} = \boldsymbol{u}(c_{ee}) = u(\boldsymbol{c}_{ee})\,\boldsymbol{u}_{mean}(\boldsymbol{c}_{ee})$.
- Using algorithm 2 to set the proper priority for each desired task.
- Using the control law (4.32) with the null space damping (4.33).

Algorithm 2 Arranging tasks priorities for the TPM method

$\ddot{\boldsymbol{p}} = \ddot{\boldsymbol{p}}_{ee_d}, \quad \boldsymbol{J} = \boldsymbol{J}_{p_{ee}}, \quad \dot{\boldsymbol{J}} = \dot{\boldsymbol{J}}_{p_{ee}}$
if $D_{min}(\boldsymbol{c}_{ee}) < \epsilon$ **then**
  $\ddot{\boldsymbol{p}} = \ddot{\boldsymbol{p}}_{ee_d} + \ddot{\boldsymbol{p}}_{c_{ee}}, \quad \boldsymbol{J} = \boldsymbol{J}_{p_{ee}}, \quad \dot{\boldsymbol{J}} = \dot{\boldsymbol{J}}_{p_{ee}}$
end if
if $D_{minimal} < \epsilon$ **then**
  $\ddot{\boldsymbol{p}} = [\ddot{p}_{n_k}^T \quad \ddot{\boldsymbol{p}}^T]^T$
  $\boldsymbol{J} = [\boldsymbol{J}_{n_k}^T \quad \boldsymbol{J}_{p_{ee}}^T]^T$
  $\dot{\boldsymbol{J}} = [\dot{\boldsymbol{J}}_{n_k}^T \quad \dot{\boldsymbol{J}}_{p_{ee}}^T]^T$
end if
if $\alpha > \alpha_d$ **or** $\alpha < 0$ **then**
  $\ddot{\boldsymbol{p}} = [\ddot{\boldsymbol{p}}^T \quad \ddot{p}_{rp_d}^T]^T$
  $\boldsymbol{J} = [\boldsymbol{J}^T \quad \boldsymbol{J}_{rp}^T]^T$
  $\dot{\boldsymbol{J}} = [\dot{\boldsymbol{J}}^T \quad \dot{\boldsymbol{J}}_{rp}^T]^T$
end if

The special features of $\boldsymbol{F}$ matrix, make it simple to enforce any desired change in the SoT. Note that, the algorithm 2 can be extended/modified easily to apply other control strategies. For example, applying the collision avoidance for more than one interest point, or dividing the surveillance area into two different danger regions. In the farther one, the avoidance tasks are added in lower priorities preventing the robot in advance to near from the closer danger region.

4.6.2 Comparative Simulations

In this section we present different simulation scenarios and comparisons. First, we introduce a comparison between using TPM at the velocity level (4.27) and our proposed scheme at the acceleration level (4.32). Second, the robot behavior using the classical three dimensional avoidance task in (3.7) is compared to the relaxed one in (3.11). In the last, we show the difference between using the pointing task with equality constraint (2.7) and the inequality one in (2.10). The robot behavior in the different simulations are included in a video at this link.

The previous evaluations are done in MATLAB using the 7-dofs KUKA LWR IV robot. The desired tasks are to follow an ellipsoidal path in the Cartesian space for three rounds while pointing to a direction parallel to the world-frame positive x-axes. During this, the robot should avoid any collision with three static obstacles that are located in the robot work space (i.e. see Fig. 4.15). To have a challenging task, the first obstacle crosses the desired ellipsoidal path. The desired priority for each of the robot tasks will be arranged according to the algorithm 2 where the priority in a descending order is as following

1. Robot body collision avoidance.
2. EE collision avoidance.
3. Positional task.
4. Relaxed pointing task.

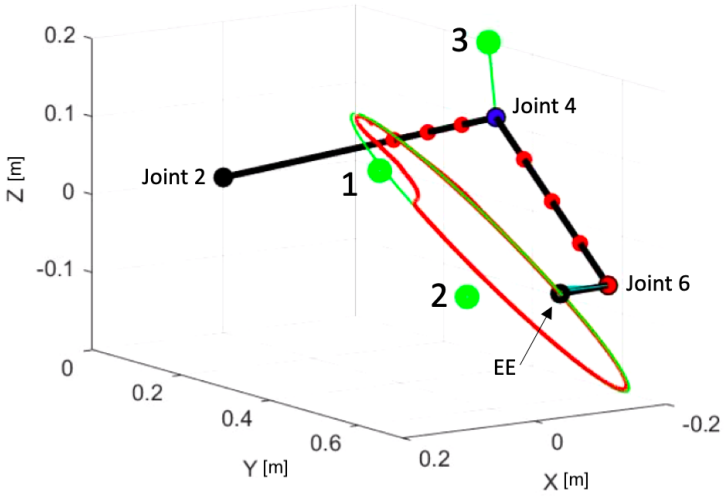


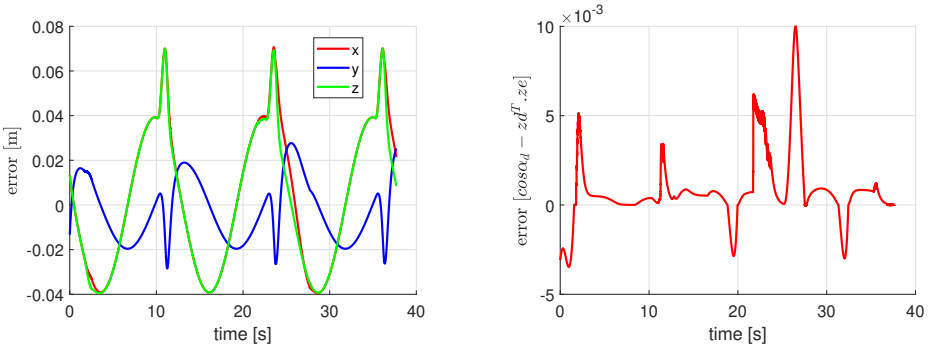
Figure 4.15: Simulation at the velocity level: a snapshot during tasks execution. Black lines represent the robot links, the black spheres denote to the positions of the robot joints and EE DH frames as in Fig. A.1. The red points represent 8 interest control points along the robot body. During the task, the active control point will be changed to the blue color and linked to the nearest obstacle by a green line. The green ellipsoid represents the desired path while the red curve shows the robot EE position. The three static obstacles are represented as green spheres.

When all obstacles are far enough from the robot, the collision avoidance tasks are eliminated, and the positional and the pointing tasks will be the first two priorities, respectively.

Velocity level vs. Acceleration level

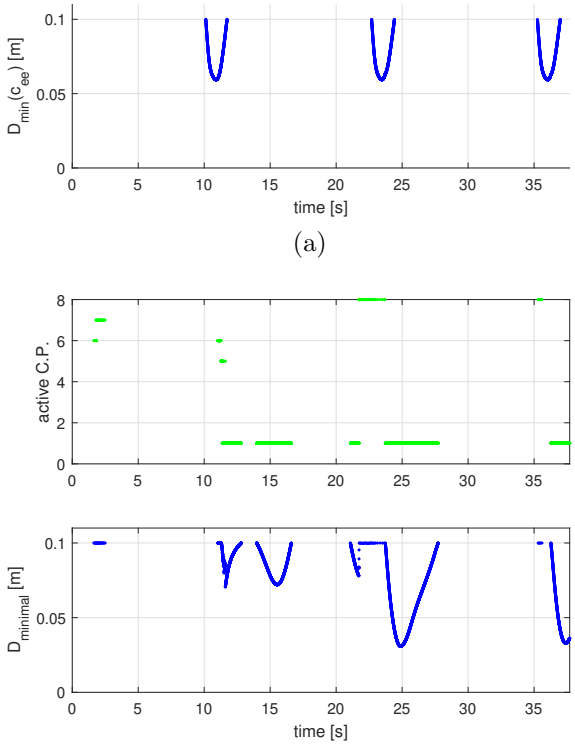
In the first comparison, the robot has a positional and an inequality pointing tasks with $\alpha_d = 5°$. For control points of collision avoidance, the one dimensional task definition in used. At the beginning, these tasks are controlled using TPM at the velocity level. Figure 4.16 shows the Cartesian tasks errors where the peaks appear when the robot should avoid a collision. In Fig. 4.17, it is shown how far the obstacles are from robot during the whole simulation. It is clear how the EE, at each round, moves far from the obstacle 1 when the distance between them is less than the admissible one, i.e. $\epsilon = 0.1$ [m]. Also, the robot avoids any collision with its body while preserving the desired Cartesian tasks as much as possible. Figure 4.18 shows the evolution of joints positions and velocities. One can see clearly, the discontinuity in the velocities corresponding to any change in the SoT. This happens mainly when the robot body collision avoidance task is added.

The previous tasks are repeated controlling the robot at this time using our proposed scheme at the acceleration level. In this case, the joint positions are smoother, the discontinuity in the joint velocities is eliminated with no high peaks in its values, see Fig. 4.21. Also, the error behavior in Fig. 4.19 is smoother without peaks when the robot should avoid an obstacle.



(a) (b)

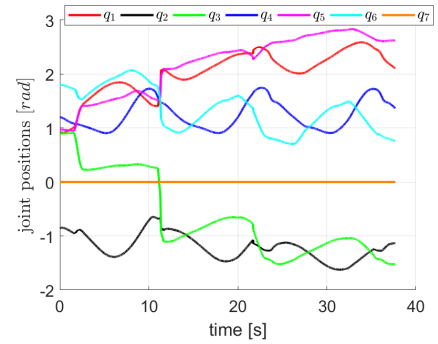
Figure 4.16: Simulation at the velocity level: (a) and (b) represent the errors of the EE positional and pointing tasks, receptively.



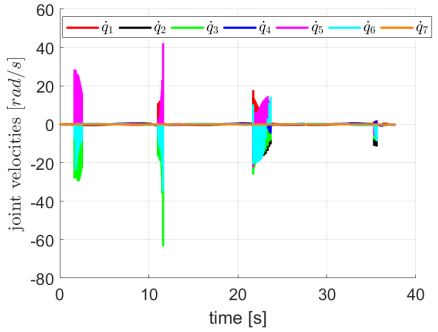
(a)

(b)

Figure 4.17: Simulation at the velocity level: (a) The distance from the EE and the nearest obstacle in the surveillance area, (b) The active control point during the simulation [above], and its corresponding $D_{minimal}$ distance (4.34) [bottom].

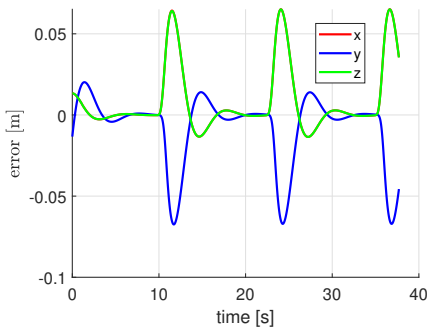


(a)

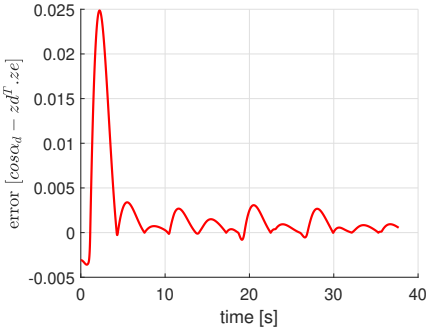


(b)

Figure 4.18: Simulation at the velocity level: the joint positions (a) and velocities (b).

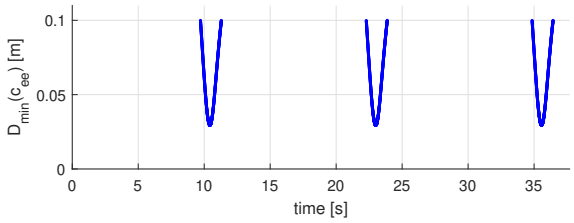


(a)

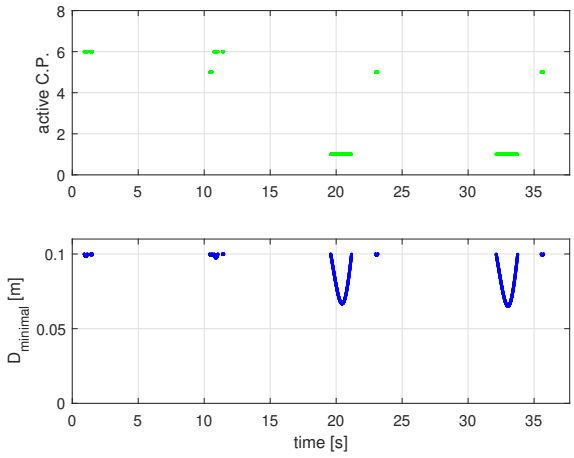


(b)

Figure 4.19: Simulation at the acceleration level: (a) and (b) represent the errors of the EE positional and pointing tasks, receptively.



(a)



(b)

Figure 4.20: Simulation at the acceleration level: (a) The distance from the EE and the nearest obstacle in the surveillance area, (b) The active control point during the simulation [above], and its corresponding $D_{minimal}$ distance [bottom].

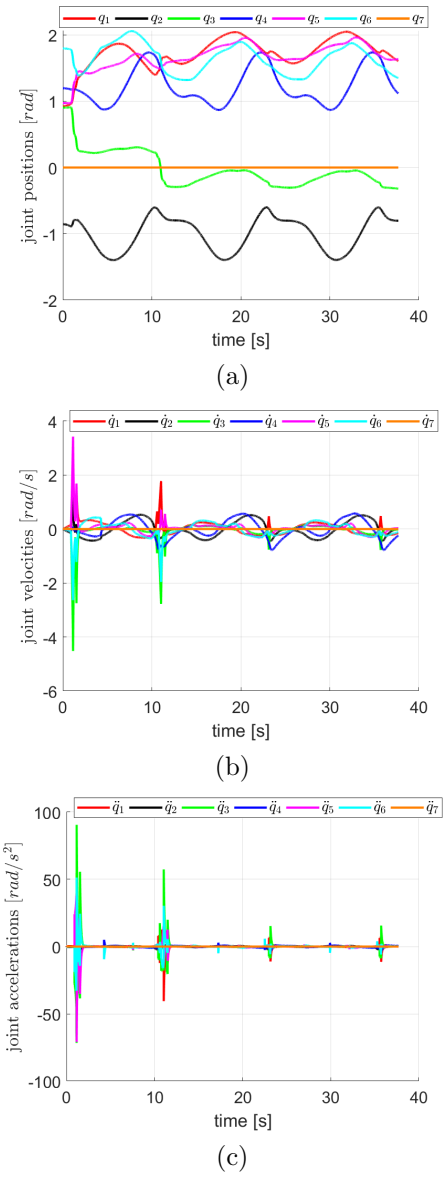


Figure 4.21: Simulation at the acceleration level: the joint positions (a), velocities (b) and accelerations (c)

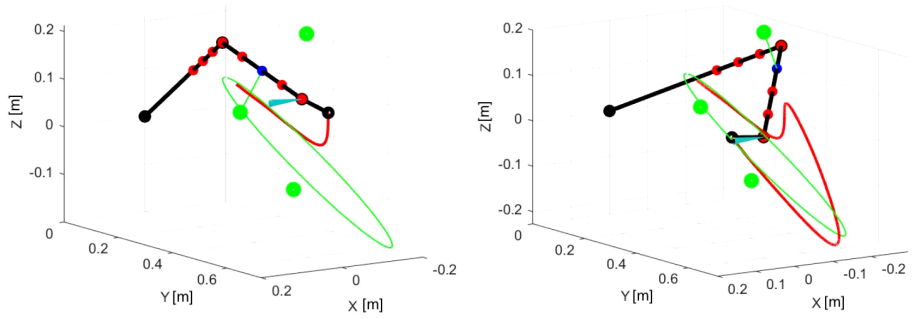


Figure 4.22: Simulation using three dimensional task for control points collision avoidance: two different snapshots during the simulation. The cyan cone represents the EE pointing task, for better visualization, its apex located at the sixth joint frame position.

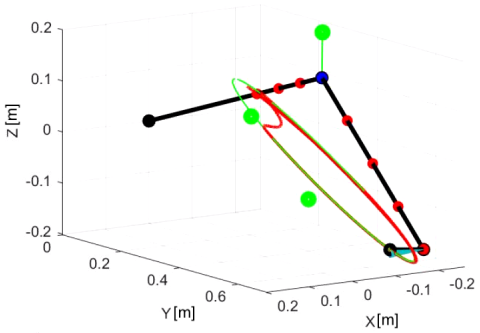


Figure 4.23: Simulation using one dimensional task for control points collision avoidance: a snapshot during tasks execution.

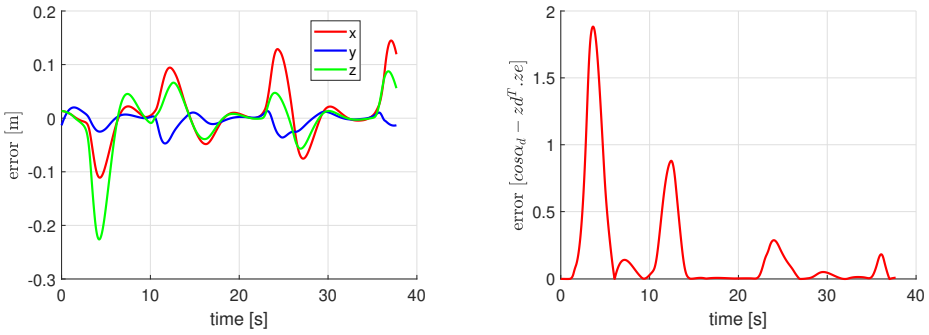
Different collision avoidance tasks for control points

We will compare between using the three and one dimensional avoidance tasks. For this, the robot should achieve the positional and an inequality pointing tasks with $\alpha_d = 5°$. The same previous simulation at the acceleration level is repeated, but here the classical avoidance task definition in (3.10) is used instead of the soft one in (3.11).

Figure 4.22 presents two robot snapshots during the simulation where we used the three dimensional task for robot body collision avoidance. In this case, at the beginning of the simulation, the robot can not handle the EE (positional and relaxed pointing) tasks, and the avoidance of robot body from the obstacle 1, simultaneously. Since the robot body collision avoidance at this case needs 3-dofs, there is no enough capability to achieve the EE tasks. On contrast, the EE can achieve the positional task when one dimensional task for robot body collision avoidance is used in Fig. 4.23,

In Fig. 4.25 (the 3-dimensional case), the robot is close to the obstacles more frequently and the distances between them are lower than in Fig, 4.20

(the relaxed case). At the same time, the errors in both position and pointing tasks in Fig. 4.24 are higher than in Fig. 4.19. This behavior is because the 3-dimensional avoidance task pushes the robot to move hardly far away from the current closest obstacle. This makes the robot be nearby other obstacles repeating the previous reaction again.



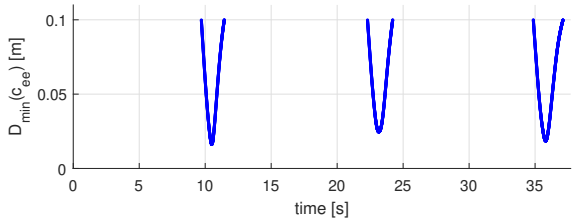
(a) (b)

Figure 4.24: Simulation using three dimensional task for control points collision avoidance: (a) and (b) represent the errors of the EE positional and pointing tasks, receptively.

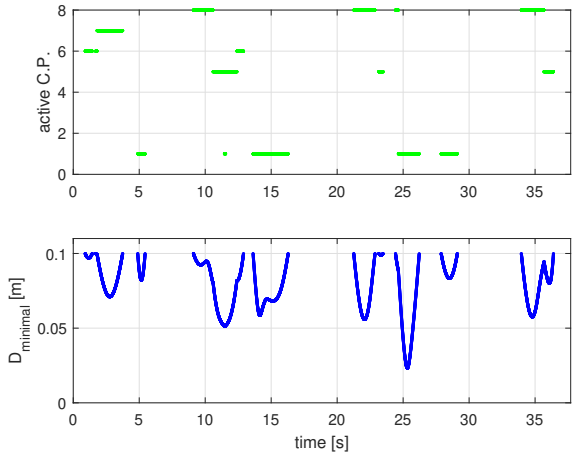
Pointing task constraint (equality vs. inequality)

For a desired relaxed pointing task $\alpha_d = 35°$. Two simulations are done using TPM at the acceleration level, 1-dimensional task for robot body collision avoidance and two different constraints for the pointing task. The different behavior in each case is very clear in Fig. 4.29, where in the equality case the EE link should be always on the surface of the cyan cone, while in the inequality case the EE link can be inside the cone or on its surface.

The error in the positional task in Fig. 4.30, during the first quarter of the simulation is larger using the equality constraint. In this period, the relaxed pointing task using the inequality constrained is deactivated and the robot has one more dof to handle efficiently the robot body collision avoidance at one of the interest points and the EE positional task. Using the inequality constraint, the pointing error has positive values when the EE points outside the cone in Fig. (4.31). In this case, the task is activated until the error value be equal or less than zero. While in the equality case, the task is always activated and the error should converge to zero.



(a)



(b)

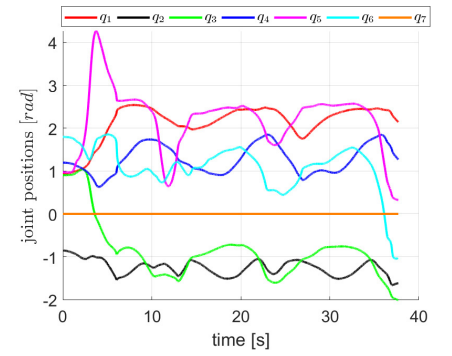
Figure 4.25: Simulation using three dimensional task for control points collision avoidance: (a) The distance from the EE and the nearest obstacle in the surveillance area, (b) The active control point during the simulation [above], and its corresponding $D_{minimal}$ distance [bottom].

4.6.3 Experimental Evaluation

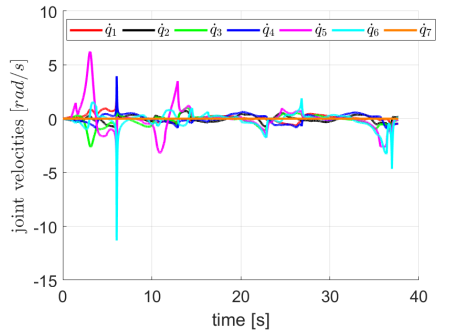
Our proposed algorithm is implemented using C++ in a ROS 2 platform and evaluated in an experimental setup using KUKA LWR robot. For this, we used the depth space approach in Sec. 3.2 to compute the distances between nine control points along the robot (including th EE), and any obstacle in the surveillance area, see Fig. (4.32).

The robot desired task is to preserve its EE to a fixed prescribed Cartesian point while pointing in a direction parallel to the x-axes with relaxed angle $\alpha_d = 5°$ using an equality constraint. During the experiment an object will be moved close to the robot frequently. The results of the experiment is presented in the following paragraph, while the robot behavior is included in a video at this link. Note that, the TPM approach can be integrated easily to the proposed general control scheme in Fig. (2).

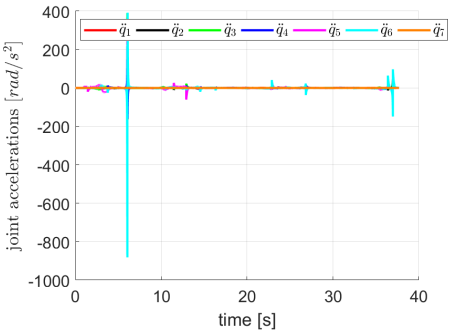
The experiment last for around 80 s, where an operator moves an obstacle toward both the EE the robot body several times. In Fig. 4.33, the high peaks in the position and pointing errors are in order to avoid any possible collision. Since the pointing task has a lower priority, its error in general is higher than the



(a)



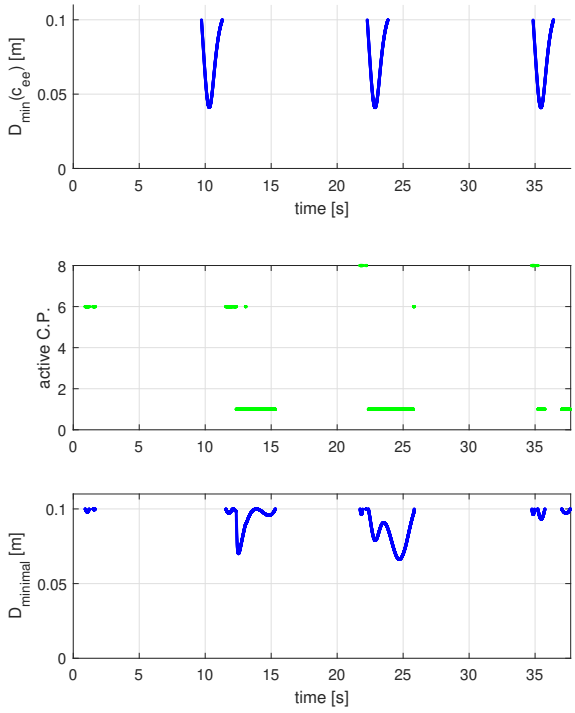
(b)



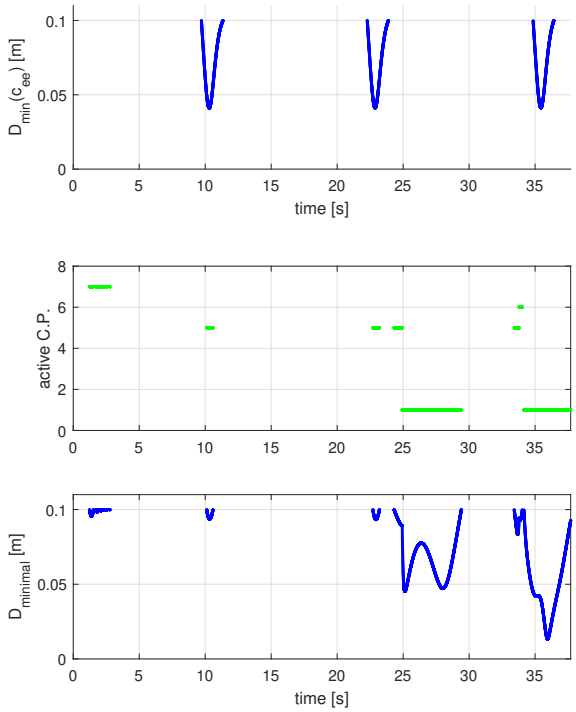
(c)

Figure 4.26: Simulation using three dimensional task for control points collision avoidance: Joints positions, velocities and accelerations.

positional one. For both, the EE and other robot control points, the minimum distance to the obstacle does not exceed 0.1 m, as shown in Fig. 4.34. This distance can be increased by raising the corresponding repulsive vector intensity, taking into account that for too large value, the robot behavior will lose its smoothness. Since the robot initial position is near to the desired Cartesian point, the joints in Fig. 4.35, are almost static during the first 10 s of the experiment. Then, the joints move in order to avoid the moving obstacle while preserving the main tasks. Since, the dynamic obstacle moves near to the upper part of the robot, the second and third joints get the main load for collision avoidance task, see Fig. 4.35(b).



(a) Inequality constraint.



(b) Equality constraint.

Figure 4.27: Simulation: The distances between the robot and the nearest obstacle in the surveillance area using pointing task with two different constraints.

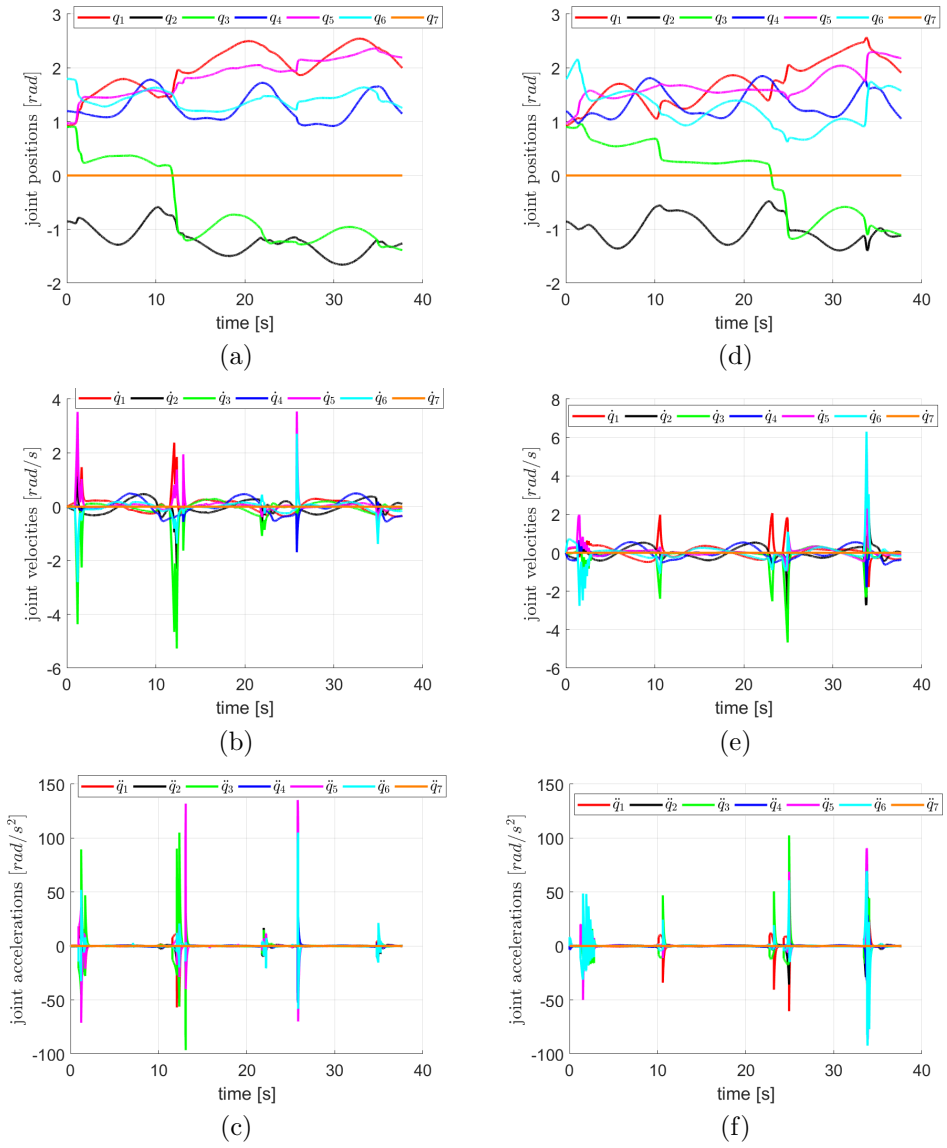
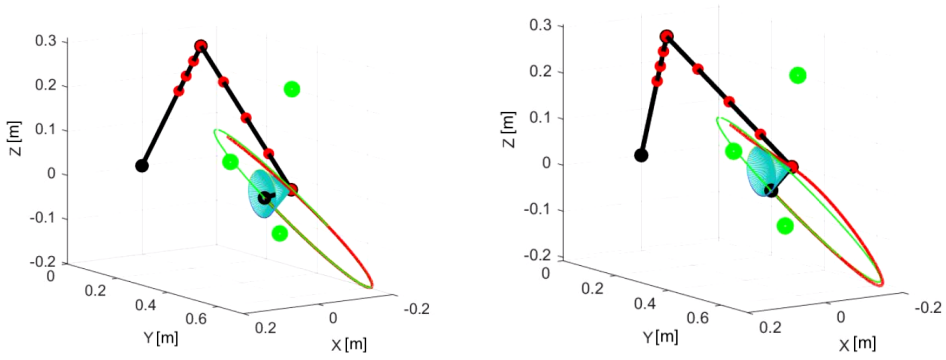
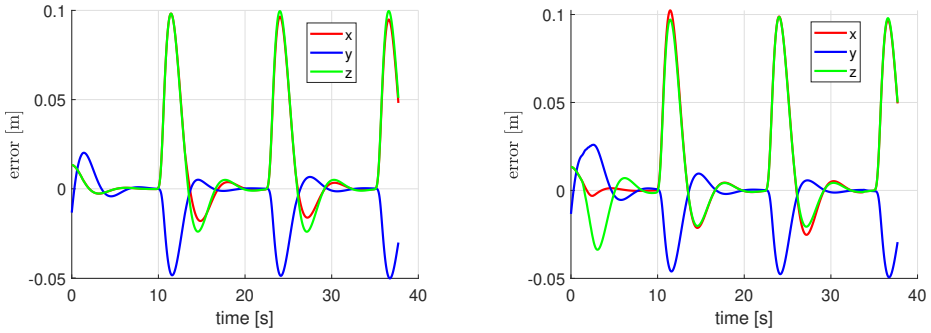


Figure 4.28: Simulation using pointing task with inequality constraint (left) vs equality constraint(right): Joints positions, velocities and accelerations.



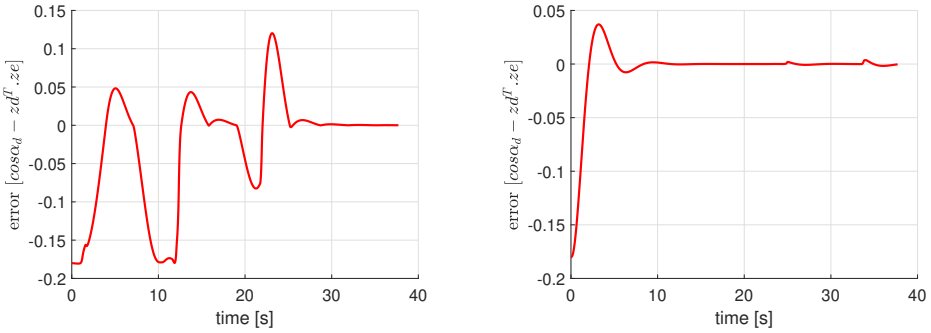
(a) (b)

Figure 4.29: (a) Using inequality constraint for relaxed pointing task; (b) Using equality constraint. The cyan cone represent the desired pointing task coin, where for better visualization, its apex is shifted to the sixth joint frame position.



(a) (b)

Figure 4.30: EE positional error using pointing with: (a) inequality constraint, and (b) equality constraint.



(a) (b)

Figure 4.31: EE pointing error using: (a) inequality constraint, and (b) equality constraint.

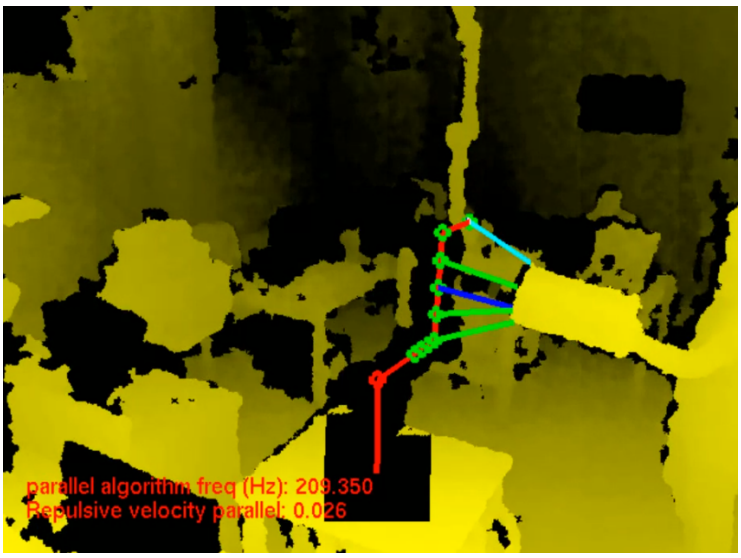
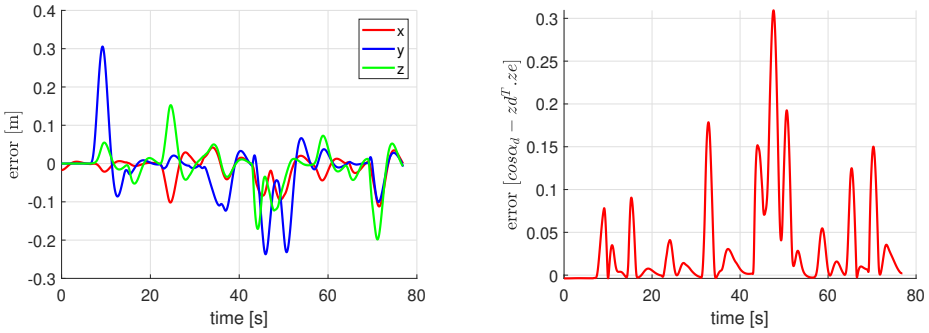
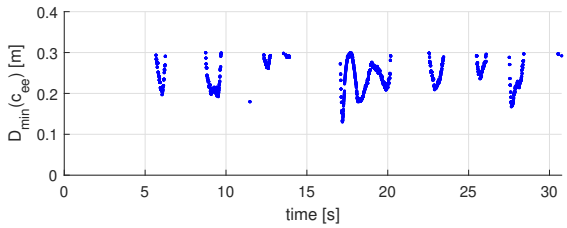


Figure 4.32: Snapshot of a depth image superposed with the computed distances between control points (green circles) on the robot arm and a close object in the surveillance area. The blue line refers to the control point of minimum distance to the obstacle. The picture shows also the computed distance between the end-effector and the same obstacle (cyan line).

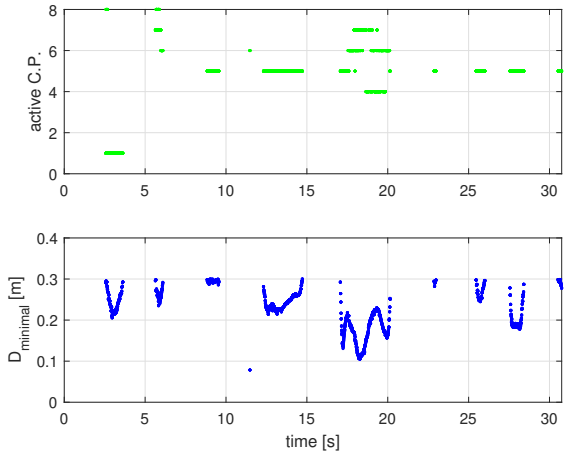


(a) (b)

Figure 4.33: An experiment using our proposed approach in Sec. 4.6.1: (a) and (b) represent the errors of the EE positional and pointing tasks, receptively.

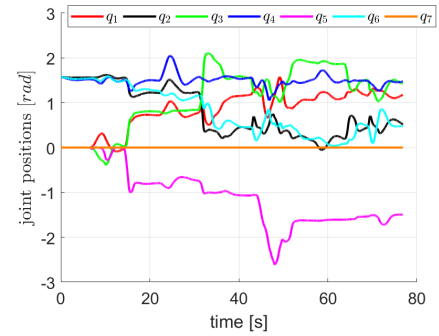


(a)

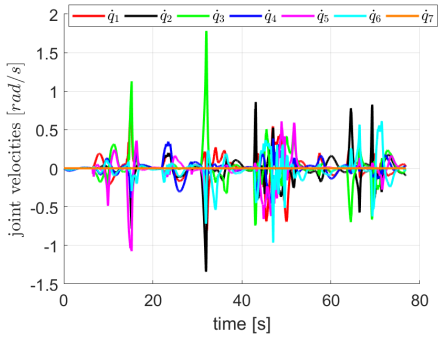


(b)

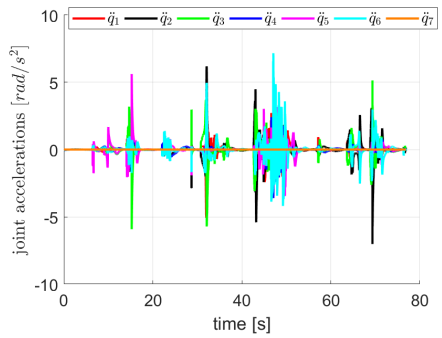
Figure 4.34: An experiment using our proposed approach: the distances between the robot and the nearest obstacle in the surveillance area.



(a)



(b)



(c)

Figure 4.35: An experiment using our proposed approach: Joints evolution.

Conclusion

Within the general hierarchical control architecture for pHRI in Fig. 1, a multi-sensor control system for safe and friendly human-robot interaction is proposed to achieve contactless collaboration according to desired coordination tasks. The proposed system is useful for different stages along the general manufacturing lines, e.g. in the quality assurance phase, where the operator should check a desired level of quality in a service or product. To design a robust and efficient solution, contactless collaboration is separated into four main subproblems.

The first problem is how to localize the collaborator position and orientation continuously and accurately (Chapter 1). Several RGB-D camera-based solutions are presented and compared. Furthermore, different enhancements are proposed to get a reliable method. That is, an RGB camera attached to a moving human is efficiently localized online by an enhanced method that combines the ARToolkit library and data processing by EKF and IMU. Alternatively, besides the human pose localization, the awareness between the robot and the human can be bidirectional using a stereo camera attached to the Oculus Rift HMD. In this case, a mixed-reality interface is built to provide the operator with different information about the current robot task and let him communicate directly with the robot, using the supported controller, to change the desired task as preferred at any time during the collaboration.

The second problem is how to define suitable Cartesian tasks for the desired contactless and coordinated collaboration (Chapter 2). The coordinated motion of the robot with the human is obtained via a special task definition, which involves three positional variables and only one angular component. The tasks are defined in order to follow the operator head motion while pointing to it. This can be done either in a regulation mode or by tracking a specified circular path. If the desired task is out of the robot workspace, a sphere of task limit is presented for task adjustment. Furthermore, a relaxed pointing task is proposed using either equality or inequality constraint to decrease the coordination task dimension as much as possible. This increases the robot dexterity and manipulability to perform the collaboration while avoiding any obstacle.

The third problem is how to keep the desired collaboration safe from any collision with the collaborator himself or any other static or dynamic obstacle

(Chapter 3). In the proposed scheme, we consider several control points along the robot body, including the EE, and compute the distances, between these points and any object in the monitoring area, online in the depth space of a fixed RGB-D sensor. Then, these distances are involved in different possible definitions of the collision avoidance task. For this, using a relaxed collision avoidance definition, that has one dimension only, is recommended. In this case, the robot preserves more dofs to keep performing the main tasks while avoiding any obstacle.

The last considered problem is how to handle robot redundancy to achieve the desired coordination tasks while keeping far from any collision (Chapter 4). For this, several control laws have been proposed and compared. First, it is shown how the pointing task can be integrated with the positional task, to achieve the desired coordination, using either the task augmentation technique or the null-space projection. Then, including the collision avoidance task, the SNS algorithm for strict prioritized task control is presented at the acceleration level. Using this algorithm, a complete demonstration for the proposed control scheme in Fig. 2 is done. Next, as an alternative to the SNS algorithm, different improvements for the task priority matrix approach are proposed. The TPM method is developed at the acceleration level to eliminate any discontinuity in the joint velocities. Also, a simple algorithm for ordering task priority is proposed. For this, soft constraints for both pointing and collision avoidance tasks are involved. Several comparisons between different task constraints are presented. Different simulations and experiments using KUKA LWR robot are done to validate all presented control schemes.

Various enhancements could be done to boost the proposed control system. First, the mixed-reality environment can be supported by various useful augmented objects. For example, the desired EE task and task limits sphere can be added. Furthermore, user control options can be increased by including different control strategies. Also, it possible to let the operator design the desired Cartesian task as a pre-process before starting the collaboration.

In addition, to improve the collision avoidance performance, a second fixed depth camera can be used to avoid gray zones or sensor occlusion. Since, a stereo camera is used for building the mixed-reality interface, it is possible to investigate how to exploit it for robot collision avoidance. Finally, using TPM approach with different Cartesian/joint task constraints could be explored.

A

Notes on the KUKA LWR IV robot

The KUKA LWR IV robot (*Light Weight Robot*) considered in this work Fig. A.1, is usually used for research development and consists of seven revolute elastic joints. However, in all study cases of this work, it is considered as a rigid robot. Its total weight is approximately 16 kg, with a rated payload of 7 kg. All joints are equipped with position sensors on the motor and link sides, and with a joint torque sensor. Table A.1 contains the position, velocity, and torque limits for each joint. The robot kinematics are computed according to the link frames in Fig. A.1. The associated parameters are given in Tab. A.2. The constructor, KUKA, has still not released a public version of its dynamic model. However, it can be computed depending on the reverse engineering approach by Gaz et al. [2014].

For simulations, we used the robot model provided by the V-REP. For the supplied KUKA LWR robot model, several aspects make its behavior different from the real system Cefalo [2015]. First, the motor dynamics and the low level electronic controllers are not modeled. Also, the friction is neglected and the mass distribution is considered uniform. In this work all simulations done with KUKA LWR were in the static mode.

For the experiments, the KUKA LWR IV system in Fig. A.2 is used. The robot controller is connected to a remote PC node via an Ethernet connection and the Fast Research Interface library (FRI) by KUK [2011] is used through ROS/ROS 2 node with C++ programming language to set up the desired control architecture. The robot system can work with the sampling rates $1, 2$ and 5 [ms].

Table A.1: Joint position, torque and velocity limits of the KUKA LWR IV.

Joint number	Range of motion [rad]	Maximum torque [Nm]	Maximum velocity [rad/s]
1	± 2.97	176	1.92
2	± 2.09	176	1.92
3	± 2.97	100	2.23
4	± 2.09	100	2.23
5	± 2.97	100	3.56
6	± 2.09	38	3.21
7	± 2.97	38	3.21

Table A.2: Denavit-Hartenberg parameters of the KUKA LWR IV.

Link number	a_i [m]	α_i [rad]	d_i [m]	θ_i [rad]
1	0	$\pi/2$	0	q_1
2	0	$-\pi/2$	0	q_2
3	0	$-\pi/2$	$d_3 = 0.4$	q_3
4	0	$\pi/2$	0	q_4
5	0	$\pi/2$	$d_5 = 0.39$	q_5
6	0	$-\pi/2$	0	q_6
7	0	$\pi/2$	$l + d_7 = l + 0.078$	q_7

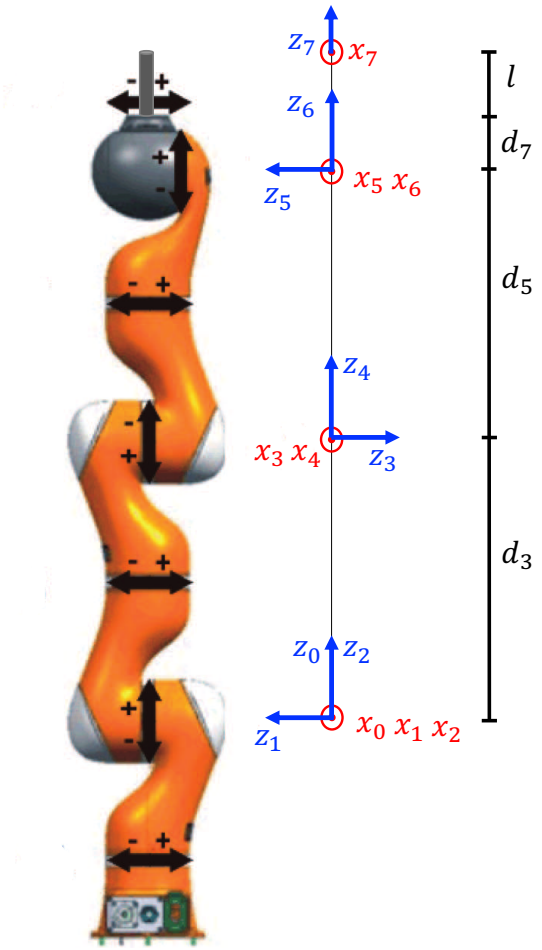


Figure A.1: Denavit-Hartenberg frames of the KUKA LWR IV: All x-axes point toward the viewer (frames are displaced sideways for better clarity).

Figure A.2: The KUKA LWR system at DIAG Robotics Lab. (A) The LWR body. (B) The KR C2 lr robot controller unit. (C) PC node. (D) Kuka control panel.

Bibliography

P.J. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(12):239–256, 1992.

O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 341–346, 1999.

G. Buondonno and A. De Luca. A recursive newton-euler algorithm for robots with elastic joints and its application to control. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 5526–5532, 2015.

G. Buondonno and A. De Luca. Efficient computation of inverse dynamics and feedback linearization for vsa-based robots. *IEEE Robotics and Automation letters*, 1(2):908–915, 2016.

M. Cefalo. Notes on the KUKA LWR4 dynamic model, 2015. URL http://www.coppeliarobotics.com/contributions/LBR4p_dynamic_model.pdf.

S. Chiaverini, G. Oriolo, and I.D. Walker. Kinematically redundant manipulators. In *Handbook of Robotics*, pages 245–268. Springer, 2008.

A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. 9th IEEE Int. Conf. on Computer Vision*, pages 1403–1410, 2003.

A. De Luca and L. Ferrajoli. A modified newton-euler method for dynamic computations in robot fault detection and control. In *Proc. Int. Conf. IEEE on Robotics and Automation*, pages 3359–3364, 2009.

A. De Luca and F. Flacco. Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration. In *Proc. 4th IEEE Int. Conf. on Biomedical Robotics and Biomechatronics*, pages 288–295, 2012.

A. De Luca and R. Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 999–1004, 2005.

A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 1623–1630, 2006.

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.

F. Fabrizio and A. De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, 2016.

F. Flacco. The tasks priority matrix: a new tool for hierarchical redundancy resolution. In *Proc. Int. Conf. IEEE-RAS 16th on Humanoid Robots*, pages 1–7, 2016.

F. Flacco and A. De Luca. Multiple depth/presence sensors: Integration and optimal placement for human/robot coexistence. In *Proc. Int. Conf. IEEE on Robotics and Automation*, pages 3916–3923, 2010.

F. Flacco, A. De Luca, and O. Khatib. Prioritized multi-task motion control of redundant robots under hard joint constraints. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 3970–3977, 2012a.

F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 338–345, 2012b.

F. Flacco, A. De Luca, and O. Khatib. Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Trans. on Robotics*, 31 (3):637–654, 2015a.

F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach for evaluating distance to objects – with application to human-robot collision avoidance. *J. of Intelligent & Robotic Systems*, 80, Suppl. 1:7–22, 2015b.

S.Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris. End-user robot programming using mixed reality. In *Proc. Int. Conf. IEEE on Robotics and Automation*, 2019.

C. Gaz and A. De Luca. Payload estimation based on identified coefficients of robot dynamics—with an application to collision detection. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 3033–3040, 2017.

C. Gaz, F. Flacco, and A. De Luca. Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1386–1392, 2014.

C. Gaz, E. Magrini, and A. De Luca. A model-based residual approach for human-robot collaboration during manual polishing operations. *Mechatronics*, 55:234–247, 2018.

M. Geravand, F. Flacco, and A. De Luca. Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. In *Proc. Int. Conf. IEEE on Robotics and Automation*, pages 4000–4007, 2013.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.

S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 3356–3363, 2008.

S. Haddadin, A. De Luca, and A. Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Trans. on Robotics*, 33(6): 1292–1312, 2017.

M. Hägele, K. Nilsson, J.N. Pires, and R. Bischoff. Industrial robotics. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics (2nd Ed.)*, pages 1385–1421. Springer, 2016.

H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proc. 2nd IEEE/ACM Int. Work. on Augmented Reality*, pages 85–94, 1999.

M. Khatib, K. Al Khudir, and A. De Luca. Visual coordination task for human-robot collaboration. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3762–3768, 2017.

M. Khatib, K. Al Khudir, and A. De Luca. Multi-sensor control system for safe human-robot collaboration with mixed-reality interface. *In preparation to Robotics and Computer-Integrated Manufacturing*, 2019a.

M. Khatib, K. Al Khudir, and A. De Luca. Robot collision avoidance using tasks priority matrix with soft constraints at the acceleration level. *In preparation to IEEE Robotics and Automation Letters*, 2019b.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*, pages 396–404. Springer, 1986a.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. of Robotics Research*, 5(1):90–99, 1986b.

G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. 6th IEEE/ACM Int. Symp. on Mixed and Augmented Reality*, pages 225–234, 2007.

KUKA.FastResearchInterface 1.0. KUKA System Technology (KST), D-86165 Augsburg, Germany, 2011. Version 2.

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

J. Lee, N. Mansard, and J. Park. Intermediate desired value approach for task transition of robots in kinematic control. *IEEE Trans. on Robotics*, 28(6): 1260–1277, 2012.

Y. Lu. Industry 4.0: A survey on technologies, applications and open research issues. *J. of Industrial Information Integration*, 6:1–10, 2017.

E. Magrini and A. De Luca. Hybrid force/velocity control for physical human-robot collaboration tasks. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 857–863, 2016.

E. Magrini and A. De Luca. Human-robot coexistence and contact handling with redundant robots. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 4611–4617, 2017.

E. Magrini, F. Flacco, and A. De Luca. Estimation of contact forces using a virtual force sensor. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 2126–2133, 2014.

E. Magrini, F. Flacco, and A. De Luca. Control of generalized contact motion and force in physical human-robot interaction. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2298–2304, 2015a.

E. Magrini, F. Flacco, and A. De Luca. Control of generalized contact motion and force in physical human-robot interaction. In *Proc. Int. Conf. IEEE on Robotics and Automation*, pages 2298–2304, 2015b.

E. Magrini, F. Ferraguti, A.J. Ronga, F. Pini, A. De Luca, and F. Leali. Human-robot coexistence and interaction in open industrial cells. *Robotics and Computer-Integrated Manufacturing*, 61:101846, 2020.

J. Mainprice, E.A. Sisbot, T. Siméon, and R. Alami. Planning safe and legible hand-over motions for human-robot interaction. In *Proc. IARP Work. on Technical Challenges for Dependable Robots in Human Environments*, 2010.

E. Mariotti, E. Magrini, and A. De Luca. Admittance control for human-robot interaction using an industrial robot equipped with a f/t sensor. In *Proc. Int. Conf. IEEE on Robotics and Automation*, pages 6130–6136, 2019.

G. Milighetti, L. Vallone, and A. De Luca. Adaptive predictive gaze control of a redundant humanoid robot head. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3192–3198, 2011.

C. Nehaniv, K. Dautenhahn, J. Kubacki, M. Haegele, C. Parlitz, and R. Alami. A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 371–377, 2005.

Oculus. URL https://developer.oculus.com.

M.P. Polverini, A.M. Zanchettin, and P. Rocco. Real-time collision avoidance in human-robot interaction based on kinetostatic safety field. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, pages 4136–4141, 2014.

O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann. Using gesture and speech control for commanding a robot assistant. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 454–459, 2002.

J. Serafin and G. Grisetti. NICP: Dense normal based point cloud registration. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 742–749, 2015.

B. Siciliano and J. J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Proc. 5th Int. Conf. on Advanced Robotics*, pages 1211–1216, 1991.

B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, planning and control*. Springer, 2010.

M. Svenstrup, S. Tranberg, H.J. Andersen, and T. Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3571–3576, 2009.

I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1572–1577, 1998.

Unity. URL https://unity.com.

L. Villani, A. De Santis, V. Lippiello, and B. Siciliano. Human-aware interaction control of robot manipulators based on force and vision. In *Proc. 7th Work. on Robot Motion Control*, pages 209–225. Lecture Notes in Control and Information Sciences, vol. 396, Springer, 2009.

ZED-Mini. URL https://www.stereolabs.com/zed-mini.

L. Zlajpah and B. Nemec. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In *Proc. Int. Conf. IEEE/RSJ on Intelligent Robots and Systems*, volume 2, pages 1898–1903, 2002.