ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET MATHÉMATIQUES

Année : 2013 N° B.U.:

THÈSE DE DOCTORAT

Spécialité : INFORMATIQUE, AUTOMATIQUE, ÉLECTRONIQUE ET GÉNIE ÉLECTRIQUE

Présentée et soutenue publiquement par :

Amine ABOU MOUGHLBAY

le 21 mai 2013

à l'École Centrale de Nantes

TITRE

Contributions à l'enchaînement des tâches et à la résolution de la redondance
Application aux robots humanoïdes et multi-bras

JURY

Président	Wisama KHALIL	*Professeur à l'École Centrale de Nantes - France*
Rapporteurs	Angel P. DEL POBIL	*Professeur à l'Université de Jaume I, Castellón - Espagne*
	Nicolas ANDREFF	*Professeur à l'Université de Franche-Comté, Besançon - France*
Examinateur	Philippe MARTINET	*Professeur à l'École Centrale de Nantes - France*

Directeur de thèse: Philippe MARTINET
Laboratoire : Institut de Recherche en Communications et Cybernétique de Nantes (UMR CNRS 6597)

N° **ED: 503-188**

ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET MATHÉMATIQUES

Année : 2013 N° B.U.:

THÈSE DE DOCTORAT

Spécialité : INFORMATIQUE, AUTOMATIQUE, ÉLECTRONIQUE ET GÉNIE ÉLECTRIQUE

Présentée et soutenue publiquement par :

Amine ABOU MOUGHLBAY

le 21 mai 2013

à l'École Centrale de Nantes

TITRE

Contributions à l'enchaînement des tâches et à la résolution de la redondance
Application aux robots humanoïdes et multi-bras

TITLE

Contributions in task sequencing and redundancy resolution
Application to humanoid and multi-arm robots

JURY

Président	Wisama KHALIL	*Professeur à l'École Centrale de Nantes - France*
Rapporteurs	Angel P. DEL POBIL	*Professeur à l'Université de Jaume I, Castellón - Espagne*
	Nicolas ANDREFF	*Professeur à l'Université de Franche-Comté, Besançon - France*
Examinateur	Philippe MARTINET	*Professeur à l'École Centrale de Nantes - France*

Directeur de thèse: Philippe MARTINET
Laboratoire : Institut de Recherche en Communications et Cybernétique de Nantes (UMR CNRS 6597)

N° **ED: 503-188**

Abstract

Redundancy is either intrinsic and explicit in the robot's mechanical architecture, and/or implicit and appears only when applying specific tasks. The purpose of this thesis is to show how redundancy can be considered as an exploitable advantage of the system, rather than a problem to avoid, as it was classically perceived. In this aim the classification and identification of redundancy is performed, then the kinematic resolution of the redundancy problem is developed taking into account several simultaneous and prioritized tasks on a general redundant system.

The comparison and discussion of the existing resolution methods' efficiency in several cases will lead to the development of a new technique for redundancy resolution, which overcome the encountered problems and improve the system's behavior and performance. The feasibility and capability of the existing and developed techniques are tested by simulation on various robots in different configurations and with multiple case studies on the executed tasks.

Furthermore, a kinematic/dynamic multi-control points approach is developed to control any robotic system, first by studying and monitoring the robot/environment interaction on several points using various types of sensors, and then by decomposing the desired robotic scenario into several elementary prioritized tasks (defined in a generic form) which are finally pushed into one of the presented task sequencing methods.

The developed framework and the redundancy resolution formalisms are validated by the application of several industrial, service and assistive tasks on three different platforms: a multi-arm system and two humanoid robots (HRP-2 and Nao). Using various types of embedded and external vision sensors, several techniques were integrated into these robotic platforms to apply the desired scenarios in simulation and real-time.

First, the multi-arm platform, for meat cutting and muscles separation, is controlled to apply several simultaneous tasks (cutting, pulling, visibility) while respecting the system's constraints such as collision and occlusion avoidance. Second, a localization of the Nao robot, in an indoor environment, allows it to navigate and pick up an object from the floor. Third, a robust tracking technique is used to control HRP-2 and Nao robots when performing grasping of several objects in the environment.

Keywords:
Redundant robot, Redundancy resolution, Multi-arm system, Sensor-based control, Multi-control points, Manipulation, Localization, Nao, HRP-2.

Résumé

La redondance est présente dans toutes les plateformes robotiques, elle est soit intrinsèque et explicite dans l'architecture mécanique du robot, ou implicite et n'apparaît que lors de l'exécution de certaines tâches. L'objectif de cette thèse est de considérer la redondance comme un avantage à exploiter, plutôt qu'un problème à éviter, comme il a été classiquement perçu. Cette thèse identifie, classifie les différents types de redondances et traite leur résolution cinématique lors de l'application de plusieurs tâches simultanées sur un système redondant avec une priorité spécifique.

La comparaison et discussion sur l'efficacité des méthodes existantes dans plusieurs configurations nous amène à l'élaboration de nouvelles techniques généralisées pour la résolution de la redondance. Elles permettent de surmonter les problèmes rencontrés et d'améliorer le comportement et les performances du système. La faisabilité et l'efficacité de ces techniques ont été évaluées par simulation sur plusieurs robots dans des différentes configurations et avec plusieurs définitions de tâches.

En outre, une approche multi-points de contrôle est développée, en cinématique et dynamique, pour commander tout système robotisé. Elle assure l'interaction entre le robot et son environnement en plusieurs points à l'aide de divers types de capteurs. L'application désirée est alors décomposée en plusieurs tâches élémentaires et génériques qui sont finalement envoyées vers un environnement d'enchaînement de tâches.

L'approche élaborée et les formalismes de résolution de la redondance ont été validés par l'application de plusieurs tâches de service et d'assistance sur un système multi-bras et deux robots humanoïdes HRP-2 et Nao. En effet, en utilisant différents types de capteurs, notamment des systèmes de vision embarqués et déportés, plusieurs techniques ont été implémentées sur ces plateformes robotiques pour appliquer les tâches désirées en simulation et en temps réel.

La commande de la plateforme multi-bras pour la découpe et la séparation de muscles de viande permet d'appliquer plusieurs tâches simultanées (coupe, traction, visibilité) tout en respectant les contraintes du système comme l'évitement de la collision et de l'occlusion. En plus, la localisation du robot Nao lui permet de naviguer dans un environnement d'intérieur pour ramasser un objet du sol; une technique robuste de suivi et de saisie est utilisée pour amener les robots HRP-2 et Nao à saisir plusieurs objets.

Mots Clés:
Robot redondant, Résolution de la redondance, Système multi-bras, Commande référencée capteurs, Multi-points de contrôle, Manipulation, Localisation, Nao, HRP-2.

Contents

List of Figures

List of Tables

Acknowledgments

The research described in this thesis was carried out from October 2009 to August 2011 at ROSACE (RObotics and Autonomous Complex Systems) team of the "Laboratoire des Sciences et Matériaux pour l'Electronique et d'Automatique" (**LASMEA**) in Clermont-Ferrand, and from September 2011 to May 2013 at the Robotic team of the "Institut de Recherche en Communications et Cybernétique de Nantes" (**IRCCyN**). The first three years of my thesis work were financially supported by a french research allocation, and the remainder by the project ARMS of the French National Agency of Research (ANR-10-SEGI-002).

First of all, I am deeply indebted to my supervisor Professor **Philippe MARTINET**, for his encouragement, his valuable advices, continuous and indispensable assistance, and his great help. I would like to thank him for his confidence in me, for his helpful guidance and great care. The numerous discussions and his motivating investment in the research surrounding my thesis is more than greatly appreciated. It is really my pride to perform my PhD. research under his supervision.

I would like also to express my gratitude to Professor **Wisama KHALIL** to have honored me for presiding the jury and for his interest in my work, his rigorous reading and useful critiques and suggestions allowing me to enforce certain perspectives in my research subject.

I would like to thank Professors **Angel P.DEL POBIL** and **Nicolas ANDREFF** for participating as members of the jury of my thesis and for having accepted to assess my work. Thanks for their time spent in reading, analyzing the thesis and writing their report, and thanks for their useful comments and feedback.

I wish to express my gratitude to Dr. **Enric CERVERA**, Associate Professor in the Department of Computer Science and Engineering at Jaume I University, Castelló - Spain, for working together during his stay as a visiting professor at IFMA (French Institute of Advanced Mechanics) in 2011 and at IRCCyN Laboratory in 2012. It was a great opportunity for me to have several long and deep discussions with him about my work. This fruitful collaboration allowed me to advance significantly in my thesis especially in the experimental part with numerous applications on the Nao robot. Despite his short stay, it was my pleasure to know him also at the personal plane and I hope we will have the chance to work together again.

I would like to thank Prof. **Eric MARCHAND** for receiving me in the Lagadic group at the Université de Rennes 1 for one week. This short visit was a great opportunity for me to have several discussions with him about several visual servoing issues and a continuous support during my thesis while using the ViSP software. I would like also to thank **Nicolas MANSARD** for receiving me in the Gepetto group at LAAS Laboratory in Toulouse and for his help with the HRP2 simulator. I wish to acknowledge also the help provided by Olivier Stasse, Thomas

Moulard and all researchers and PhD students in these groups.

I would like to thanks all professors and students that I worked with them while giving tutorials and lab works in Blaise Pascal University and Ecole Centrale de Nantes. I would like also to thank my colleagues and all the staff in IRCCyN and LASMEA for their support and friendship including those who have shared their office with me giving me the chance to discuss several issues with them. I would also like to thanks **Philip Long** for the several useful discussions, for the great moments we spent working in ARMS project and of course for his friendship as well as his efforts in correcting my English writings.

I would like to deeply thanks my friends especially in Nantes, Clermont-Ferrand and in the Faculty of Engineering of Lebanese University for the unforgettable moments we spent together. Finally, I owe a huge debt of gratitude and sincere thanks to my parents and family, they have lost a lot due to my research abroad. Without their encouragement and support it would have been impossible for me to finish this work. Thank you.

Amine Abou Moughlbay

Notation List

Generalities

$\mathbf{I_n}$	Identity matrix of dimension $n \times n$
$\mathbf{0_n}$	Null matrix of dimension $n \times n$
$\dot{\mathbf{A}}$	Time derivative of A
$\ddot{\mathbf{A}}$	Second time derivative of A
$\mathbf{A}_\times$	Skew symmetric matrix of a vector A
$\mathbf{A}^+$	Moore-Penrose pseudo-inverse of $\mathbf{A}$ matrix
$\mathbf{A}^\#$	Generalized pseudo-inverse of $\mathbf{A}$ matrix
$\mathbf{A}^{+\lambda}$	Damped least square inverse of $\mathbf{A}$ matrix
$\mathbf{A}^{\#B}$	B-matrix weighted pseudo-inverse of $\mathbf{A}$ matrix
$\mathbf{A}^\top$	Transpose of the $\mathbf{A}$ matrix/vector
$Ker(A)$	Null space of A matrix

Vision

$\mathbf{s}$	Feature vector
$\mathbf{s}^*$	Desired feature vector
$\mathbf{e}$	Error vector
$\mathbf{L_s}$	Interaction matrix corresponding to $\mathbf{s}$
$\mathcal{F}_\mathbf{s}$	Sensor's frame
$\mathcal{F}_\mathbf{cp}$	Control point's frame
$\dot{\mathbf{s}}$	Feature vector variation
ViSP	Visual Servoing Platform
MBT	Model Based Tracker
PCL	Point Cloud Library

Robotics

n	Dimension of the joint space
m	Dimension of the operational space
t	Dimension of the task
${}^\mathbf{a}\mathbf{T_b}$	Homogeneous transformation matrix of frame b wrt. frame a
${}^\mathbf{a}\mathbf{R_b}$	Rotation matrix from frame b to frame a
${}^\mathbf{a}\mathbf{t_b}$	Position vector from frame b to frame a

$\theta\mathbf{u}$	Angle-axis representation for the orientation (rotation of an angle θ around the **u** axis)
$\mathbf{q}$	Vector of joint angles
$\mathbf{X}$	Vector of 3D pose (position/orientation)
$\dot{\mathbf{X}}$	Vector of 3D velocity (linear/angular)
$\mathbf{v}$	Velocity tensor
$\mathbf{J_q}$	Jacobian Matrix
$\mathbf{J_{ext}}$	Matrix of extended Jacobian
$\mathbf{X_{ext}}$	Vector of extended task (position/orientation)
$\dot{\mathbf{X}}_{\mathbf{ext}}$	Vector of extended task velocity
$\mathbf{T_i}$	i^{th} Task
$\mathbf{P}$	Generalized projection matrix
$\mathbf{z}$	Secondary task vector
$\mathbf{z_i}$	i^{th} element of the secondary task vector
$\mathbf{P_e}$	Orthogonal projection matrix
$\mathbf{P_z}$	Directional projection matrix
$\mathbf{P}_\eta$	Projection matrix corresponding to the minimun norm solution
$\mathbf{W}$	Weighting matrix in the joint space
w_i	i^{th} element of the weighting matrix in the joint space
$\mathbf{H}$	Weighting matrix in the task space
h_i	i^{th} element of the weighting matrix in the task space
DOF	Degree(s) of Freedom
DGM	Direct Geometric Model
CP	Control Point

I

Introduction

Contents

A robot is an automatically controlled, reprogrammable, multipurpose mechanical system with several degrees of freedom, which may be either fixed in place or mobile. It has been widely used so far in various industrial automation applications. Since the last decade, other areas of application have emerged: medical, service (spatial, civil security, ...), transport, underwater, entertainment ..., where the robot either works in an autonomous manner or in cooperation with an operator to carry out complex tasks in a more or less structured environment [KD04].

In addition to single-arm serial robots, several multi-arm systems were designed to perform advanced complex cooperative tasks in industrial and domestic environments. Furthermore, to imitate the human body structure, dual-arm manipulators and humanoid robots were

designed and adapted to perform the desired robotic applications. Some examples of these systems and their development is presented in the first section.

Later on, in the second section, examples on these robots applications are presented: the control approaches which are used to perform service tasks, and especially the manipulation ones, are discussed. Furthermore, a presentation and comparison between the classical joint space and operational space control is given in the third section. Finally, a presentation of the objectives and motivations of this study are presented with a brief descriptions of the next chapters.

I.1 Examples of Robot Architecture

I.1.1 Multi-Arm Systems

In the last decade, advantages of multi-arm systems over the use of single-arm systems [SK08] have been recognized as a key factor in achieving a higher level of flexibility and productivity of robotic workcells. It has been recognized that many tasks that are difficult or impossible to execute by a single robot become feasible when two or more manipulators are employed in a cooperative way. Such tasks include, for instance, carrying heavy or large payloads, the assembly of multiple parts without using special fixtures, and handling of objects that are flexible or possess extra DOF.

Examples of research work in the early days include that by Fujii and Kurono [FK75], Nakano et al. [NOIK74], and Takase et al. [TISH74] where important key issues in the control of multi-arm robots were investigated. In the 1980s, based on several fundamental theoretical results for single-arm robots, strong research on multi-arm robotic systems was renewed. Definition of task vectors with respect to the object to be handled [DZ85], dynamics and control of the closed kinematic chain formed by the multi-arm robot and the object [McC86, TBY88], and force control issues, such as hybrid position/force control [Hay86, UD92] were explored (Fig. I.1 - I.2). Through this research work, a strong theoretical background for the control of multi-arm robots has been formed, providing the basis for research on more advanced topics from the 1990s.

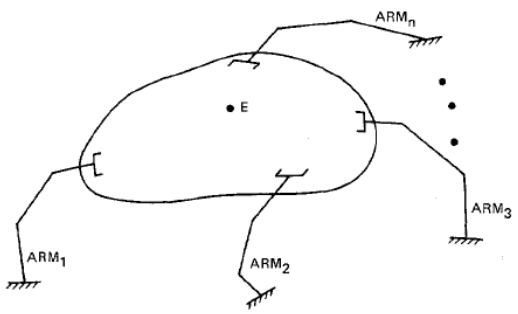


Figure I.1 – Schematic drawing of a multi-arm cooperating robot affecting a single object [Hay86]

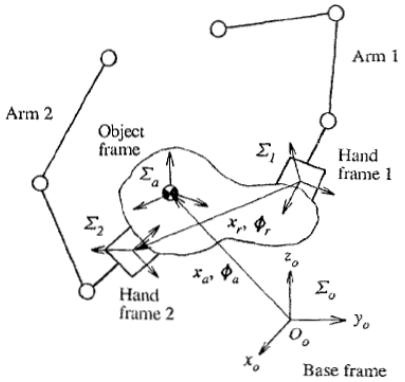


Figure I.2 – An example of workspace coordinates defined for two-arm robot [UD92]

Figure I.3 – A multi-arm robotic system for muscle separation [ARM] (ARMS Project)

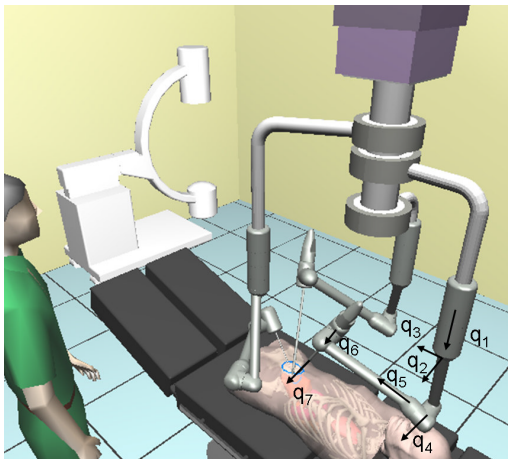


Figure I.4 – The DLR medical operating scenario with three KineMedics disposing of 7 joints each, mounted to an actuated carrier [KOH+06]

Examples of cooperative multi-arm systems that can be adopted to achieve higher levels of flexibility and efficiency of industrial workcells are studied in [TPD09] for an automatic disassembly workcell composed of two manipulators, a rotating table, a tool changer and a deposit zone, and in [AHN+08] where a dual-arm system measures the intensity of the radioactivity of the wastes of dismissed reactors. Another example is the ARMS project [ARM] which deals with the robotization of bovine muscle separation in meat cutting and transformation processes, using a multi-arm robotic system for muscle handling, cutting and active vision (Fig. I.3).

Aside from industrial robotics, medical robotics [KOH+06] is also an important application area of cooperative working of multi-arm robot systems as in [CDW+12] where a multi-arm medical robot assisted maxillofacial surgery using optical navigation was proposed to improve surgical precision and keep the doctors away from the heavy manual work (Fig. I.4).

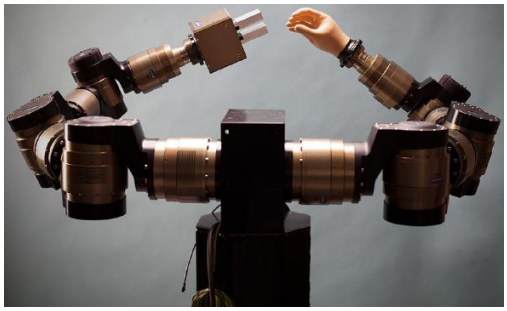
I.1.2 Dual-Arm Manipulators

There is an increasing trend of robots being moved into environments originally designed for human use. In industry, anthropomorphic robots of human size are expected to replace human workers without major redesigns of the workplace. The ability to use human and robot workers interchangeably is thought to be the key to low-cost, flexible automation. This has, during the past few years, led to increased interest for the field of anthropomorphic or dual-arm manipulation.

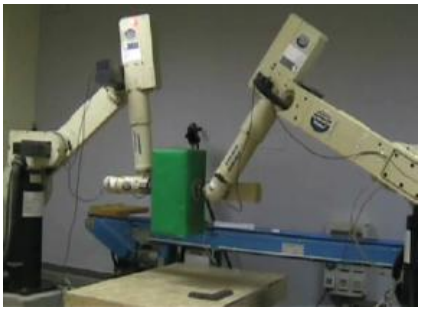
Robot manipulation in its basic forms is a well-studied field that has seen remarkable developments in the last 50 years. Some of the very first robotic manipulators were dual-arm systems. Early examples include the manipulators constructed by Goertz in the 1940's and 1950's for handling of radioactive goods [Goe52] that were used in pairs with the operator controlling one with each hand. The late 1950's also saw dual-arm teleoperation setups for deep-sea exploration [Fle60]. NASA's Johnson Space Center started experimenting with anthropomorphic dual-arm teleoperators in 1969 [AAA+00].

a - Mobile Dual Manipulator PR2

b - Dual manipulator Motoman SDA



c - Semi-anthropomorphic robot at CAS/KTH



d - Double single-arm at PRISMA Lab

Figure I.5 – Examples of dual-arm manipulators

While some research is carried out on systems built by simply placing two single-arm manipulators to share the same workspace [YTU99, CCMV08], considerable effort has also been put into constructing dedicated dual-arm platforms. Some of these put the effort on manipulation capability and target industrial manufacturing applications, such as the Toyota Dual-Arm Robot [YNS95], the Yaskawa Motoman SDA10D [Blo10], the ABB Frida [KJLH11] the Korea Institute of Machinery and Materials dual-arm robot [PPPK09], the SHARP household robot [NYK+06], and the Pi4 Workerbot [KSS11], each of which is shaped like a "torso", and primarily intended for stationary deployment (Fig. I.5).

As mentioned by [SKN+12], future direction in robotic systems in general, and in dual-arm manipulation in particular, will be the integration of elements from systems theory with tools from cognitive methodologies. These will involve the consideration of vision and learning capabilities in the actual feedback design. This seems necessary in advanced collaborative control tasks where the manipulators have incomplete knowledge of the environment, and might have been assigned their tasks independently. For more details, the history of dual-arm manipulators and manipulation has been widely presented in several earlier review papers [Whi87, DN90, SKN+12].

I.1.3 Humanoid Robots

Several dual-arm systems have analogous kinematics to that of the human body: torso, arms, legs, end-effectors similar to human hands and a 'head' which, eventually, includes an artificial vision system. These robots are known as humanoids.

There is a long history of mechanical systems with human form that perform human-like movements. For example, Al-Jazari designed a humanoid automaton in the 13^{th} century [Ros94], Leonardo da Vinci designed a humanoid automaton in the late 15^{th} century [Ros06], and in Japan there is a tradition of creating mechanical dolls called Karakuri ningyo that dates back to at least the 18^{th} century [Hor06].

In the 20^{th} century, advances in digital computing enabled researchers to incorporate significant computation into their robots for sensing, control, and actuation. They developed isolated systems for sensing, locomotion, and manipulation that were inspired by human capabilities. However, the first humanoid robot to integrate all these functions and capture widespread attention was WABOT-1, developed by Ichiro Kato et al. at Waseda University in Japan in 1973 (Fig. I.6). Since then the WABOT robots integrated functions that have been under constant elaboration: visual object recognition, speech generation, speech recognition, bimanual object manipulation, and bipedal walking.

In 1986, Honda began a confidential project to create a humanoid biped, and in 1996 it unveiled the result: Honda Humanoid P2. This robot was the first full-scale humanoid capable of stable bipedal walking with onboard power and processing. Successive designs reduced its weight and improved performance (see Fig. I.8). Compared to humanoids built by academic laboratories and small manufacturers, the Honda humanoids were a leap forward in sturdiness, using specially cast lightweight high-rigidity mechanical links, and harmonic drives with high torque capacity.

In parallel with these developments, the decade long Cog project began in 1993 at the MIT Artificial Intelligence laboratory in the USA with the intention of creating a humanoid robot that would learn to 'think' by building on its bodily experiences to accomplish progressively more abstract tasks [BS94]. This project gave rise to an upper-body humanoid robot whose design was heavily inspired by the biological and cognitive sciences. Another milestone was the

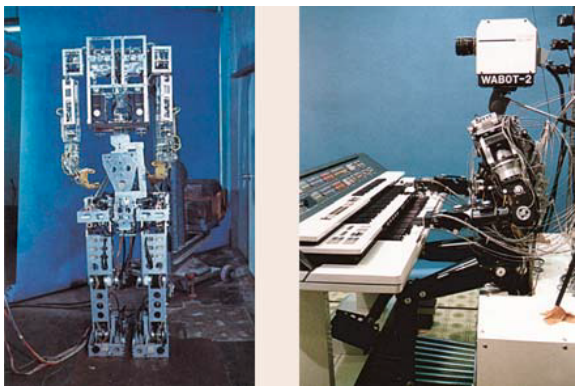


Figure I.6 – WABOT-1 (1973) and WABOT-2 (1984)

Figure I.7 – The NASA Robonaut consists of an upper body placed on a wheeled mobile base

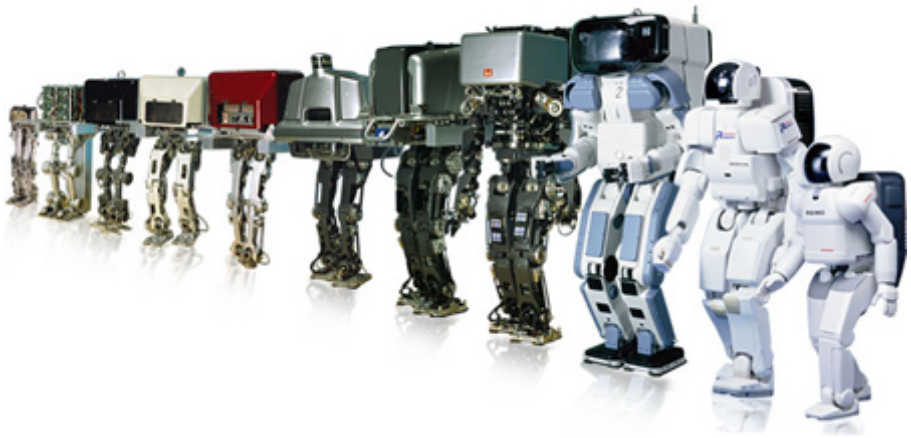


Figure I.8 – Evolution of Honda's Robots: from E0 (1986) to ASIMO (2000)

Sony Dream Robot, unveiled by Sony in the year 2000. The small humanoid robot, which was later called Qrio, was able to recognize faces, could express emotion through speech and body language, and could walk on flat as well as on irregular surfaces. Yet others place the emphasis on completely mimicking human appearance and structure, such as the Honda Asimo [HT01] and the Kawada HRP Series [KKK+02].

In the early 21st century, many companies and academic researchers have become involved with humanoid robots, and there are numerous humanoid robots across the world with distinctive features: further additions were done by HOAP and ACTROID in 2003, PERSIA and KHR-1 in 2004, PKD android and WAKAMARU in 2005, Nao and TOPIO in 2007, Justin, KT-X and NEXI in 2008, SURALP in 2009, Robonaut-2, SURENA-II and HRP-4C in 2010.

I.2 Examples of Robot Applications

The first successful applications of robot manipulators generally involved some sort of material transfer, such as injection molding or stamping where the robot merely attended a press to unload and either transfer or stack the finished part. However, the important applications of these robots are not limited to those industrial jobs where the robot is directly replacing a human worker.

In addition, there are many other applications of robotics in areas where the use of humans is impractical or undesirable. Among these are undersea and planetary exploration, satellite retrieval and repair, the defusing of explosive devices, and work in radioactive environments. Furthermore, prostheses, such as artificial limbs, are themselves robotic devices requiring methods of analysis and design similar to those of industrial manipulators [SHV06].

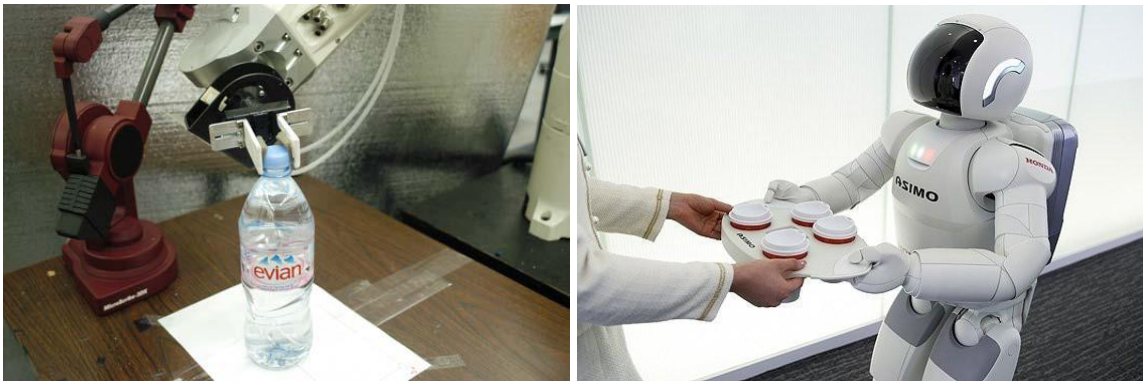
Moreover, a wide market segment comes from entertainment, where robots are used as toy companions for children, such as humanoid robots and the pet robots being developed in Japan. In addition to that, several robots are more integrated into our society, as the service and assistive robots. This type of applications will be presented in the next paragraphs.

I.2.1 Service Robots

According to the International Federation of Robotics (IFR), *"a service robot is a robot which operates semi or fully autonomously to perform services useful to the well being of human and equipment, excluding manufacturing operations"*. Indeed, service robots do a service that can be identified as a complete task with its actions, which are often more extended and complicated than those in industrial applications.

However, generally a service robot is understood as a robotic system with a certain level of autonomy in performing service operations for given tasks within a specified environment and interaction with human users, if any. The level of autonomy of a service robot can be considered as a function of the service task and is due to the level of supervision or interaction by a human operator or user. In some cases, full autonomy with even artificial intelligence is necessary for service goals, mainly when the service robot is expected to operate fully autonomously in unstructured environments.

In other tasks, because of the well structured configuration of the environment, the system's autonomy can be conveniently designed by just using proper sensors and suitable trajectory planning. Moreover, in other scenarios, because of the strong interaction with a human user, the autonomy is fully constrained for a proper interaction with a human user. Finally,



a - Bottle Manipulation

b - ASIMO serving coffee

c - Tokyo University's IRT domestic robot

d - Robot opening a drawer

Figure I.9 – Examples of service and assistive robots

in other specific cases, simple operations can be used to obtain a desired service operation [Cec11].

In recent decades, many robotic platforms have been developed, most of which include mobility, some type of autonomous navigation and also manipulation capabilities (Fig. I.9). It is rare, however, to find service robot platforms which incorporate vision system with object recognition, gesture recognition, scene modeling, comprehensive reasoning and planning components, and elaborate user interaction concepts.

I.2.2 Robot's Manipulation

Manipulation planning differs from standard motion planning in that the focus is not on the robot and its displacements but rather on the object(s) to be manipulated. Manipulation of rigid objects consists in changing their pose (position and orientation) while avoiding collisions, in the context of pick-and-place or assembly tasks [Jim12].

Moreover, object manipulation with humanoid robotic systems is an essentially different problem compared to solving the same task with an industrial robotic manipulator. The main difference lies in the accuracy of the hand-eye calibration. Furthermore, the performance of the manipulation tasks differs with respect to the robot's environment and object's architecture.

I.2.2.1 Manipulation environments

Within factories around the world; robots perform heroic feats of manipulation on a daily basis. They lift massive objects, move with blurring speed, and repeat complex performances with unerring precision. Yet outside of carefully controlled settings, even the most sophisticated robot would be unable to get a glass of water. To date, robots have been very successful at manipulation in controlled environments such as a factory. Outside of controlled environments, robots have only performed sophisticated manipulation tasks when operated by a human.

Within controlled environments, the world can be adapted to the capabilities of the robot. The robot typically needs to perform a few tasks using a few known objects, and people are

Figure I.10 – A work at AIST with the HRP-2 humanoid has combined high-level teleoperation with autonomous perception and control. [SSY+06]

usually banned from the area while the robot is in motion. If a robot needs to sense the world, the environment is made favorable to sensing by controlling factors such as the lighting and the placement of objects relative to a sensor. Moreover, since the objects and tasks are known in advance, perception can be specialized and model-based.

Furthermore, through teleoperation, even highly complex humanoid robots have performed a variety of challenging everyday manipulation tasks, such as grasping everyday objects, using a power drill, throwing away trash, and retrieving a drink from a refrigerator (Fig. I.10).

I.2.2.2 Manipulation approaches

To apply autonomous robot manipulation in human environments, several approaches are used [KETJ07]. Prior knowledge about the object and a special representation can be used to increase the robustness of the tracking system such as the commonly used CAD models (wire–frame models), view–and appearance–based representations [KC02a].

Different works have been carried out in teleoperation area, by controlling a robotic system to perform manipulation tasks at a distance using a multi-modal human system interface. Telerobotics combines a precise and fast device like a robot with the intelligence of a human being. These systems have been motivated by human safety issues in hazardous environments (e.g., nuclear or chemical plants), the high cost of reaching remote environments (e.g., space), scale (e.g., power amplification or position scaling in micro-manipulation or minimally invasive surgery), and many others.

However, continuous direct teleoperation of a complex robot can be tiring and difficult for an operator. The operator fatigue can lead to reduced precision during task execution [WMB03]. To overcome this problem, a recent study [DW12] works on the recognition of the teleoperator's intended motion, to autonomously continue the execution of recognized routine tasks. To do this, the robot learns offline a library of generalized activities from a training set of user demonstrations.

In fact, the framework of Learning from Demonstration or imitation learning [ACVB09] has been widely used to take advantage of humans and uses their guidance to make the problem tractable for the robot learner. When observing either good or bad examples, one can reduce the search for a possible solution, by either starting the search from the observed good solution (local optima), or conversely, by eliminating from the search space what is known as a bad solution.

I.2.2.3 Manipulation objects

Besides the classical manipulation of rigid objects, several service tasks deal with deformable and articulated objects. In fact, the robotic systems that address manipulation of deformable objects focus on force and position control of environments possessing compliant properties. In these systems, the robot manipulator is designed to control the deformation of an object [WR94]. It is usually incorporated directly into force calculation, and force feedback is utilized to ensure grasp stability [MP95]. Deformable objects also appear in the field of robotics in terms of grasping with soft fingers [SG96].

Fewer studies deal with articulated objects, in these cases, the manipulation task execution

process is divided into two stages: a reaching phase, where the hand of the robot must be moved towards the handle until the grasp is executed successfully, and an interaction phase, where the hand is in contact with the object and a particular mechanism must be activated [PMdPL07]. It is difficult to find a general method for automatically setting the task frame for all kind of tasks. However, authors applied this strategy for manipulation of everyday articulated objects with translational and revolute joints, such as doors, drawers, buttons, etc. The task frame was set naturally from the object structural model [PSdP08].

I.3 Main Control Approaches

Generally, manipulation tasks are specified in the task space in terms of a desired trajectory of the end-effector, while control actions are performed in the joint space to achieve the desired goals. This fact naturally leads to several kinds of general control methods, namely joint space control, operational space control (task space control) and sensor space control schemes.

I.3.1 Joint Space Control

The main goal of the joint space control is to design a feedback controller such that the joint coordinates track the desired motion as closely as possible. In this case, the control of robot manipulators is naturally achieved in the joint space since the control inputs are the joint torques.

Figure I.11 shows the basic outline of the joint space control methods. Firstly, the desired motion, which is described in terms of end-effector coordinates ($\mathbf{X}_d$), is converted to a corresponding joint trajectory ($\mathbf{q}_d$) using the inverse kinematics of the manipulator. Then the feedback controller determines the necessary joint torque (τ) to move the manipulator along the desired trajectory specified in joint coordinates starting from measurements of the current joint states ($\mathbf{q}$) [DWBS96, SS96].

Since it is always assumed that the desired task is given in terms of the time sequence of the joint motion, joint space control schemes are quite adequate in situations where manipulator tasks can be accurately preplanned and little or no online trajectory adjustments are necessary [AAH88, Yos90]. Typically, inverse kinematics is performed for some intermediate task points, and the joint trajectory is interpolated using the intermediate joint solutions. Although the command trajectory consists of straight-line motions in end-effector coordinates between interpolation points, the resulting joint motion consists of curvilinear segments that match the desired end-effector trajectory at the interpolation points.

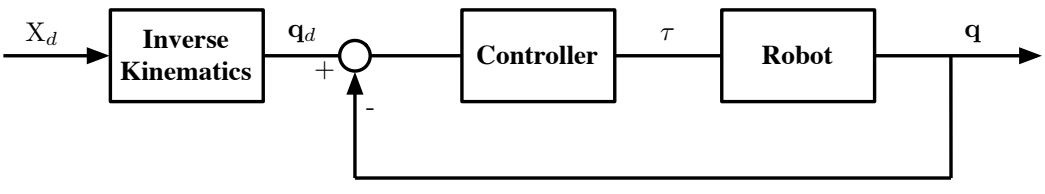


Figure I.11 – Basic schema of joint space control

Using only proprioceptive sensors leads the control law to be more sensitive to geometrical imprecision. This corresponded to a configuration where the robot consider that the task is realized (joints turned at the exact desired velocity), but actually it is not (the robot is not at the right place). This phenomenon is amplified for mobile robots, which can be subject to unmeasured displacements (sliding on the ground, drift due to wind). In practice, such control is possible for industrial repetitive tasks after a learning stage to ensure that the final position of the end-effector is the desired one for the task. The next paragraph deals with the operational space control which gives a first degree of autonomy to robots.

I.3.2 Operational Space Control

In more complicated and less certain environments, end-effector motion may be subject to online modifications in order to accommodate unexpected events or to respond to sensor inputs. In particular, it is essential when controlling the interaction between the manipulator and environment is of concern.

Since the desired task is often specified in the operational space and requires precise control of the end-effector motion, joint space control schemes are not suitable in these situations. This motivated a different approach, which can develop control schemes directly based on the dynamics expressed in the operational space [LWP80, Kha87].

The main goal of the operational space control is to design a feedback controller that allows execution of an end-effector motion that tracks the desired end-effector motion ($\mathbf{X}_d$) as closely as possible. For this case, Fig. I.12 shows a schematic diagram of the operational space control methods. There are several advantages to such an approach because operational space controllers employ a feedback loop that directly minimizes task errors. Inverse kinematics is not calculated explicitly since the control algorithm embeds the velocity-level forward kinematics, as shown in the scheme. In this case, motion between points can be a straight-line segment in the task space.

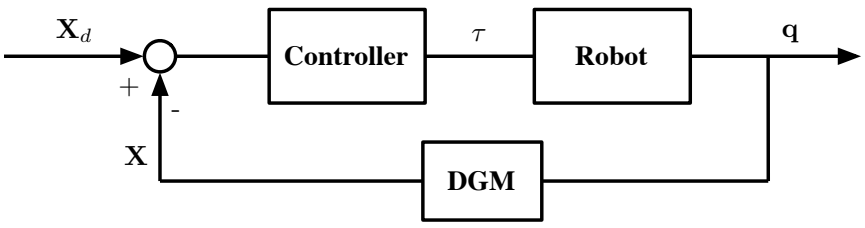


Figure I.12 – Basic schema of operational space control

Control approaches based on the definition of a task objective, such as the task function approach [SLBE91] or the operational space formulation [Kha87], have been introduced to simplify the control problem by working directly in a properly chosen task space. It also enables to address the control problem directly in the sensor space, which closes more tightly the control loop and improves the control law robustness and accuracy [CH06].

I.3.3 Sensor Space Control

To make control robust with respect to modeling errors, another control scheme can be used, where control is performed in a sensor space, which should be the image of the Cartesian space by a diffeomorphism (Fig. I.13).

Basically, sensor-based control allows to replace the direct geometric model in the feedback by a sensor measuring the feature vector $\mathbf{s}$ which is regulated to the desired value $\mathbf{s}_d$. If control is performed directly in the sensor space, one gets rid of almost all modeling errors. In fact, since they do not appear any more in the regulated error (end-effector pose is not estimated via a model), only the transient phase might be affected by them but not the convergence [WSN87].

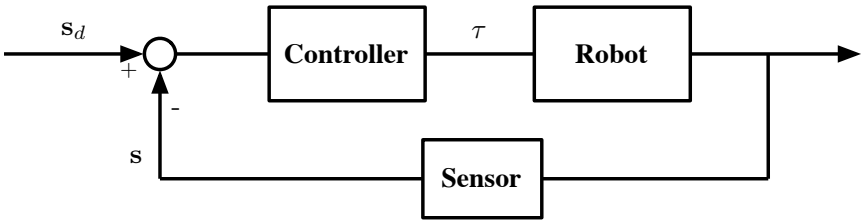


Figure I.13 – Basic schema of sensor space control

Sensor-based robot control overcomes many difficulties of uncertain models and unknown environments which limit the domain of application of current robots used without external sensory feedback. Both industrial arms and mobile robots require sensing capability to adapt to new tasks without explicit intervention or reprogramming [CKP+10].

Fig. I.14 shows an example of a sensor-based control task in which a robot arm acquires an object from the table using visual feedback control. A task-level command specifies manipulation of the object; however, the robot has not been preprogrammed with knowledge of the object position. In this sense, the task environment is "unstructured". A camera is attached to the robot arm and provides visual sensing capability. The acquired image must be processed by a computer vision system to identify the object and infer relationships between the spatial position of the object and the camera position. Such relative position information may be used to guide the robot to acquire the object from the table.

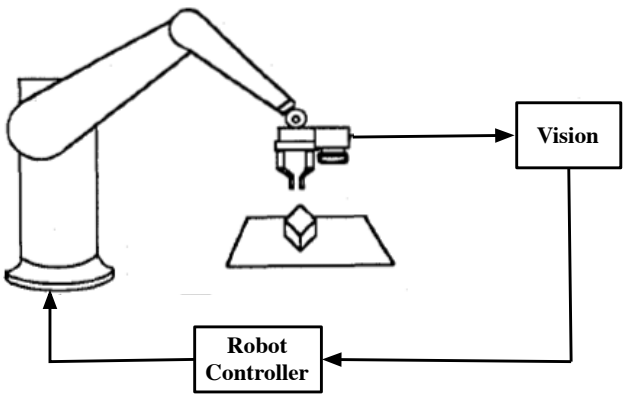


Figure I.14 – Robot acquiring object using vision feedback [WSN87]

I.4 Motivations and Objectives

Redundancy is a wide concept which is not limited to robotics. It can be literally defined by "*the state of being not or no longer needed or useful*", or by "*the use of data that could be omitted without loss of function*". Moreover, from an engineering perspective, it is "*the inclusion of extra components which are not strictly necessary for functioning*".

Therefore, all robotic systems, from multi-arm to humanoids, present several types of redundancies which may appear not only in the original mechanical structure of the robot, but also in the interaction between the robotic system and its surrounding objects. The architecture of these latter (e.g. deformable and articulated items) inherit a redundancy which also leads to different strategies in the definition of the applied task. Furthermore, the excessive use of external and embedded sensors results in a sensor redundancy. However, such redundancy can be useful when high reliability is necessary or when the sensors' workspace is very restricted.

One or several types of the mentioned redundancies can simultaneously exist when controlling robotic platforms. In fact, their presence depends on the used elements (robots, objects, sensors, ...), on the desired application that defines the applied tasks, and on the required level of precision and robustness of the robot's controller.

On the other hand, when applying a task on such redundant systems, there exist an infinite number of possible solutions. Thus, the choice among which should be based on certain criteria. This fact gives rise to the definition of new control laws to manage the possible internal motions of the system by applying additional tasks, specifying some constraints on the system's behavior, or optimizing (locally or globally) a desired performance.

Regarding constraints, two types were identified: first, the classical equality constraint that, for example, imposes a desired configuration of the system, second, the inequality or unilateral constraint that is used to limit on the controls (e.g. velocity and acceleration bounds in robot's joints), or to avoid collision with obstacles in the environment or self-collision between the robot's parts.

I.4.1 Motivations and Problems

While it is widely present in all robotic systems, regardless of their domain of application, few research studies identify all types of redundancies present in their systems, and fewer benefit from such redundancies in their control techniques. Nevertheless, the diversity in system redundancies leads to the development of several methods for redundancy resolution. These techniques allow for choosing the best solution between the infinite possible ones and managing the priority and interactions between the main task and the additional tasks/constraints.

However, since the work of Liegeois in [Lie77] most published approaches are based on local optimization with a redundancy resolution at the velocity level using orthogonal projection into the null space of the main task. In spite of the simplicity of this scheme, its disadvantage is its existence in the local nature of the optimization process, which can lead to unsatisfactory performance over long tasks. Besides, this technique uses a very restrictive stability condition that does not allow the system to completely profit from the available redundancy.

Recently, several approaches try to enhance the system's performance by enlarging the free space to apply additional tasks. However, the absence of a clear comparison between their performance and the classical one, in addition to the lack of studying these methods' behavior in several particular and critical cases, restricts their use to few applications.

Therefore, depending on the controlled system, the applied tasks, and the existing types of redundancy, the choice of the most relevant method for redundancy resolution should be discussed. In fact, explicit performance criteria should be defined and considered when making this choice. Moreover, when the robot cannot apply all the desired prioritized tasks/constraints, the system should remain stable and a recommended behavior should be defined, for example, the tasks and constraints should be fully or partially executed, or the system should be blocked and none of the tasks be applied.

I.4.2 Objectives

The goal of this thesis is to consider the redundancy as an advantage that we should benefit from its presence in the system, and not to consider it as a problem that we should avoid as classically done. In fact, we are interested in studying the redundant system's behavior when executing simultaneous tasks with a specific hierarchy and priority. Therefore, we carry out a comparison between the different redundancy resolution techniques to better characterize their advantages and drawbacks and determine the appropriate applications where they can be used. In addition, to control a general robotic system, a kinematic/dynamic approach is developed and validated by the application of several service and assistive tasks on HRP-2 and Nao humanoid robots.

In **Chapter II**, we address the definition and classification of the several types of redundancies present in a general robotic platform. Furthermore, the different control laws for redundancy resolution and task sequencing are presented in a unified notation with more details on the hierarchical control techniques.

After classifying the different techniques, we choose in **Chapter III**, the appropriate ones to achieve our objectives, and we compare them based on specific performance criteria. The comparison is conducted for different types of planar robots and for the 7-degrees-of-freedom LWR Kuka robot. Regarding the applied tasks, positioning tasks are executed on both types of robots as primary and secondary tasks. Also, equality and inequality constraints are studied: the first type can be considered, by definition, as equivalent to a generic task, and the second corresponds to a restriction or obstacle in the robot's workspace.

Subsequently, to achieve a deeper comprehension of task sequencing techniques for simultaneous execution of several tasks, we consider various case studies where the robotic system can or cannot execute all the desired tasks. In the latter case, the best robot's performance consists in maintaining the stability of the system, executing the main task and performing at the best the additional tasks/constraints in the given priority.

In light of this comparison, we can choose the applicable and optimal control solution to apply the desired tasks on redundant systems depending on the number of the system's and

the task's degrees of freedom, in addition to the type of applied tasks and constraints. Finally, the improvement of the existing techniques will also be addressed to overcome these methods' drawbacks and to resolve the encountered problems during simulation. Ultimately, the theoretical study and performance comparison of the different methods will lead to the definition of new generalized and simple methods for redundancy resolution.

Later on, after addressing the redundancy resolution techniques, we develop, in **Chapter IV**, the kinematic and dynamic multi control points approach which is a methodology to control a general redundant system. Instead of the classical joint space and operational space control approaches, this formalism relies on a sensor-based technique which involves studying and monitoring the robot/environment interaction on several points of the system.

When performing service and assistive tasks on redundant robots, especially humanoids, the same tasks are usually executed. Thus, we define the recurrent tasks in a generic form to maintain the adaptability of the presented control approach.

On the other hand, to perform a precise and robust sensor-based control, we use various types of vision systems that are already embedded in the robot or mounted in the system's environment. They are used to implement several domestic service and assistive tasks which are decomposed into multiple generic tasks. In addition, the previously presented redundancy resolution techniques are used to control the robotic system when performing these tasks while managing their relative priority.

Finally, this formalism is used in **Chapter V** to perform several applications in three platforms: first, on a multi-arm platform for meat cutting and muscles separation; second, by simulation, on the HRP-2 humanoid robot using OpenHRP simulator, then in real-time on the Nao humanoid robot. Several visual servoing techniques are used to track the robot and the environment's objects and to perform localization, navigation, object tracking, and manipulation tasks.

II

Kinematic Redundancy Resolution

Contents

Redundancy provides a robotic system with an increased level of dexterity that may be used to optimize many performance criteria and complete additional tasks. In order to achieve this, the existing redundancies should be first identified and classified, then an appropriate strategy should be developed to benefit from their presence in the robotic system to apply several tasks simultaneously while managing the priority and compatibility between them.

Due to the complexity of redundant robots, several techniques were developed to control such systems using a variety of control laws. Thus, the first goal in this chapter is to present the different methods for redundancy resolution in a unified notation to facilitate the comparison between them. Later on, the discussion on the advantages and drawbacks of each method will lead to a better comprehension of the methods used to exploit the redundant degrees of freedom in the solution of the inverse kinematics problem. Finally, this study will to leads to the development of new redundancy resolutions methods, to overcome the limitations of the existing methods.

In this chapter, based on the diversity in redundancy interpretations, the first section gives a classification of the different types of redundancies. It takes into account the architecture of the overall robotic system, including his interaction with the environment's items and the redundancy coming from the sensory tools. This section also discusses the advantages of using redundant robots over classical ones.

In the second section, several techniques for kinematic redundancy resolution are presented and discussed. In addition to that, these methods are illustrated with several example applications in various robotic domains where they are used to solve specific problems or enhance the system's performance. Furthermore, depending on the used control strategy the different methods are arranged in four main categories: partitioned, commutative, hybrid and hierarchical.

The last control type is studied in detail in the third section where the three principal projection operators (orthogonal, bidirectional and minimum norm solution) are constructed and theoretically compared. The extension of the presented techniques, from the simple two tasks execution to the case of multiple hierarchical tasks execution, is addressed in the fourth section where several task sequencing methods are discussed and applied to the presented methods.

Based on the study and discussion results, the consistent redundancy resolution methods will be compared, in the next chapter, by simulation on different types of robots and under several conditions. Various comparison and performance criteria will be also defined and used to show the benefits of each control law and the possible applications where it can be applied. New projection operators will be also defined to improve the performance of the existing ones and to overcome the encountered problems during simulation.

II.1 Definition and Classification of Redundancies

In the robotics literature, there exist few global classifications of robotic systems redundancies [Nak90]. One of them was elaborated in the technical note [CB97] where the redundancy concerning robotic manipulators was studied and only two categories were identified: sensor redundancy and mechanical redundancy.

In this section, we will present a more global classification that takes into account the overall robotic system including his interaction with the environment's items. In fact, redundancy is studied at three levels: the robotic system, the robot-environment interaction and the sensory tools.

II.1.1 Global Redundancy Classification

We consider:

— 'n' the degree of mobility of the robot
— 'm' the degree of mobility of the end effector
— 't' the dimension of the task
— 'N_a' the number of motorized articulations of the robot

Thus, we can identify the following cases:

Case of Non-Redundancy: $n = m$

In this case, the robot has the same degree of mobility as the degree of mobility of the end-effector. This is the case of **non-redundant** robot.

Note that when there is a reduction of the dimension 'm' in specific configurations ($n > m$), the robot is called in a degenerate or singular configuration.

Case of Kinematic Redundancy: $n > m$

A manipulator is intrinsically redundant when the dimension of the operational space is smaller than the dimension of the joint space. Therefore, when 'n' is designed to be greater than 'm', the device is **kinematically redundant**. In this case, the inverse geometric model has an infinite number of solutions. Furthermore, in such situations, the

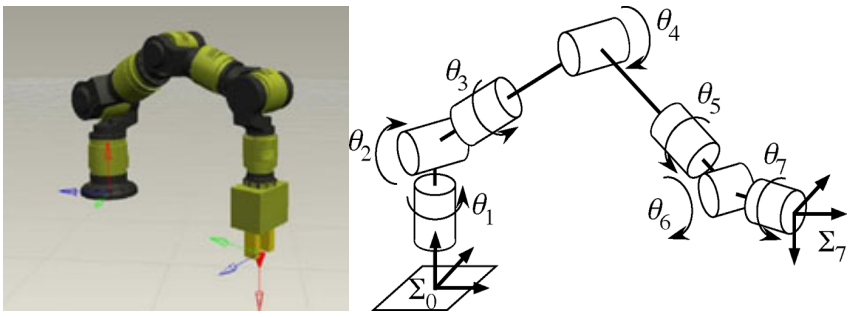


Figure II.1 – Example of kinematically redundant manipulator

self shape of a device can be varied without changing the end-effector configuration since the joints do not produce independent motion in end-effector space. This motion is also known as null-space (zero space) motion, self-motion or internal motion.

An example of kinematic redundancy is found, for example, in manipulators of 7 DOF (Fig. II.1), and in the four link planar manipulator, shown in Fig. II.2. This type of devices clearly fulfils the given definition since even the simplest tasks cannot be completed without selecting a solution out of an infinite number of solutions to control the internal configuration of links to reach the desired location.

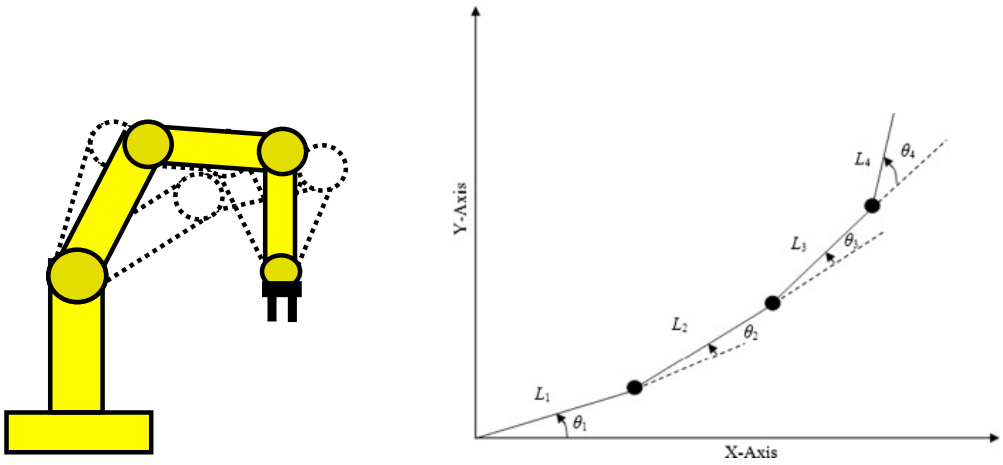


Figure II.2 – Examples of kinematically redundant planar robot

Case of Task Redundancy: $n > t$

When the dimension of the task 't' is smaller than the robot's degree of mobility, then this describes **task redundancy** (or functional redundancy).

The redundancy in the interaction between the robot and his environment appears especially in the definition of the applied task that can be performed in different strategies. Thus, task redundancy is not defined only by the system's architecture, nor by the object's form alone, it is rather characterized by the interaction between them; thus, it changes according to user parameters and the desired task to be executed.

An example of such type of redundancy appears when dealing with object manipulation; in this case it is known as grasp redundancy [PSdP11]. It is generally exploited in order to further increase the number of redundant DOF and thus increase the space of secondary motion, which allows to keep a comfortable arm posture while performing the main grasping task.

Grasp redundancy is defined in the operational space, as the number of DOF's which do not need to be controlled for a particular grasp. For example, when grasping a handle, a rotation around the handle axis can be considered as grasp-redundant if the grasp and

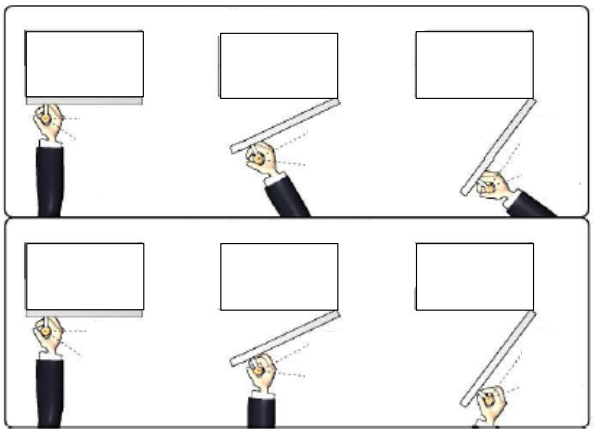


Figure II.3 – Example of door opening with and without consideration of grasp redundancy.

In the top row, the task redundancy was not considered, thus a full-constrained grasp is performed (the relative orientation between the hand and the handle remains fixed during the interaction task).

In the bottom row, the grasp redundancy is exploited; thus, the grasp and the hand frames are not rigidly attached during the task execution.

handle geometry allow it (as it was shown in Fig. II.3).

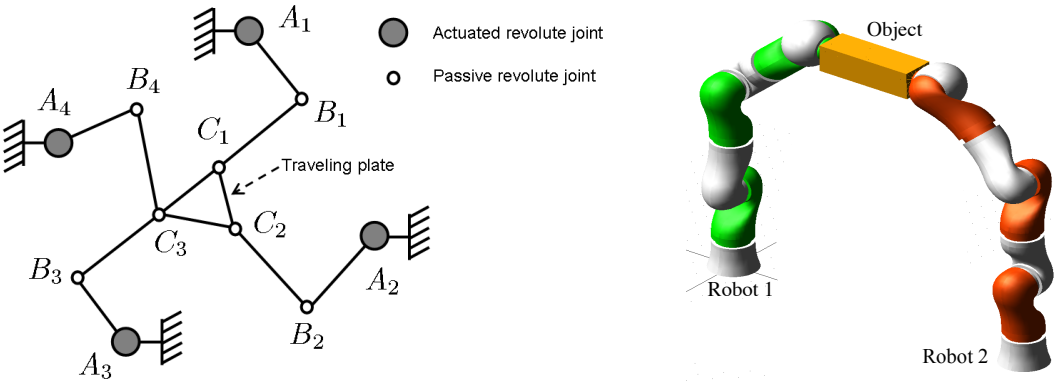
Case of Actuation Redundancy: $N_a > n$

When the number of motorized articulations is greater than the degree of mobility of the robot, thus an **actuation redundancy** is present.

For example, a parallel manipulator with actuation redundancy is shown in Fig. II.4a; it has 3 DOF and an end-effector spatiality equal to 3. However, four actuated revolute joints are used; thus, this mechanism with actuation redundancy is "over-measured".

Furthermore, actuator redundancy occurs when two or more manipulators grasp a common object, the manipulators and the object form a closed kinematic chain in such the same way as a parallel manipulator [LCK12]. Although unlike parallel mechanisms each arm is composed of a serial manipulator, thus fully actuated. Therefore, the system is composed of more actuators than the object DOF (Fig. II.4b).

Case of Hyper Redundancy: $n >> m$



a - Parallel manipulator

b - Two robots grasping an object

Figure II.4 – Examples of actuator redundancy

In addition to the above definitions, there exist several devices that are categorized as highly or hyper redundant [MSK96]. Such devices have a joint space dimension that is much greater than the dimension of the end-effector space, i.e. '$n >> m$'.

Case of Sensor Redundancy:

There is no extensive and clear definition of **sensor redundancy** which is applicable for all type of structures and sensors. In general, such redundancy is related to the presence of more sensing or measuring elements than theoretically necessary, it is usually used when high reliability is necessary or due to limitations in the sensor's workspace [CKP+10]. Furthermore, it is used in case of parallel robots where some passive joints are motorized to find out the configuration of the robot.

When the manipulator has such redundancy, the extra sensor(s) information can be used to compute the end-effector coordinates more accurately under an environment with noise and some other uncertainties. Moreover, such redundancy is necessary when some locations are not reachable by the sensor due to the kinematic motion model of the robot or due to obstacles in the sensor's area.

II.1.2 Advantages of Redundant Systems

While most non-redundant manipulators possess enough DOF to perform their main task(s), i.e., position and/or orientation tracking, it is known that their limited manipulability results in a reduction in the workspace due to mechanical limits on joint articulation and presence of obstacles in the workspace.

On the other hand, the increased dexterity characterizing kinematically redundant manipulators may allow not only to avoid singularities [Chi88], joint limits [Ser91], and workspace obstacles [CSG89], but also to include dynamic measures of performance by defining kinematic functions as the configuration-dependent terms in the manipulator dynamic model, e.g., impact force [LPB95], inertia control [SC90]. Furthermore, it can be used to minimize torque/energy over a given task, ultimately meaning that the robotic manipulator can achieve a higher degree of autonomy.

Moreover, the principal use of actuator redundancy [HG05, SOH12] is to manage the internal forces experienced by the object, and if desired maintain a constant grasp on the object or minimize driving torque. Thus, actuator redundancy is also useful for increasing the factor of safety for a device since a failure of an actuator will not leave the system uncontrollable.

Finally, sensor redundancy enables sensor fusion that reduces measurement error and allows a wider and more accurate range of information by combining data from different types of sensors. Furthermore, it enables an increased reliability and weaknesses compensation, in addition to detection and handling of sensors failure. Several typical applications can benefit from sensor redundancy such as parallel robots [MKP+02], modular space robots [TKD11], and soft robots [ASEG+08].

II.2 Redundancy Resolution and Task Sequencing

In this section, we will investigate the resolution techniques of the kinematic redundancy. Other types of redundancies will be approached in next chapters when comparing these methods by simulation and when applying service tasks to humanoid robots.

In fact, we are interested in taking advantage of the manipulator's kinematic redundancy to apply, besides the main task, several additional secondary tasks or constraints. Therefore, first of all, we give the notation which is used for representing the geometric and kinematic models of the robot, then we present and classify the different techniques that are used in the literature to solve the kinematic redundancy. The corresponding control laws are given in a unified notation to make easier the comparison between the different methods.

Geometric Model
Using the following notations: $\mathbf{q}$ the vector of n joint positions defining the configuration of the manipulator ($\mathbf{q} \in \mathcal{R}^n$), and $\mathbf{X}$ the m-dimensional vector of end-effector pose ($\mathbf{X} \in \mathcal{R}^m$). The joint and task coordinates are related by the following forward geometric equations:

$$\mathbf{X} = DGM(\mathbf{q}) \tag{II.1}$$

Kinematic Model
Differentiation with respect to time of the geometric model equations (II.1) yields to a set of equations of the form:

$$\dot{\mathbf{X}} = \mathbf{J}(\mathbf{q})\, \dot{\mathbf{q}} \tag{II.2}$$

where $\dot{\mathbf{X}}$ is the linear/angular velocity of the end-effector, and $\mathbf{J}(\mathbf{q})$ is a $m \times n$ matrix whose elements are, in general, nonlinear functions of $q_1, ..., q_n$. $\mathbf{J}(\mathbf{q})$ is called the Jacobian matrix of this algebraic system and is expressed relative to the same reference frame as the spatial velocity $\dot{\mathbf{X}}$. The Jacobian matrix $\mathbf{J}(\mathbf{q})$ is always dependent of joint angles, however, in next paragraphs $\mathbf{q}$ will be omitted and the Jacobian will be simply noted $\mathbf{J}$.

In order to accomplish a task, a proper joint motion must be commanded to the manipulator; therefore, it is necessary to derive mathematical relations which allow computing joint-space variables corresponding to the assigned task-space variables. In fact, for a kinematically redundant manipulator, the inverse kinematics problem admits an infinite number of solutions, so that a criterion to select one of them is needed.

As seen in the previous section, kinematic redundancy can be used to apply, in addition to the main task, supplementary tasks or constraints. Thus, a formalism should be found to manage the priority and the compatibility between the executed tasks. In the rest of this section, we will present several task sequencing approaches with their corresponding control laws.

Four major techniques are usually used for task sequencing. The first one consists in assigning to each task a certain number of DOF of the system, this is called partitioned control. A second approach is the commutative control: one task at a time is applied and criteria are defined to switch from one task to another. The third approach consists in applying a compromise between the different tasks by defining fixed or variable weights for each one. The last method involves a hierarchy between tasks; In this case, some tasks have priority over others.

The difficulty is then to allow secondary tasks to be achieved under the constraint of the highest priority task.

In next paragraphs, we consider the generic elementary robotic task which is defined by three elements:

- A vector $\mathbf{e_i}$ that corresponds to the error between a given signal and its desired value.
- A Jacobian $\mathbf{J_i}$ that binds the error and the vector of joint parameters $\mathbf{q}$, according to $\mathbf{J_i} = \frac{\partial \mathbf{X}}{\partial \mathbf{q}}$ (the corresponding Moore-Penrose pseudo-inverse is noted $\mathbf{J}_i^+$).
- A reference evolution of the task function $\dot{\mathbf{e}}_\mathbf{i}$.

In fact, the task function vector $\mathbf{e_i}$ characterizes a generic robotic task to be performed. For example, it can represent a desired trajectory to be followed in the joint or operational space, or a regulation of a distance between the robot's end effector and an object in the environment.

II.2.1 Partitioned Control

The partitioned control schema [Bil87, OA99] is used when applying independent tasks, thus each task controls separately its own DOF. Therefore, it is not possible to couple between tasks in the articular space.

This method was used also to merge data coming from different independent sensors especially in hybrid vision/force control [HIA98]. The general control law can be written:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\dot{\mathbf{e}}_\mathbf{1} + \mathbf{J_2^+}\dot{\mathbf{e}}_\mathbf{2} + ... \tag{II.3}$$

II.2.2 Commutative Control

Using this approach, tasks are applied one by one using the control law below:

$$\dot{\mathbf{q}} = \begin{cases} \mathbf{J_1^+}\dot{\mathbf{e}}_\mathbf{1} & \text{if} \quad C_1 \\ \mathbf{J_2^+}\dot{\mathbf{e}}_\mathbf{2} & \text{if} \quad C_2 \end{cases} \tag{II.4}$$

where C_1 and C_2 are the conditions that allow to move from one scheme to another, typically with hysteresis behavior.

For example, in [GH07] authors defined a visual servoing with 2D data for the first task, and 3D data for the second one. To minimize the error simultaneously in joint and task spaces, the robot apply by turns the image based and position based visual servoing depending on whether the error on 2D or 3D data is greater.

Furthermore, in [CSC11], visual servoing is used to control a wheeled vehicle equipped with an actuated pinhole camera and a forward-looking range scanner. During this application, the main task of the mobile robot is to navigate by following a visual path represented by key images. Moreover, a secondary task is applied to avoid collision with the ground obstacles and to avoid occlusion by applying the camera visibility task. Therefore, for the secondary task, a commutative compact controller is used for the transition between safe and dangerous contexts. In fact, the secondary task is the obstacle avoidance if an obstacle is present, otherwise it is the camera visibility task.

Although the commutative approaches are attractive because of their simplicity, the conditions defining the transition from one task to another are not always simple to determine. Moreover, if we want to achieve all tasks at the same time, other approaches should be used.

II.2.3 Hybrid Control

Many techniques use this method which consists of introducing additional constraints or tasks into the inverse differential problem.

In [COW08], authors defined the extended/augmented Jacobian methods and studied their algorithmic singularities. Thus, the task-space augmentation introduces constraint tasks to be fulfilled along with the main task, to identify a single solution among the infinite compatible with the main task.

If we consider the vector $\mathbf{X}_c$ which describes the additional $(n-t)$ tasks/constraints to be fulfilled besides the t-dimensional main task $\mathbf{X}$. The relation between the joint-space coordinate vector and the constraint-task vector can be considered as a direct geometric equation:

$$\mathbf{X}_c = \mathbf{h_c}(\mathbf{q})$$

where $\mathbf{h}_c$ is a continuous nonlinear vector function.

Accordingly, it is useful to consider the mapping $\dot{\mathbf{X}}_\mathbf{c} = \mathbf{J_c}\,\dot{\mathbf{q}}$ that can be obtained by differentiating the above equation. $\dot{\mathbf{X}}_\mathbf{c}$ is the constraint-task velocity vector, and $\mathbf{J_c}(\mathbf{q}) = \frac{\partial \mathbf{h_c}(\mathbf{q})}{\partial \mathbf{q}}$ is the $(n-t)\times n$ constraint-task Jacobian matrix.

At this point, an extended-task vector can be defined by stacking the main task vector with the constraint-task vector as:

$$\mathbf{X}_{ext} = \begin{bmatrix} \mathbf{X} \\ \mathbf{X}_c \end{bmatrix} = \begin{bmatrix} DGM(\mathbf{q}) \\ \mathbf{h}_c(\mathbf{q}) \end{bmatrix}$$

According to this definition, finding a joint configuration $\mathbf{q}$ that results in some desired value for $\mathbf{X}_{ext}$ means satisfying both the main and the secondary task(s) at the same time. A solution to this problem can be found at the differential level by inverting the mapping:

$$\dot{\mathbf{X}}_\mathbf{ext} = \begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{X}}_\mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}_c \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J_{ext}}\,\dot{\mathbf{q}} \tag{II.5}$$

in which the matrix $\mathbf{J}_{ext} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J}_c \end{bmatrix}$ is termed the extended Jacobian.

In practice, hybrid control uses this method to resolve the redundancy coming from combining several sensors data for example. They use the extended Jacobian and a weighting matrices to manage the priority between tasks [Nak90, KC11b]. However, when using these approaches, the compatibility between the main task and the additional constraints should be respected to avoid getting into singular configuration of the extended Jacobian. Moreover, these methods are not sufficiently reactive when dealing with tasks and constraints of variable dimensions and priorities.

Classification of Hybrid Control approaches:

Several approaches of hybrid control have been elaborated to combine and manage the priority between the defined tasks/constraints. These methods can be sorted into three categories:

II.2.3.1 Hybrid control without weighting

The first approach is based only on the extended Jacobian inverse [Nak90, DLOG06]. As defined in equation (II.5), the extended Jacobian $\mathbf{J}_{ext}$ includes data of all tasks, thus all components of $\dot{\mathbf{e}}$ converges at the same rate with the following control law:

$$\dot{\mathbf{q}} = \mathbf{J}_{\mathbf{ext}}^{+} \dot{\mathbf{e}} \tag{II.6}$$

The extended Jacobian is used in hybrid visual servoing scheme as in [MLS+00] where 2D and 3D data are used in the same control law. However, this approach is insufficient to take constraints into account. Furthermore, regrouping of variables of various natures (distances, forces, angles, pixels,...) in the same error vector may cause some conditioning problems. Thus using a generalized inverse with a weighting matrix can be helpful.

II.2.3.2 Hybrid control with weighting in the joint space

This method consists on combining tasks in the joint space using weights w_i to manage the contribution of each one in the final control law. In fact, it is a kind of partitioned control with independent weights in the joint space:

$$\dot{\mathbf{q}} = w_1 \, \dot{\mathbf{q}}_{\mathbf{1}} + w_2 \, \dot{\mathbf{q}}_{\mathbf{2}} + ... \tag{II.7}$$

The following control law, which consists of a diagonal weighting matrix $\mathbf{W}$, was also used to perform a weighting in the joint space. For example, in [CD95] it was applied to avoid joints limit for a redundant manipulator.

$$\dot{\mathbf{q}} = \mathbf{W}(\mathbf{J}_{\mathbf{ext}}\mathbf{W})^{+}\dot{\mathbf{e}} \tag{II.8}$$

More recently, [XZW10] used this approach and added virtual constraints limits, but problems of discontinuity occurs because of the variable rank of the inverted matrix. In fact, it is a common problem which especially occurs with tasks of varying dimension as discussed in [MRC09].

II.2.3.3 Hybrid control with weighting in the task space

The use of sensor based control approach makes more intuitive the use of the weighting in the task space instead of the joint space. Thus the following control law uses an extended Jacobian with a weighting matrix $\mathbf{H}$ on the task space as following:

$$\dot{\mathbf{q}} = (\mathbf{H}\,\mathbf{J}_{\mathbf{ext}})^{+}\,\mathbf{H}\,\dot{\mathbf{e}} \tag{II.9}$$

Various methods used this technique, they can be classified with respect to the weights type and to their method of calculation:

— **Dynamic**:
In [Kha87], **H** is calculated from the inertial matrix to minimize the kinetic energy of the robot.

— **Experimental**:
[WES87] and [PNK92] calculated **H** using the Linear Quadratic control (LQ) to minimize some criterion.

— **Constant**:
[SG07] uses constant weights h_i of **H** for each task, but some constraints could be violated if the associated weights are not sufficiently high.

— **Variable**:
Other approaches used variable weights h_i to control the priority between tasks/constraints. In this case, multiple variation methods of h_i arise:

- **Activation Matrix**:
  Weights could be $h_i = 0$ or $h_i = 1$.
- **Active weights**:
  In [CMC06] weights reflects the level of confidence in the task ($0 < h_i < 1$).
- **Relative weights**:
  In [KC11a] weights h_i are found such that the system is stable ($0 < h_i < \infty$).

Moreover, a hybrid method given by [HJ06] is left out of this classification. In this case, the weighting matrix **H** is not considered in the Jacobian: $\dot{\mathbf{q}} = (\mathbf{J_{ext}})^{+}\mathbf{H}\,\dot{\mathbf{e}}^{*}$, however the performances of this method are less satisfactory.

We present below one of the hybrid control methods with variable weighting matrix in the task space: when the system approaches to constraint violation, the weights values changes to promote the constraint over the main task execution. This method will be used in the next chapter when comparing the redundancy resolution methods.

This method uses a generic weighting as follows:

$$h_i = h_i^t + h_i^c \tag{II.10}$$

where h_i^t is the general weighting of a task given by:

$$\forall i\,, \quad h_i^t = \begin{cases} 1 & \text{if} \quad T_i \text{ corresponds to a task} \\ 0 & \text{if} \quad T_i \text{ corresponds to a constraint} \end{cases}$$

For the h_i^c part, the goal is to maintain the constraint value in an internal $\mathcal{I} = [s_{min}, s_{max}]$, thus another interval can be defined as $\mathcal{I}_0 = [s^-, s^+]$ with:

$$\begin{cases} s^- = s_{min} + \rho\,(s_{max} - s_{min}) \\ s^+ = s_{max} - \rho\,(s_{max} - s_{min}) \end{cases}$$

Figure II.5 – Variation of h_i^c value with respect to feature's value **s**

where $\rho \in [0, 0.05]$ is an adjustable parameter.

Thus the constraint is not considered ($h_i^c = 0$) in the interval $\mathcal{I}_0$ and increases progressively (see Fig. II.5) when approaching $\mathcal{I}$ limits ($0 < h_i^c < inf$) :

$$h_i^c(s) = \begin{cases} \frac{s-s^+}{s_{max}-s} & \text{if } s > s^+ \\ \frac{s^--s}{s-s_{min}} & \text{if } s < s^- \\ 0 & \text{otherwise} \end{cases} \tag{II.11}$$

II.2.4 Hierarchical Control

To impose a desired hierarchy between tasks, we should find possible motions that can be realized without disturbing the execution of the main task. Several classical approaches project the secondary task(s) into the null space of the Jacobian of the main task. This method will execute at best the other constraints, without disturbing the execution of the main task.

In order to solve the linear system of equations in the joint rates obtained by decomposing (II.2) into its component equations when $\dot{\mathbf{X}}$ is known, it is necessary to invert the Jacobian matrix using the generalized pseudo-inverse $\mathbf{J}^{\#}$.

The general solution of the first order differential kinematics equation can be written as:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\,\dot{\mathbf{X}} + \mathbf{P}\,\mathbf{z} \tag{II.12}$$

where $\mathbf{P}$ represents a generalized projection matrix, and $\mathbf{z}$ an arbitrary joint-space velocity.

When the Moore–Penrose pseudo-inverse ($\mathbf{J}^{\#} = \mathbf{J}^{+}$) is used, the second part of the solution (II.12) above is a null-space velocity. Therefore, this solution provides all least-squares solutions to the main task constraint, i.e., it minimizes the norm $\left\|\dot{\mathbf{X}} - \mathbf{J}\dot{\mathbf{q}}\right\|$.

By acting on $\mathbf{z}$, one can still obtain different joint velocities that give the same task velocity; therefore, the solution above is typically used in the context of redundancy resolution.

By setting $\mathbf{z} = \mathbf{0}$, we obtain the particular solution: $\dot{\mathbf{q}} = \mathbf{J}^{+}\,\dot{\mathbf{X}}$, which provides the least-squares solution with minimum norm, and is known as the pseudo-inverse solution. In terms of the inverse differential kinematics problem, the least-squares property quantifies the accuracy

of the extended task realization, while the minimum norm property may be relevant for the feasibility of the joint-space velocities.

Most of the early methods used the orthogonal projection defined by [Lie77], and determined secondary tasks via (local or global) optimization of defined performance criterion. Although global optimization is generally more successful, it requires optimization over the entire range of control and is thus not suitable for real-time control. On the other hand, local optimization needs less computation, but can result in a poorer performance from the global viewpoint. Nevertheless, local optimization remains the only choice for sensory guided robots, where the task path is generally not predictable.

Regarding the control input, most published approaches are based on local optimization techniques with a redundancy resolution at the velocity level using the Gradient Projection Method (GPM) as in [Lie77, Nak90, CCSS91, SLBE91]. In fact, it is used to calculate in real-time the possible secondary tasks with respect to the actual main task by projecting the gradient of a desired performance criteria. The main advantage of this redundancy resolution scheme is its simplicity, but its disadvantage is to be found in the local nature of the optimization process, which can lead to unsatisfactory performance over long tasks.

Later on, [Tay79] used the general solution of the inverse differential kinematics problem, and introduced the dampest-least square technique (DLS) including the selection of its damping factor. Additionally, more recently, [MDC+09] introduced two algorithms of a heuristic iterative method to control hyper-redundant robotic manipulators, a comparison between these methods and the pseudo-inverse Jacobian method was also evaluated.

However, instead of acting on the secondary task definition, early studies tried to change the used projection operator to get supplementary free space which is usually used to enhance robot's performance and apply additional tasks.

For example, in [MSEK09], the directional redundancy for robot control was presented and applied in continuous and discrete approaches. An application to visual servoing with the integration of different constraints laws was also applied to show the efficiency and robustness of this method, which was used in the "Stack of Tasks" formalism [MC09] to manage the execution of several tasks simultaneously. Later on, same authors presented a method to generalize the hierarchy-based control schemes to account unilateral constraints at any priority level [MKK09].

Furthermore, in [MC10], a new projection operator for the redundancy framework is proposed, it is based on a task function defined as the norm of the usual error. This projection operator allows performing secondary tasks even when the main task is full rank.

In the next sections, these projection operators will be detailed in a unified notation and compared to the classical orthogonal one.

II.3 Secondary Tasks Projection Methods

In this section, we present the different approaches for redundancy resolution which consists of projecting secondary tasks/constraints into the null space of the main task. The general form of robot's motion was given by (II.12).

II.3.1 Orthogonal Projection Method

The classical approach [Lie77] tries to ensure the execution of the main task $\mathbf{e}$ while considering a secondary task $\mathbf{z}$. It leads to the following control scheme:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}} + (\mathbf{I_n} - \mathbf{J}^{+}\mathbf{J})\,\mathbf{z} \tag{II.13}$$

where $\mathbf{P_e} = (\mathbf{I_n} - \mathbf{J}^{+}\mathbf{J})$ is the orthogonal projection operator into the null space of $\mathbf{J}$ so that $\mathbf{z}$ is realized at best under the constraint that it does not perturb the regulation of $\mathbf{e}$.

As detailed in the previous section, several methods use the classical approach to project any vector representing the desired motion in the space of the secondary task onto the null space of the Jacobian of the main task to modify the behavior of the system, but not the convergence of the main task.

Fig. II.6 represents this method in case of 1-dimensional main task. The null space of the main task $\mathbf{e}$ is the red line $Ker(\mathbf{J})$. In fact, the orthogonal projection approach projects all secondary tasks $\mathbf{z}_i$ on this line: $\mathbf{P_e z_i}$ $(i = 1, 2, 3)$ is the orthogonal projection of the secondary task $\mathbf{z}_i$ onto the null space $Ker(\mathbf{J})$ of the main task $\mathbf{e}$.

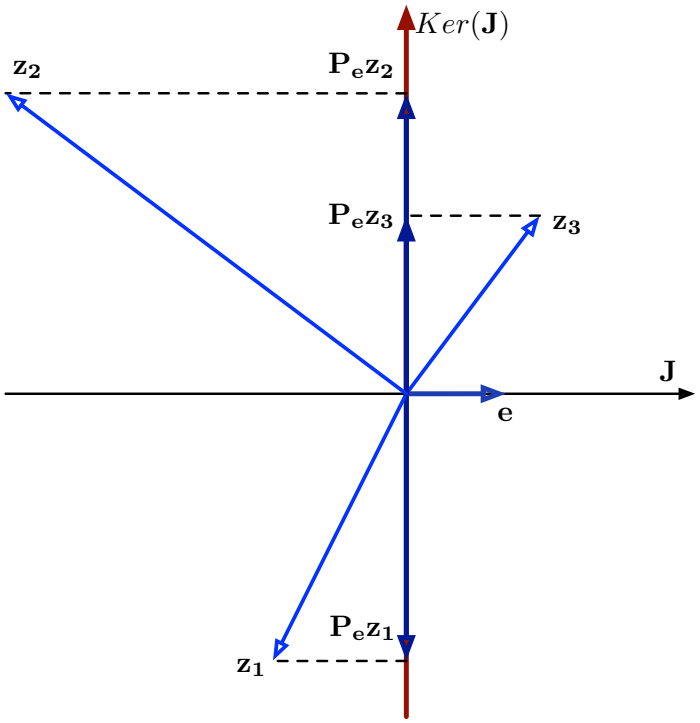


Figure II.6 – Example of orthogonal projection of simple tasks using the control law (II.13)

This method requires that the main task does not constrain all the robot's DOF. In fact, in that case, the main task Jacobian becomes of full rank and no redundancy space is left for projecting any constraint. Thus, this projection operator may be too constraining, which is the main limitation of the classical gradient projection method.

Stability proof

In case of the classical exponential error decrease $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, we can prove the stability of the system using the classical Lyapunov function which is based on the error norm $\|\mathbf{e}\|$:

$$\mathcal{L}(\mathbf{e}) = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}\mathbf{e}^\top\mathbf{e} \tag{II.14}$$

Thus, the derivative of this function is given by:

$$\dot{\mathcal{L}}(\mathbf{e}) = \frac{\partial\mathcal{L}}{\partial\mathbf{e}}\dot{\mathbf{e}} = \mathbf{e}^\top\dot{\mathbf{e}} = -\lambda\|\mathbf{e}\|^2 < 0 \tag{II.15}$$

Furthermore, it is easy to check that the addition of the secondary term does not modify the stability of the control law. In fact, $\dot{\mathcal{L}}(\mathbf{e})$ does not depend of the secondary term $\mathbf{z}$:

$$\dot{\mathcal{L}}_{(\mathbf{P_e})} = \frac{\partial\mathcal{L}}{\partial\mathbf{e}}\dot{\mathbf{e}} = \frac{\partial\mathcal{L}}{\partial\mathbf{e}}\mathbf{J}\dot{\mathbf{q}} = \frac{\partial\mathcal{L}}{\partial\mathbf{e}}\left(\mathbf{J}\mathbf{J}^+\dot{\mathbf{e}} + \mathbf{J}\mathbf{P_e}\mathbf{z}\right) = \frac{\partial\mathcal{L}}{\partial\mathbf{e}}\dot{\mathbf{e}} \tag{II.16}$$

The derivative of the Lyapunov function when considering the secondary task $\mathbf{z}$ in (II.16) gives the same result as when considering only the main task (II.15).

II.3.1.1 Damped Least-Square Method

At first glance, the solution given in (II.13) is quite attractive since the pseudo-inverse has the least squares property that generates the minimum norm joint velocities. However, [BHB84] proved that kinematic singularities are not avoided in any practical sense, since joint velocities are minimized only instantaneously and then can become arbitrarily large near singular configurations.

In order to overcome this drawback, [Wam86] and [NH86] independently proposed the use of $\mathbf{J}^{+\lambda} = \mathbf{J}^\top(\mathbf{J}\,\mathbf{J}^\top + \lambda^2\,\mathbf{I_n})^{-1}$ which is the damped least-square inverse of the Jacobian matrix corresponding to a modified Jacobian that is nonsingular in the whole workspace. This solution minimize $\left\|\dot{\mathbf{X}} - \mathbf{J}\dot{\mathbf{q}}\right\|^2 + \lambda^2\|\dot{\mathbf{q}}\|^2$ where $\lambda \in \mathcal{R}$ is a non-zero damping constant. Thus, the general control law will be:

$$\dot{\mathbf{q}} = \mathbf{J}^{+\lambda}\dot{\mathbf{e}} + \left(\mathbf{I_n} - \mathbf{J}^{+\lambda}\mathbf{J}\right)\mathbf{z} \tag{II.17}$$

Under this control, one obtains only an approximate inverse kinematic solution, and the problem becomes the selection of suitable values for the damping factor λ which sets the weight of the minimum norm solution. High values of λ give a good behavior, but reduce the accuracy in the neighborhood of singular points, thus λ=0 is considered faraway from singular points. Furthermore, it can be recognized that the appropriate choice of λ depends on the minimum singular value of the matrix $\mathbf{J}$ which is a measure of proximity to singularities.

[MK88] presented a technique to determine a good estimate of the minimum singular value to set λ; a refinement of the technique is also proposed which performs selective filtering only in the direction of the singular components for a given task trajectory.

Using the singular value decomposition, it can be shown that the damped least squares method tends to act similarly to the pseudo-inverse method away from singularities (when λ=0) and effectively smooths out the performance of this method in the neighborhood of singularities.

II.3.1.2 Transpose-Based Method

If a computationally cheaper solution is desired, one may advise the use of Jacobian transpose method which was used for inverse kinematics by [BDMS84, WE84]. The basic idea is to use the transpose of the Jacobian ($\mathbf{J}^\top$) instead of the pseudo-inverse ($\mathbf{J}^+$). However, the transpose of the Jacobian is not the same as the inverse; in fact, it is possible to justify the use of the transpose in terms of virtual forces.

$$\dot{\mathbf{q}} = \mathbf{J}^\top \dot{\mathbf{e}} \tag{II.18}$$

It can be shown, via a simple Lyapunov argument, that limited tracking errors and null steady-state errors are guaranteed [SY87]. An intrinsic advantage of the solution (II.18) is that it may avoid the typical numerical instabilities which occur at kinematic singularities, since no pseudo-inverse of the Jacobian matrix is required [SS88].

II.3.2 Bidirectional Non-Linear Projection Method

The general idea of the bidirectional non-linear projection is to enable the motions produced by the secondary control law that help the main task to be completed faster. The main advantage is to enhance the performance of the secondary task by enlarging the number of available DOF [MC09].

In addition, authors proposed a general solution (directional redundancy) that only imposes to the secondary control law not to increase the error of the main task (to preserve the system's stability), that is when the secondary task goes in the same direction as the main task.

A non-linear projection operator $\mathbf{P_z}$ is proposed to enlarge the free space on which the secondary task is projected:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \dot{\mathbf{e}} + \mathbf{P_z}\,\mathbf{z} \tag{II.19}$$

Using the same stability proof with the classical Lyapunov function, we find:

$$\dot{\mathcal{L}}_{(\mathbf{P_z})} = \frac{\partial \mathcal{L}}{\partial \mathbf{e}} \mathbf{J}\dot{\mathbf{q}} = \frac{\partial \mathcal{L}}{\partial \mathbf{e}} (\mathbf{J}\mathbf{J}^+ \dot{\mathbf{e}} + \mathbf{J}\dot{\mathbf{q}}_\mathbf{2}) = \dot{\mathcal{L}}_{(\mathbf{P_e})} + \frac{\partial \mathcal{L}}{\partial \mathbf{e}} \mathbf{J}\dot{\mathbf{q}}_\mathbf{2} \tag{II.20}$$

The directional projection operator is built such that $\dot{\mathbf{q}}_\mathbf{2} = \mathbf{P_z}\,\mathbf{z}$ respects the latter condition (of enlarging the free space for the secondary task projection) which can be represented by:

$$\nabla \mathcal{L}^\top \, \mathbf{J} \, \dot{\mathbf{q}}_2 \;\leq\; \mathbf{0} \tag{II.21}$$

where $\nabla \mathcal{L} = (\frac{\partial \mathcal{L}}{\partial \mathbf{e}})^\top$ and $\mathcal{L}$ is the Lyapunov function associated to $\dot{\mathbf{q}} = \mathbf{J}^+ \dot{\mathbf{e}}$.

By considering the singular values decomposition (SVD) of $\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U}$ is a basis of the task function space, $\mathbf{V}$ is a basis of the joint space, and $\mathbf{\Sigma} = [\mathbf{\Delta}_\sigma \ \mathbf{0}]$ where $\mathbf{\Delta}_\sigma$ is the diagonal matrix whose coefficients are the m singular values of $\mathbf{J}$, noted σ_i ($\sigma_i > 0$), the condition (II.21) can be written as:

$$\nabla\mathcal{L}^\top \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \dot{\mathbf{q}}_\mathbf{2} = \widetilde{\nabla\mathcal{L}}^\top \mathbf{\Sigma}\, \widetilde{\dot{\mathbf{q}}_\mathbf{2}} \leq \mathbf{0} \tag{II.22}$$

where $\widetilde{\nabla\mathcal{L}} = \mathbf{U}^\top \nabla\mathcal{L}$ and $\widetilde{\dot{\mathbf{q}}_\mathbf{2}} = \mathbf{V}^\top \dot{\mathbf{q}}_\mathbf{2}$.

To simplify the construction of the projection operator, and by considering the elements $\tilde{l}_i$ of the diagonal matrix $\widetilde{\nabla\mathcal{L}} = (\tilde{l}_1, ..., \tilde{l}_m)$, this condition is restricted to the following:

$$\forall i \in [1...m], \quad \tilde{l}_i\, \sigma_i\, \widetilde{\dot{q}}_{2_i} \leq 0 \tag{II.23}$$

This condition is more restrictive than the previous one given in (II.22). However, it ensures a better behavior of the robot. Particularly, if the Lyapunov function is the norm of the error (as classically done), this condition ensures the convergence of each singular components of the error separately, while (II.22) ensures only the convergence of the norm of the error, which can lead to a temporary increase of one or several components. By (II.23), we ensure that each component will be faster than the required motion $\dot{\mathbf{e}}$, avoiding thus such problems.

For a given secondary control law $\mathbf{z}$, the secondary term $\dot{\mathbf{q}}_\mathbf{2}$ that respect (II.23) is built by keeping the components that respect this condition and by nullifying the other components:

$$\widetilde{\dot{\mathbf{q}}_{\mathbf{2i}}} = \begin{cases} \tilde{z}_i & \text{if} \quad i > m \ \text{ or } \ \tilde{z}_i = 0 \\ \tilde{z}_i & \text{if} \quad \tilde{z}_i\, \tilde{l}_i\, \sigma_i < 0 \\ 0 & \text{if} \quad \tilde{z}_i\, \tilde{l}_i\, \sigma_i > 0 \end{cases}$$

This equation can be represented in a matrix form:

$$\widetilde{\dot{\mathbf{q}}_\mathbf{2}} = \mathbf{V}^\top \dot{\mathbf{q}}_\mathbf{2} = \mathbf{V}^\top \mathbf{P}_{(\tilde{\mathbf{z}})} \mathbf{z} = \mathbf{P}_{(\tilde{\mathbf{z}})} \mathbf{V}^\top \mathbf{z} = \mathbf{P}_{(\tilde{\mathbf{z}})} \tilde{\mathbf{z}} = \begin{bmatrix} p_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & p_n(\tilde{z}) \end{bmatrix} \tilde{\mathbf{z}} \tag{II.24}$$

where $\tilde{\mathbf{z}} = \mathbf{V}^\top \mathbf{z}$ and the components $p_i(\tilde{\mathbf{z}})$ of $\mathbf{P}_{(\tilde{\mathbf{z}})}$ are defined by:

$$p_i(\tilde{\mathbf{z}}) = \begin{cases} 1 & \text{if} \quad i > m \ \text{ or } \ \tilde{z}_i = 0 \\ 1 & \text{if} \quad \tilde{z}_i\, \tilde{l}_i\, \sigma_i < 0 \\ 0 & \text{if} \quad \tilde{z}_i\, \tilde{l}_i\, \sigma_i > 0 \end{cases} \tag{II.25}$$

Finally, the control law that realizes the main task $\mathbf{e}$ and ensures that the secondary control law $\mathbf{z}$ respects condition (II.23) can finally be written:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \dot{\mathbf{e}} + \mathbf{P}_\mathbf{z}\, \mathbf{z} \qquad \text{where} \ \ \mathbf{P}_\mathbf{z} = \mathbf{V}\, \mathbf{P}_{(\tilde{\mathbf{z}})} \mathbf{V}^\top \tag{II.26}$$

Using the same representation as for the orthogonal projection, we show in Fig. II.7 the main and secondary tasks with their projection. Using this approach the null space is the non-hashed half plan, it includes the null space line of the classical approach. If the secondary task is in the null space, the vector is not modified by the projection (such as vectors $\mathbf{z}_1$ and $\mathbf{z}_2$). If the secondary task is out of this space, it is projected into the $Ker(\mathbf{J})$ line (such as vector $\mathbf{z}_3$).

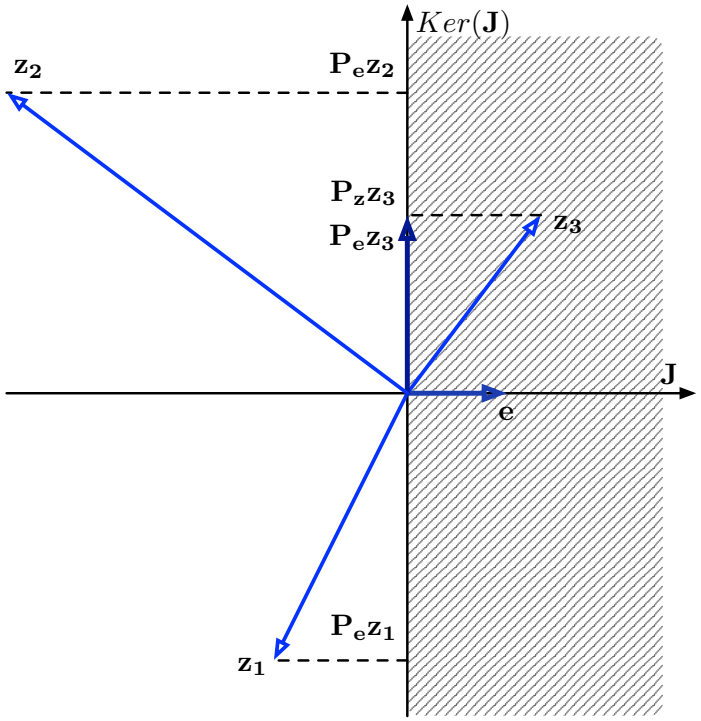


Figure II.7 – Example of directional projection of simple tasks using the control law (II.24)

However, the second term $\mathbf{P_z}\,\mathbf{z}$ is not proportional to the error to regulate and can be arbitrary large when the main task converges to zero, which may introduce oscillation (as for vector $\mathbf{z}_2$). The effect is very similar to the oscillations induced by a too large value of the gain on a sampled system. This problem is corrected by computing the projection operator from the second-order Taylor expansion which introduces an upper bound on the value of the secondary term.

In this case, when considering the norm of the error as a Lyapunov function ($\mathcal{L} = \frac{1}{2}\mathbf{e}^\top\mathbf{e}$), the condition to be respected is:

$$\nabla\mathcal{L}^\top\mathbf{J}\Delta\mathbf{q}_2 + \frac{1}{2}\Delta\mathbf{q}_2{}^\top\mathbf{J}^\top\mathbf{J}\Delta\mathbf{q}_2 < \mathbf{0}$$

where $\nabla\mathcal{L}^\top = (\mathbf{e} + \Delta\mathbf{e}^*)^\top$, such as $\Delta\mathbf{e}^* = \dot{\mathbf{e}}\,\Delta t$ is an infinitesimal motion in the task space, and Δt the sampling interval (for more details refer to [MC09]).

When introducing the SVD bases and performing some simplifications, the above condition can be written as:

$$\forall i \in [1...m], \qquad 2(\tilde{e}_i + \widetilde{\Delta e_i^*})\sigma_i\widetilde{\Delta q_{2_i}} + \sigma_i^2\widetilde{\Delta q_{2_i}}^2 \leq 0$$

where $\tilde{\mathbf{e}} = U^\top\mathbf{e} = (\tilde{e}_1, ..., \tilde{e}_m)$, $\tilde{\mathbf{q}} = V^\top\mathbf{q}$, and $\widetilde{\Delta q_2}$ is a possible secondary motion that belongs to the free space of the main task if and only if:

$$\forall i \in [1...m] \quad \begin{cases} \widetilde{\Delta q_{2_i}} = 0 \\ \text{or} \quad 0 < \widetilde{\Delta q_{2_i}} \leq -\frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \\ \text{or} \quad -\frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \leq \widetilde{\Delta q_{2_i}} < 0 \end{cases} \tag{II.27}$$

In order to ensure that $\widetilde{\Delta q_{2_i}}$ respects condition (II.27), the general expression of $\widetilde{\Delta \mathbf{q}_{2_i}}$ is defined by:

$$\widetilde{\Delta \mathbf{q}_{2_i}} = \begin{cases} \tilde{z}_i & \text{if} \quad i > m \text{ or } \tilde{z}_i = 0 \\ \tilde{z}_i & \text{if} \quad 0 < \tilde{z}_i \leq -\frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \\ \tilde{z}_i & \text{if} \quad -\frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \leq \tilde{z}_i < 0 \\ 0 & \text{if} \quad \sigma_i \tilde{z}_i > 0 \text{ and } (\tilde{e}_i + \widetilde{\Delta e_i^*}) < 0 \\ 0 & \text{if} \quad \sigma_i \tilde{z}_i < 0 \text{ and } (\tilde{e}_i + \widetilde{\Delta e_i^*}) > 0 \\ 2(\tilde{e}_i + \widetilde{\Delta e_i^*}) & \text{otherwise} \end{cases}$$

This equation can be written in a matrix form, and the projection operator will be given by:

$$\widetilde{\mathbf{P}_{\tilde{\mathbf{z}}}} = \mathbf{V} \begin{bmatrix} p_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & p_n(\tilde{z}) \end{bmatrix} \mathbf{V}^\top \tag{II.28}$$

where $p_i(\tilde{z})$ is defined by:

$$p_i(\tilde{z}) = \begin{cases} 1 & \text{if} \quad i > m \text{ or } \tilde{z}_i = 0 \\ 1 & \text{if} \quad 0 < \tilde{z}_i - \frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \\ 1 & \text{if} \quad -\frac{2}{\sigma_i}(\tilde{e}_i + \widetilde{\Delta e_i^*}) \leq \tilde{z}_i \\ 0 & \text{if} \quad \sigma_i \tilde{z}_i > 0 \text{ and } (\tilde{e}_i + \widetilde{\Delta e_i^*}) < 0 \\ 0 & \text{if} \quad \sigma_i \tilde{z}_i < 0 \text{ and } (\tilde{e}_i + \widetilde{\Delta e_i^*}) > 0 \\ -\frac{2(\tilde{e}_i + \widetilde{\Delta e_i^*})}{\sigma_i \tilde{z}_i} & \text{otherwise} \end{cases} \tag{II.29}$$

This projection operator (II.28) obtained by considering the robot as a discrete system allows the evasion of oscillations that appear when the projection operator (II.24) is used. Compared to the control law in (II.24), the second projector (II.28) ensure an upper boundary for the secondary task.

This upper limit is represented in Fig. II.8. If the classical redundancy formalism (II.13) is used, the secondary control law is projected onto the null space of the main task: the secondary terms $\mathbf{z}_i$ are projected to the corresponding $\mathbf{P_e z_i}$ ($i = 1...3$). In the case of directional projection (II.24), only one bound is used: the free space is the left half-plane.

The secondary terms $\mathbf{z}_1$ and $\mathbf{z}_2$ are not modified by the projection. However, $\mathbf{z}_2$ has a large effect on the main task, and may result in oscillations or even instability, in particular if the time interval between two iterations is large.

A second bound is then added by equation (II.28): the left hashed region is forbidden too. The secondary term $\mathbf{z}_2$ is then projected on $\mathbf{P_z z}_2$, which prevents any oscillation at task regulation.

Furthermore, compared to the classical method (II.13), the number of non-zero coefficients of $\mathbf{P_z}$ is greater than the number we can find in the classical projector. In fact, each time an element of the diagonal is non-zero, a DOF is released and can be used to improve the execution of secondary tasks.

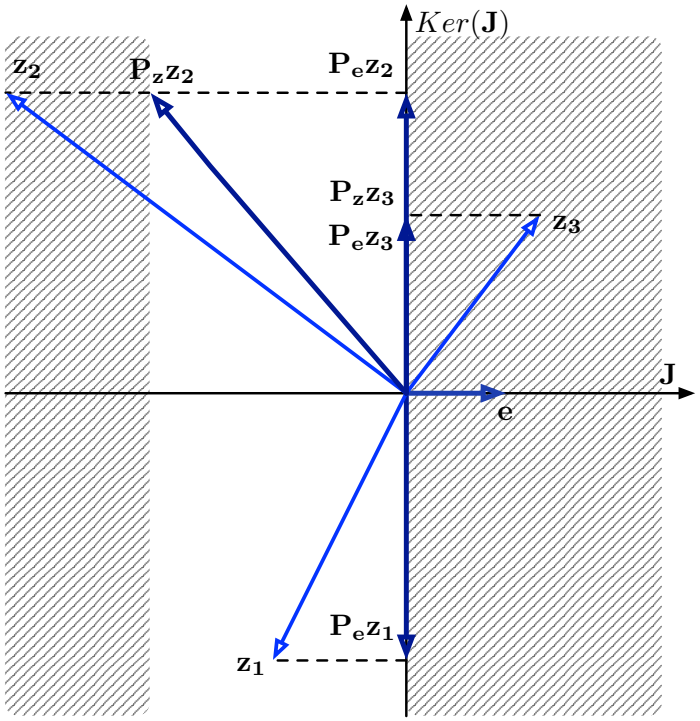


Figure II.8 – Comparison of the free space using the classical and directional redundancy frameworks.

The null spaces of the classical approach and the directional one are represented in Fig. II.9, in case of a task of dimension 2 in a joint space of dimension 3. Using classical projection, the null space of the main task is a line: the first component of the secondary task is considered, the others are cancelled. Using directional redundancy, the null space is a cone which includes the line of the classical approach [Man06].

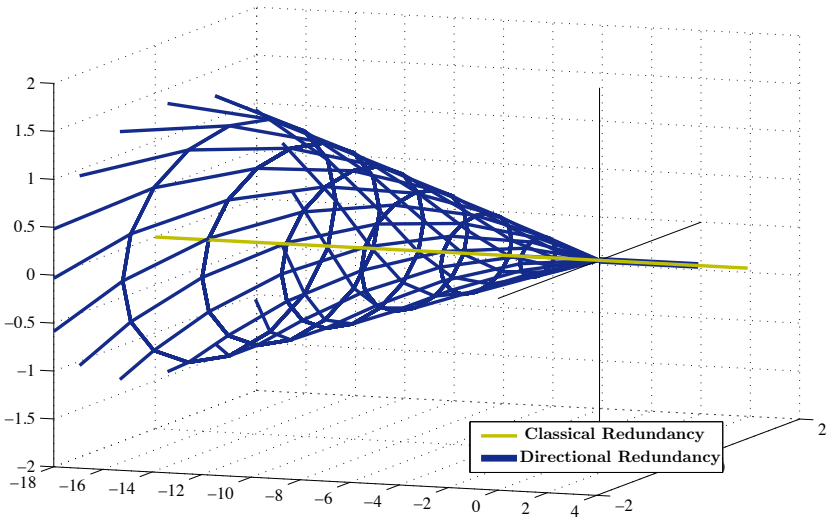


Figure II.9 – Comparison between the null spaces of the orthogonal projection and the directional one in case of a task of dimension 2 in a joint space of dimension 3 [Man06]

II.3.3 Minimum Norm Solution Method

The main idea of the minimum norm solution method given by [MC10] is to consider the norm $\eta = \|\mathbf{e}\|^\gamma$ where $\gamma \in \mathbb{R} - \{0\}$ to build a new projection operator $\mathbf{P}_\eta$.

Instead of considering a desired decrease $\dot{\mathbf{e}}$ of the error vector $\mathbf{e}$, this approach considers a desired decrease $\dot{\eta}$ of the error norm, such as the classical exponential decrease: $\dot{\eta} = -\lambda\eta$.

Thus, the least square solution $\dot{\mathbf{q}}_\eta$ will be given by:

$$\dot{\mathbf{q}}_\eta = \mathbf{J}_\eta^+ \, \dot{\eta} \tag{II.30}$$

where $\mathbf{J}_\eta = \gamma\|\mathbf{e}\|^{\gamma-2}\mathbf{e}^\top\mathbf{J} \in \mathbb{R}^{1\times n}$ and $\mathbf{J}_\eta^+ = \frac{1}{\gamma\|\mathbf{e}\|^{\gamma-2}(\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e})}\mathbf{J}^\top\mathbf{e}$

The general control law will be:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_\eta + \dot{\mathbf{q}}_\eta^\top = \dot{\mathbf{q}}_\eta + \mathbf{P}_\eta \, \mathbf{z} \tag{II.31}$$

where $\mathbf{z}$ is any vector that can be designed to try to realize additional secondary tasks and $\mathbf{P}_\eta = \mathbf{I}_\eta - \mathbf{J}_\eta^+\mathbf{J}_\eta$ is a projection operator into the null space of $\mathbf{J}_\eta$ given by:

$$\mathbf{P}_\eta = \mathbf{P}_{\|\mathbf{e}\|} = \mathbf{I}_n - \frac{1}{\mathbf{e}^\top \, \mathbf{J} \, \mathbf{J}^\top \, \mathbf{e}} \, \mathbf{J}^\top \, \mathbf{e} \, \mathbf{e}^\top \, \mathbf{J}$$

Note that $\mathbf{P}_\eta$ doesn't depend of the value of η, it is thus denoted $\mathbf{P}_{\|\mathbf{e}\|}$. Furthermore, since $\mathbf{J}_\eta$ is at most of rank 1, $\mathbf{P}_{\|\mathbf{e}\|}$ is at least of rank $n-1$, which will thus not filter a lot the secondary task $\mathbf{z}$.

That is the main contribution of this method especially that, in the classical approach, the rank of $\mathbf{P}_\mathbf{e}$ is equal to m. As soon as $(n-m) > 1$ supplementary directions of motions are thus available to achieve the secondary tasks. That is particularly true when $\mathbf{J}$ is of full rank n, in which case $\mathbf{P}_\mathbf{e} = 0$ and no secondary task at all can be considered in that usual case.

However, the drawback of this method appears near convergence (that is when $\mathbf{e} \to \mathbf{0}$) where the denominator of $\mathbf{J}_\eta^+$ approaches zero. Thus, the Jacobian is singular, and the system (II.31) becomes unstable. A solution to this problem was given in [MC10]: as soon as the system nears its goal, they switch from $\mathbf{P}_{\|\mathbf{e}\|}$ to the classical projection operator $\mathbf{P}_\mathbf{e}$. This switching will ensure the convergence of the system since it allows solving the instability problem of $\mathbf{P}_{\|\mathbf{e}\|}$ as $\mathbf{e} \to \mathbf{0}$.

The switching strategy consists in defining a convex combinatory $\mathbf{P}_\alpha$ between the classical and the new projection operator such that:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_\eta + \mathbf{P}_\alpha \, \mathbf{z} \quad \text{with} \quad \mathbf{P}_\alpha = \bar{\alpha}\left(\|\mathbf{e}\|\right)\mathbf{P}_{\|\mathbf{e}\|} + \left(1 - \bar{\alpha}\left(\|\mathbf{e}\|\right)\right)\mathbf{P}_\mathbf{e} \tag{II.32}$$

where the switching function $\bar{\alpha}\left(\|\mathbf{e}\|\right) \to [0,1]$ is defined by:

$$\bar{\alpha}\left(\|\mathbf{e}\|\right) = \begin{cases} 1 & if \quad e_1 < \|\mathbf{e}\| \\ \frac{\alpha(\|\mathbf{e}\|)-\alpha_0}{\alpha_1-\alpha_0} & if \quad e_0 \leq \|\mathbf{e}\| \leq e_1 \\ 0 & if \quad \|\mathbf{e}\| < e_0 \end{cases}$$

where e_0, e_1 are two threshold values that define the starting and the ending conditions for the switching period, they should be selected such that the system does not converge too fast during the interval $[e_0, e_1]$. $\alpha(t)$ is a continuous monotonically decreasing function, such that $\alpha_0 = \alpha(e_0) \approx 0$ and $\alpha_1 = \alpha(e_1) \approx 1$ (Fig. II.10).

A good selection for the function $\alpha(\|\mathbf{e}\|)$ is the sigmoid function given as:

$$\alpha(\|\mathbf{e}\|) = \frac{1}{1 + \exp\left(-12\frac{\|\mathbf{e}\| - e_0}{e_1 - e_0} + 6\right)}$$

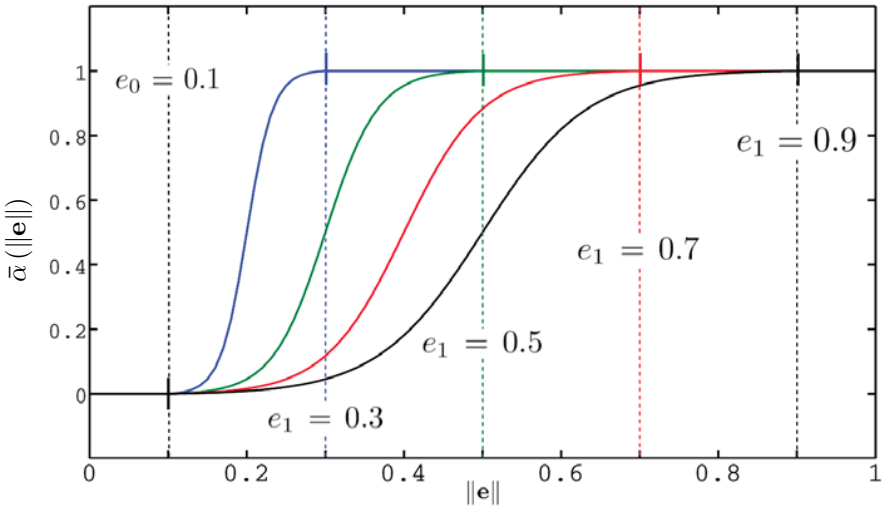


Figure II.10 – Switching function $\bar{\alpha}(\|\mathbf{e}\|)$ for the minimum norm solution method

Stability proof

Finally, the stability analysis for the control scheme in (II.30) is carried out using the candidate Lyapunov function $\mathcal{L}_{(\mathbf{P}_\eta)} = \eta^2$. By taking the derivative of $\mathcal{L}_{(\mathbf{P}_\eta)}$ and injecting $\mathbf{J}_\eta$ value in the result, we obtain:

$$\dot{\mathcal{L}}_{(\mathbf{P}_\eta)} = 2\,\eta\,\dot{\eta} = 2\,\gamma\,\|\mathbf{e}\|^{2\gamma-2}\,\mathbf{e}^\top\,\mathbf{J}\,\dot{\mathbf{q}}$$

Using (II.30) and the exponential decrease of the error norm ($\dot{\eta} = -\lambda\eta$), we get:

$$\dot{\mathcal{L}}_{(\mathbf{P}_\eta)} = \frac{-2\,\lambda\,\|\mathbf{e}\|^{2\gamma}}{\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e}}\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e} = -2\,\lambda\,\|\mathbf{e}\|^{2\gamma} \quad \text{when} \quad \mathbf{J}^\top\mathbf{e} \neq \mathbf{0}$$

We have $\dot{\mathcal{L}}_{(\mathbf{P}_\eta)} < 0$ as soon as $\mathbf{e} \neq \mathbf{0}$ and $\mathbf{e} \notin Ker(\mathbf{J}^\top)$, thus ensuring the same asymptotic stability properties of the system as in the classical orthogonal case.

II.3.4 Comparison Between Projection Operators

The classical orthogonal projection approach (II.13) projects the vector representing the secondary task onto the null space of the Jacobian of the main task. Thus, the secondary tasks/ constraints have no effect on the convergence of the main task in the cartesian space. For a desired exponential decrease of the main task error, the stability of the system can be verified by using the classical Lyapunov function based on the error norm $\left(\mathcal{L}(\mathbf{e}) = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}\mathbf{e}^\top\mathbf{e}\right)$.

The main limitation of this method is that only the DOF which are not controlled by the main task can be exploited to perform other criteria. The more complicated the main task is, the more DOF it uses, and the more difficult it is to apply the secondary constraints. Of course, if the main task uses all the DOF of the robot, no secondary constraint can be taken into account.

The bidirectional projection approach (II.26) helps the main task to be completed faster by allowing motions produced by the secondary task. In fact, this method ensures the stability of the system by prohibiting the secondary control law from increasing the error of the main task. Moreover, it enhances the performance of the secondary task by enlarging the number of available DOF.

In contrast to the first method which projects all secondary tasks onto the null space of the main task, this approach differentiate between two cases: if the secondary task goes in the same direction as the main task, it is maintained as long it does not increase the main task error and retain the stability condition, otherwise if the secondary task goes in the opposite direction to the main task, it is projected onto the null space of the main task.

Using the desired exponential decrease of the task error and the classical Lyapunov function, the bidirectional projection operator is established by considering the above cases in the space of singular values. However, even if this projection operator improves the performance of the system, the number of DOF can be insufficient to respect the constraints.

Instead of decreasing the error components, the minimum norm solution method (II.31) considers a desired exponential decrease of the error norm ($\dot{\eta} = -\lambda\eta$). An orthogonal projector is used to project secondary task onto the null space of the norm Jacobian $\mathbf{J}_\eta$ of the main task. In this case, the secondary tasks/constraints have no effect on the error's norm of the main task in the norm space.

A switching strategy is used to ensure that the new projection operator smoothly switches to the classical orthogonal projection operator as soon as the error nears zero. For the desired decrease of the error's norm, the stability of the system can be verified by using the classical Lyapunov function based on the error norm $\mathcal{L}(\eta) = \eta^2$. The main advantage of this solution is that it is always at least of rank $(n-1)$ allowing it to be used even if the main task is of full rank. However, the switching to the classical projection may lead to unsatisfactory behavior and in some cases to instability.

The performance of these methods, when applied to several tasks, will be studied in detail in next section using different task sequencing methods.

II.4 Task Priority Formalisms

In contrast to the previous section, where a single secondary task is projected on the main task, the following section presents various frameworks proposed in order to manage the redundancy among several tasks to control their levels of priority. The successive, augmented and intersection projections (using Jacobian pseudo-inverse and transpose) are presented. An efficient task priority formalism based on the least squares optimal solution is also introduced. These methods are applied when executing several tasks using the three projection operators described in the previous section.

II.4.1 Classical Task Sequencing Methods

II.4.1.1 Case of two tasks

Two task functions $\mathbf{e}_1$ and $\mathbf{e}_2$ with their corresponding perfectly known and full rank Jacobians $\mathbf{J}_1$ and $\mathbf{J}_2$ respectively are considered. The system is controlled using joint velocity $\dot{\mathbf{q}}$ which is given by:

$$\dot{\mathbf{q}} = \mathbf{J}_1^+ \dot{\mathbf{e}}_1 + \mathbf{P}_1 \mathbf{z} \tag{II.33}$$

where $\dot{\mathbf{e}}_1$ is the desired variation of $\mathbf{e}_1$ (that will perform the desired motion in the task space), and $\mathbf{P}_1$ is the generalized projection matrix applied on the additional joint-space velocity vector $\mathbf{z}$.

The vector $\mathbf{z}$ will be used to apply the secondary task $\mathbf{e}_2$. A first solution is to choose $\mathbf{z}$ equal to the control calculated from $\mathbf{e}_2$ only ($\mathbf{z}_2 = \mathbf{J}_2^+ \dot{\mathbf{e}}_2$) which gives:

$$\dot{\mathbf{q}}_{12} = \mathbf{J}_1^+ \dot{\mathbf{e}}_1 + \mathbf{P}_1 \mathbf{J}_2^+ \dot{\mathbf{e}}_2 \tag{II.34}$$

II.4.1.2 Case of several tasks

(a) Successive inverse-based projections

In [MC04], several approaches were discussed to propose suitable ones that enable stacking n different tasks $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n$ using redundancy. A first solution consider the case $\mathbf{e}_3$ directly in the control calculated from $\mathbf{e}_2$:

$$\dot{\mathbf{q}} = \mathbf{J}_1^+ \dot{\mathbf{e}}_1 + \mathbf{P}_1 (\mathbf{J}_2^+ \dot{\mathbf{e}}_2 + \mathbf{P}_2 \mathbf{z})$$

Thus the successive inverse-based projections can be extended to n tasks: the last task is projected on the null space of its previous task and then project this composed task on the null space of their previous tasks to get:

$$\dot{\mathbf{e}} = \dot{\mathbf{e}}_1 + \mathbf{P}_1 \dot{\mathbf{e}}_2 + \mathbf{P}_1 \mathbf{P}_2 \dot{\mathbf{e}}_3 + ... + \prod_{i=1}^{n-1} \mathbf{P}_i \dot{\mathbf{e}}_n \tag{II.35}$$

The null space projectors are not commutative and the solution obtained by (II.35) may lead to conservative stability conditions. Since $\mathbf{P}_1 ... \mathbf{P}_{n-1} \dot{\mathbf{e}}_n$ is not in the null space of $\mathbf{e}_2, ..., \mathbf{e}_{n-1}$, the low level priority task $\mathbf{e}_n$ may modify tasks $\mathbf{e}_2, ..., \mathbf{e}_{n-1}$ of higher level priority. This stability problem can be simply solved by the augmented method, or better by the intersection method [Ant09].

(b) Augmented inverse-based projections

In this method, the generic task is projected onto the null space of the task achieved by considering the extended Jacobian of all the higher priority ones. By assuming that $\mathbf{e_{1...n}}$ is a task that realizes the n first tasks while respecting their priorities, the orthogonal projection operator $\mathbf{P_{1...n}}$ is given by [BB98]:

$$\mathbf{P_{1...n}} = \mathbf{I} - \mathbf{J^{+}_{1...n}}\mathbf{J_{1...n}}$$

where $\mathbf{J_{1...n}}$ is the Jacobian of $\mathbf{e_{1...n}}$ defined by:

$$\mathbf{J_{1...n}} = \frac{\partial \mathbf{e_{1...n}}}{\partial \mathbf{q}} = \frac{\partial(\mathbf{e_1} + \mathbf{P_1}\mathbf{e_2} + ... + \mathbf{P_{1...n}}\mathbf{e_n})}{\partial \mathbf{q}}$$

The approximation value $\mathbf{J_{1...n}} = \mathbf{J_1} + \mathbf{P_1}\mathbf{J_2} + ...\mathbf{P_{1...n}}\mathbf{J_n}$ is used to compute $\mathbf{P_{1...n}}$ by assuming that $\frac{\partial \mathbf{P_{1...n}}}{\partial \mathbf{q}} = \mathbf{0}$ since it is difficult to compute. However, this approximation does not guarantee that the error of the first n tasks remains 0 when the control due to $\mathbf{e_{n+1}}$ is high.

(c) Intersection inverse-based projections method

To ensure that the motion will be projected if it is in the null space of all n tasks, the new task $\mathbf{e_{n+1}}$ is projected onto the space $\mathbf{N_{i...n}}$ defined as the intersection of the null spaces for all n previous tasks. $\mathbf{N_{i...n}}$ is given by:

$$\mathbf{N_{i...n}} = \bigcap_{k=1}^{n} \text{Null}(\mathbf{J}_k)$$

The null space can be computed by stacking the Jacobian of the n tasks such that:

$$\mathbf{N_{i...n}} = \text{Null}\begin{pmatrix} \mathbf{J_1} \\ \vdots \\ \mathbf{J_n} \end{pmatrix}$$

In [CCSS91], instead of using inverse kinematics schemes that use the Jacobian pseudo-inverse, a transpose-based task-priority redundancy resolution has been proposed. In [Ant09], transpose projection approach is generalized by following the same guidelines previously mentioned for the inverse-based projection approaches, i.e., by successively projecting the lower priority tasks into the null space of higher priority task. The augmented transpose-based projection was also used: the lower priority task is projected onto the null space of the augmented Jacobian obtained by stacking all the higher priority tasks while the Jacobian transpose is used.

A unified formula for redundancy was defined in [Ant09], where the four priority-based approaches have been grouped in one formula and their stability analysis has been discussed. For n tasks this unified formula is written as:

$$\dot{\mathbf{q}} = \mathbf{J_1^{\#}}\dot{\mathbf{e}}_1 + \bar{\mathbf{P}}_1\mathbf{J_2^{\#}}\dot{\mathbf{e}}_2 + \bar{\mathbf{P}}_2\mathbf{J_3^{\#}}\dot{\mathbf{e}}_3 + ... + \bar{\mathbf{P}}_{\mathbf{n-1}}\mathbf{J_n^{\#}}\dot{\mathbf{e}}_n \tag{II.36}$$

where for the successive projection $\bar{\mathbf{P}}_\mathbf{k} = \mathbf{P_1}\mathbf{P_2}...\mathbf{P_k}$, and for the augmented projection $\bar{\mathbf{P}}_\mathbf{k} = \mathbf{P_{1...k}}$, while for transpose-based or for inverse-based method (#) is replaced by transpose ($\top$) or ($+$) respectively.

II.4.2 Efficient Task Sequencing for Orthogonal Projection

As previously mentioned, the solution found in (II.36) ensured that the desired priority between tasks is respected. Thus the high priority task $\mathbf{e_1}$ is executed in addition to the part of $\mathbf{e_2}$ which is not in the null space of $\mathbf{e_1}$.

We can easily verify that this control law respect the desired variation of the main task, in case of the orthogonal projection:

$$\dot{\mathbf{e}}_{1/\dot{\mathbf{q}}_{12}} = \mathbf{J_1}\,\dot{\mathbf{q}}_{12} = \mathbf{J_1}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \mathbf{J_1}\,\mathbf{P_1}\,\mathbf{J_2^+}\,\dot{\mathbf{e}}_2 = \dot{\mathbf{e}}_1$$

Thus the joint motion $\dot{\mathbf{q}}$ given by (II.34) produces exactly the specified motion $\dot{\mathbf{e}}_1$ in the task function space. However it's not the optimal one for the secondary task $\mathbf{z}$:

$$\begin{aligned} \dot{\mathbf{e}}_{2/\dot{\mathbf{q}}_{12}} &= \mathbf{J_2}\,\dot{\mathbf{q}} \\ &= \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \mathbf{J_2}\,(\mathbf{I} - \mathbf{J_1^+}\mathbf{J_1})\,(\mathbf{J_2^+}\dot{\mathbf{e}}_2) \\ &= \dot{\mathbf{e}}_2 + \mathbf{J_2}\,\mathbf{J_1^+}(\dot{\mathbf{e}}_1 - \mathbf{J_1}\,\mathbf{J_2^+}\,\dot{\mathbf{e}}_2) \end{aligned} \tag{II.37}$$

In fact, the secondary task is completely regulated if and only if the two tasks are completely decoupled (in this case $\mathbf{J_2}\,\mathbf{J_1^+} = \mathbf{0}$).

Thus, in general, the joint motion $\dot{\mathbf{q}}$ is chosen to realize exactly the motion $\dot{\mathbf{e}}_1$ in the task function space, and to perform at best a secondary task whose corresponding joint motion is $\mathbf{z}$, this control is not the optimal least squares solution. The null space projection of the prioritized tasks is not taken into account when the pseudo-inverse of the Jacobian is calculated [SS91].

Another solution is presented below, it uses the projection data to calculate the pseudo-inverse using an arbitrary vector $\mathbf{z}$ as the least squares solution.

II.4.2.1 Two tasks

Two tasks $\mathbf{e}_1$ and $\mathbf{e}_2$ are considered such as $\mathbf{e}_1$ has priority over $\mathbf{e}_2$. When the robot is controlled using its articular velocity, the general solution of the primary task is given by (II.33):

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\dot{\mathbf{e}}_1 + \mathbf{P_1}\,\mathbf{z} \tag{II.38}$$

Using $\dot{\mathbf{e}}_2 = \mathbf{J_2}\,\dot{\mathbf{q}}$, we get:

$$\dot{\mathbf{e}}_2 = \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \mathbf{J_2}\mathbf{P_1}\,\mathbf{z} \tag{II.39}$$

Solving (II.39) for $\mathbf{z}$ and injecting the computed $\mathbf{z}$ in (II.38) we get:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\dot{\mathbf{e}}_1 + \mathbf{P_1}(\mathbf{J_2}\mathbf{P_1})^+\,(\dot{\mathbf{e}}_2 - \mathbf{J_2}\mathbf{J_1^+}\dot{\mathbf{e}}_1) \tag{II.40}$$

Since $\mathbf{P}_1$ is idempotent and Hermetian then (II.40) becomes:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + (\mathbf{J_2}\mathbf{P_1})^+\left(\dot{\mathbf{e}}_2 - \mathbf{J_2}\mathbf{J_1^+}\dot{\mathbf{e}}_1\right) \tag{II.41}$$

where $\mathbf{J_2}\mathbf{P_1}$ is the limited Jacobian of $\mathbf{e}_2$ and $\left(\dot{\mathbf{e}}_2 - \mathbf{J_2}\mathbf{J_1^+}\dot{\mathbf{e}}_1\right)$ is the secondary control component after removing the part of $\mathbf{e}_2$ accomplished by $\mathbf{e}_1$.

We can check that, in contrary to the classical control law (II.37), the desired secondary tasks is executed with this control law:

$$\begin{aligned}
\mathbf{J_2}\,\dot{\mathbf{q}} &= \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \mathbf{J_2}\,(\mathbf{J_2}\,\mathbf{P_1})^+\left(\dot{\mathbf{e}}_2 - \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1\right) \\
&= \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \mathbf{J_2}\,\mathbf{P_1}\,(\mathbf{J_2}\,\mathbf{P_1})^+\dot{\mathbf{e}}_2 - \mathbf{J_2}\,\mathbf{P_1}\,(\mathbf{J_2}\,\mathbf{P_1})^+\mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 \\
&= \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 + \dot{\mathbf{e}}_2 - \mathbf{J_2}\,\mathbf{J_1^+}\,\dot{\mathbf{e}}_1 \\
&= \dot{\mathbf{e}}_2
\end{aligned}$$

In addition to that, Figures II.11 and II.12 geometrically shows that (II.41) can execute both tasks simultaneously, whereas (II.34) cannot execute the secondary task exactly.

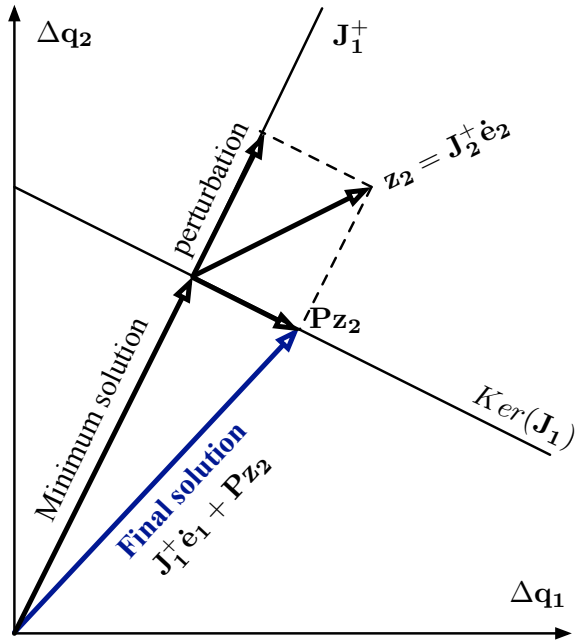


Figure II.11 – First task sequencing method (II.34): The secondary task vector is projected into the null space of the main task ($Ker(\mathbf{J}_1)$). The final solution is the resultant of this projection with the minimum solution of the main task.

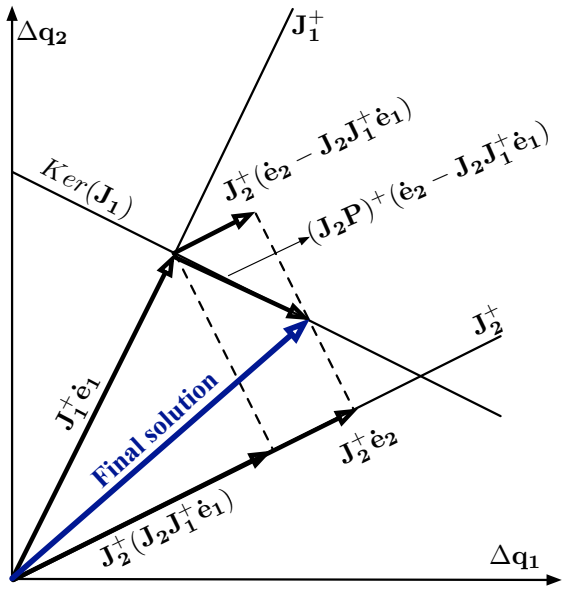


Figure II.12 – Second task sequencing method (II.41): The part of the secondary task $\mathbf{e}_2$ which is executed by the main task is removed from the minimum solution $\mathbf{J_2^+}\dot{\mathbf{e}}_2$. The resultant is then projected into the null space of the main task to obtain the final solution.

II.4.2.2 Several tasks

Let $\mathbf{e_1} \in \mathbb{R}^{m_1} ... \ \mathbf{e_k} \in \mathbb{R}^{m_k}$ be k tasks where the i^{th} task Jacobian is $\mathbf{J_i} \in \mathbb{R}^{m_i \times n}$. If the i^{th} task is to be executed with lower priority than the $(i-1)^{th}$ task, the general control scheme that allows the execution of the k tasks is given by $\dot{\mathbf{q}} = \dot{\mathbf{q}}_{\mathbf{1...k}}$.

This joint velocity vector is calculated from $\dot{\mathbf{q}}_{\mathbf{1...i}}$ which is recursively defined by:

$$\dot{\mathbf{q}}_{\mathbf{1...i}} = \begin{cases} \dot{\mathbf{q}}_{\mathbf{1...i-1}} + \left(\mathbf{J_i P^A_{i-1}}\right)^{+} \left(\dot{\mathbf{e}}_{\mathbf{i}} - \mathbf{J_i}\dot{\mathbf{q}}_{\mathbf{1...i-1}}\right) & if \quad i = 2, \cdots k. \\ \mathbf{J_1^{+}}\dot{\mathbf{e}}_{\mathbf{1}} & if \quad i = 1 \end{cases} \tag{II.42}$$

where $\mathbf{P^A_i} = \mathbf{I_n} - \mathbf{J_i^{A^{+}} J_i^A}$ is the orthogonal projection on the null space of the augmented Jacobian $\mathbf{J^A_i} = (\mathbf{J_1}, \ldots, \mathbf{J_i})$.

The recursive formula of the augmented classical projection operator can be written as:

$$\mathbf{P^A_i} = \begin{cases} \mathbf{P^A_{i-1}} - \left(\mathbf{J_i P^A_{i-1}}\right)^{+} \left(\mathbf{J_i P^A_{i-1}}\right) & if \quad i = 2, \cdots k. \\ \mathbf{I} - \mathbf{J_1^{+}} \mathbf{J_1} & if \quad i = 1 \end{cases} \tag{II.43}$$

II.4.2.3 Conclusion

Two techniques were presented to apply the classical projection onto two or several tasks. Practically, the two solutions give very close results especially when the applied tasks are fully decoupled and a sufficient number of DOF is available, thus when a system is well-conditioned. Nevertheless the control laws given by these solutions are not exactly equal.

The method given in (II.34) guaranties the respect of task priority (secondary task does not change the primary task), but the exact regulation and optimal performance of the secondary task is not guaranteed.

The second method is used to obtain an optimal solution (II.41). It guarantees the exact regulation of the secondary task (least-squares solution) and takes into account the task priority. However, this optimal solution is obtained by decreasing the conditioning of the Jacobian matrix of the secondary task, which is subsequently inverted and may lead to abnormal solution (especially when tasks are close or the number of available DOF is not large). Thus, the use of the second solution (II.41) increases the number of singularities in the system [Chi97].

Finally, it is not possible to say, in general, which is the best solution between (II.34) and (II.41). In fact, the choice is related to the applied tasks: if the task should be perfectly executed, and its practical feasibility is guaranteed, the solution (II.41) is the best one to be used. On the other hand, if it is hard to apply the secondary task because of a lack in the number of DOF, thus it is better to use the solution (II.34) to obtain a control law which is feasible even near system singularities (but the secondary task will not be perfectly applied).

II.4.3 Bidirectional Projection: Stack of Tasks

The generalization of the second technique (given by II.41) in case of directional projection is complicated because it requires the use of several optimization techniques under constraints, such as the Linear Matricial Inequality or the Simplex algorithm. However, the first method can be easily used in case of directional projection:

$$\dot{\mathbf{q}} = \mathbf{J}_1^{+}\dot{\mathbf{e}}_1 + \mathbf{P}_z\mathbf{J}_2^{+}\dot{\mathbf{e}}_2$$

Thus the following control law is used in case of directional projection of k tasks:

$$\dot{\mathbf{q}} = \mathbf{J}_1^{+}\dot{\mathbf{e}}_1 + \sum_{i=2}^{k} \mathbf{P}_{\mathbf{z}_i}^{\mathbf{e}_{1\ldots i-1}}\mathbf{J}_i^{+}\dot{\mathbf{e}}_i \tag{II.44}$$

A global architecture is proposed in [MSEK09] to sequence a "Stack of Tasks" in order to reach the desired goal while taking into account several environment constraints. Redundancy and stacking are involved in the proposed framework.

The architecture of the global system is represented in Fig. II.13, it is composed of four controller layers:

- The first controller (sensor-based level) is composed of a stack that orders the active subtasks. The subtask at the bottom level of the stack has higher priority, and the priority decreases as the stack level increases. It computes the control law from the stack.

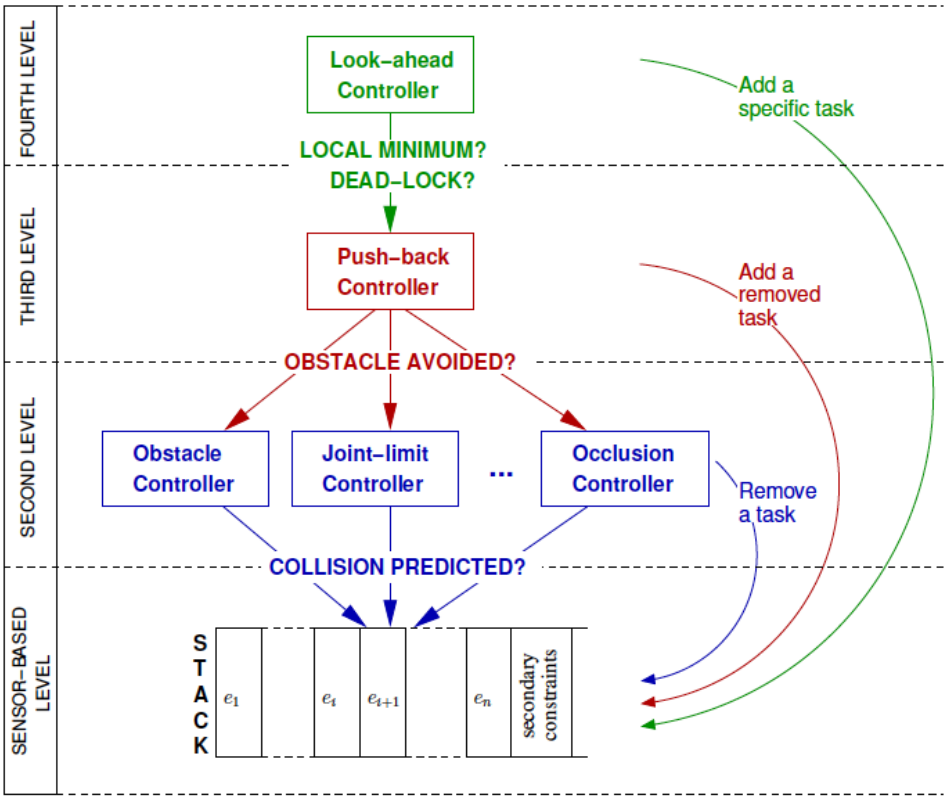


Figure II.13 – Global architecture of the "Stack of Tasks" composed of four controller levels

- The second controller ensures that enough DOF remain free to take the constraints (in blue in the figure) into account, and selects the optimal subtask to be removed from the stack (to ensure that the environment constraints are respected).

- The third controller (push-back controller, in red in the figure) observes the removed subtasks from the stack and tries to put them back in the stack as soon as possible.

- The fourth controller (convergence controller, in green in the figure) ensures the convergence of the global system by solving the dead locks of the bottom controllers.

II.4.4 Minimum Norm Solution Method

The same technique as in the classical orthogonal projection can be used to apply several tasks using the minimum norm solution. Thus, the general control scheme, that allows the execution of the k tasks, is given by $\dot{\mathbf{q}} = \dot{\mathbf{q}}_{\mathbf{1\ldots k}}$.

This joint velocity vector is calculated from $\dot{\mathbf{q}}_{\mathbf{1\ldots i}}$ which is recursively defined by:

$$\dot{\mathbf{q}}_{1\ldots i} = \begin{cases} \dot{\mathbf{q}}_{1\ldots i-1} + \left(\mathbf{J}_i^A \, \mathbf{P}^A_{\|\mathbf{e}_{i-1}\|}\right)^+ \left(\dot{\mathbf{e}}_i - \mathbf{J}_i^A \, \dot{\mathbf{q}}_{1\ldots i-1}\right) & if \quad i = 2, \cdots k. \\ \mathbf{J}_1^+ \dot{\mathbf{e}}_1 & if \quad i = 1 \end{cases} \tag{II.45}$$

where $\mathbf{P}^A_{\|\mathbf{e}_i\|}$ is the new projection operator on the null space of the augmented Jacobian $\mathbf{J}_i^A$, and is recursively given by:

$$\mathbf{P}^A_{\|\mathbf{e}_i^A\|} = \mathbf{I}_n - \frac{1}{a_i^A a_i^{A^\top}} a_i^{A^\top} a_i^A$$

where $a_i^A = \begin{cases} a_{i-1}^A + \mathbf{e}_i^\top \mathbf{J}_i & if \quad i = 2, \cdots k. \\ \mathbf{e}_i^\top \mathbf{J}_i & if \quad i = 1 \end{cases}$

Note that the computational cost required for computing the recursive formula, is higher compared to the other recursive form defined before in (II.43). Furthermore, a switch from $\mathbf{P}^A_{\|\mathbf{e}_i\|}$ to $\mathbf{P}_i^A$ has to be performed as soon as $\mathbf{e}_i \to 0$.

II.5 Conclusion

In this chapter, a classification of the various types of redundancies was developed. In fact, they can be either originally present in the robotic system and its environment, or may arise during tasks execution. In addition to that, the established classification integrates the definition of several study cases, and was illustrated with relevant examples from various robotic domains.

Afterwards, several methods for redundancy resolution and task sequencing were presented and theoretically compared. According to this discussion, we notice that most of the redundancy resolution and task sequencing techniques are achieved either by gathering the desired tasks in a unified set and using a static/dynamic weighting to manage the priority between them (hybrid control), or by stacking these tasks in a continuous formalism for sequencing/switching between the different tasks (partitioned, commutative and hierarchical control).

In addition, the hierarchical control is a particular case of the second category which consists in projecting lower priority tasks into null spaces of the higher priority ones. In this case, two techniques were used to define the projection operator:

1. The first one considers the projector that satisfies the classical (orthogonal $\mathbf{P_e}$ projectors) or extended (bidirectional $\mathbf{P_z}$ projector) Lyapunov stability condition which is based on the error norm.
2. The second one directly projects the error norm into the null space of higher priority tasks (minimum norm solution $\mathbf{P}_\eta$), but stability problems appears in this case and the control law should return to the classical projection of error components near convergence.

In some cases, these methods have better performance than the classical one, especially, due to the increase in the number of available degrees of freedom for applying additional tasks. Although these improvements, there are several limitations related to the definition of the applied tasks and the switching to another control law near convergence as discussed in paragraph II.3.4. Furthermore, there is no clear definition of applications where these redundancy resolution techniques can improve the system's behavior.

Therefore, in the next chapter, a comparison between four types of redundancy resolution techniques is carried out using simulation on different types of robots and under several conditions. Various comparison and performance criteria are also defined and used to show the advantages and drawbacks of each control law and the possible applications where it can be applied. Furthermore, a discussion on the simulation results will lead to the definition of two new generalized methods for redundancy resolution.

III

Comparison of Redundancy Resolution Methods

Contents

After presenting and exploring, in the previous chapter, the different control techniques which are used for redundancy resolution and task sequencing; we will focus, in the next sections, on the comparison between the methods which are relevant for our objective: to control a general redundant system for applying several simultaneous and prioritized tasks.

First of all, the previously addressed methods are summarized in Table III.1, and compared with respect to several criteria: the possibility of applying simultaneous tasks, coupling between tasks, having static/ dynamic priorities if the hierarchy between tasks is realizable, and finally the simplicity of parameters tuning.

Except the commutative control, all task sequencing methods allow the execution of several tasks simultaneously. However, to manage the priority between these tasks, few control laws (in hybrid and hierarchical control) can be used, the simplest one is the orthogonal projection method where no parametrization is required. Furthermore, in case of a dynamic changing of the priority between tasks, only hybrid control with weighting in task space can be used. For the other methods, a re-definition or re-stacking of the executed tasks in the control law is necessary.

In spite of the advantages of all the presented methods, according to the desired objectives of this dissertation, only few control laws are attractive to solve the redundancy in our case. In fact, we are interested in controlling a general multi-arm redundant system while executing several simultaneous tasks with specific hierarchy and priority. Therefore, referring to Table III.1, only hierarchical control method and hybrid control method with weighting in the task space are useful in controlling our system.

Therefore, in the following sections, the efficiency of using these methods to solve the kinematic redundancy is discussed. The goal is to find the most appropriate technique which should be used to control a general redundant system depending on the robot's degree of redundancy and the desired application.

First section addresses the comparison methodology: it presents the used robots, describes the simulated tasks and recalls the control laws of the compared methods. Afterwards, in the second section, we define several comparison criteria in addition to various configurations of the robot and the applied tasks. The simulation results of the explored methods in case of planar robots and LWR Kuka robot are discussed in the third and fourth section.

Finally, the theoretical study and performance comparison of the different methods will leads to the definition of new generalized and simple methods for redundancy resolution that overcome the limitations of the executed methods and pass over the encountered problems during simulation.

Control Type			Simultaneous tasks	Coupling between tasks	Use of hierarchy	Priority type	Ease of parameter tuning
Partitioned Control (II.3)			✓	×	×	-	✓
Commutative Control (II.4)			×	×	×	-	×
Hybrid Control	**Augmented Jacobian (II.6)**		✓	✓	×	-	✓
	Weighting in joint space	**Partitioned (II.7)**	✓	×	×	-	×
		Extended (II.8)	✓	✓	×	-	×
	Weighting in task space (II.9)	**Constant h_i**	✓	✓	✓	Static	×
		Variable h_i	✓	✓	✓	Dynamic	×
Hierarchical Control	**Classical Projection $\mathbf{P_e}$ (II.13)**		✓	✓	✓	Static	✓
	Bidirectional Projection $\mathbf{P_z}$ (II.26)		✓	✓	✓	Static	×
	Minimum Norm Solution $\mathbf{P}_\eta$ (II.32)		✓	✓	✓	Static	×

Table III.1 – Comparison between several control laws for redundancy resolution and task sequencing

III.1 Comparison Methodology

The comparison methodology of the appropriate redundancy resolution techniques will be addressed in this section. Thus, several robot's architectures with sufficient or insufficient degrees of freedom to execute the desired tasks are discussed, in addition to the presentation of the compared control laws.

III.1.1 Robot Presentation and Task Definition

With the aim of a simple and visible comparison between the different redundancy resolution techniques, the planar robots are used in simulation. Considering 'l' as the dimension of the secondary task(s), and '$r = n - m$' as the degree of redundancy of the robot (recall that the robot has 'n' joints and an end effector of mobility 'm'); therefore, the behavior of the redundancy resolution techniques will be studied in case of planar robots in three different cases:

- $l < r$: **Kinematically indeterminate system**
  The system has more free DOF than the dimension of the desired secondary task(s).

- $l = r$: **Kinematically determinate system**
  The secondary task(s) will use exactly all the DOF of the robot.

- $l > r$: **Kinematically over-specified system**
  The secondary task(s) needs more DOF than those available in the system.

Regarding the applied tasks on the planar robots, only positioning of the robot's end-effector and intermediate joints is considered. The choice of such tasks, with fixed number of DOF, facilitates the interpretation of the robot's behavior when applying the chosen methods, and allows the definition of particular and meaningful task's configurations. Therefore, two tasks are defined with decreasing priority (task $\mathbf{T}_1$ has more priority over task $\mathbf{T}_2$):

- $\mathbf{T}_1$ for controlling the robot's end-effector position and orientation.

- $\mathbf{T}_2$ to control the position of an intermediate point.

Thus for planar robots, the end-effector has 3 degrees of mobility ($m = 3$), and the defined secondary task $\mathbf{T}_2$ always has 2 DOF (l=2). Therefore, the comparison is applied for:

- a 7R planar robot with 4 degrees of redundancy (Indeterminate system, $l < r$).

- a 5R planar robot with 2 degrees of redundancy (Determinate system, $l = r$).

- a 4R planar robot with 1 degree of redundancy (Over-specified system, $l > r$).

The Modified Denavit-Hartenberg parameters (MD-H parameters) [KK86] of the used planar robots are given in Table III.2 for the general case of a N-R planar robot, and Fig. III.1 represents the case of a 5R-planar robot.

Joint	σ	α	d	θ	r
1	0	0	0	θ_1	0
2	0	0	1	θ_2	0
3	0	0	1	θ_3	0
4	0	0	1	θ_4	0
5	0	0	1	θ_5	0
6	0	0	1	θ_6	0
⋮	⋮	⋮	⋮	⋮	⋮
N	0	0	1	θ_N	0
N+1	0	0	1	0	0

Table III.2 – MD-H parameters of a generic N-R planar robot

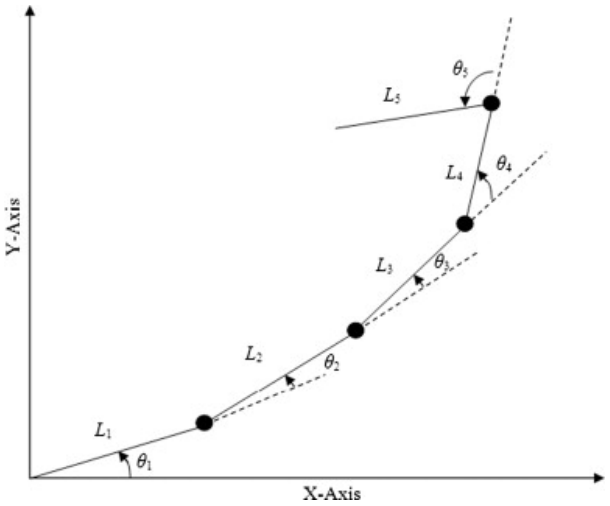


Figure III.1 – General schema of the 5R-planar robot

We recall that the tasks are defined as couples $(\dot{\mathbf{e}}_\mathbf{i}, \mathbf{J_i})$ with decreasing priorities, where $\dot{\mathbf{e}}_\mathbf{i}$ is the desired decrease of the error between actual and desired value, and $\mathbf{J_i}$ the corresponding Jacobian. We choose the classical exponential decrease of the error, thus $\dot{\mathbf{e}}_\mathbf{i} = -\lambda\ \mathbf{e_i}$ where λ is the gain which controls the convergence velocity of the task.

III.1.2 Choice of Control Laws

We are interested in studying the system's behavior when executing simultaneous tasks with specific hierarchy and priority. Thus, referring to Table III.1, only hierarchical control and hybrid control with weighting in the task space can be used. Therefore, the following control laws are chosen to be compared:

- **Orthogonal Projection $\mathbf{P_e}$**:
  Applying equation (II.41) for two tasks gives the relation below:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_\mathbf{1} + (\mathbf{J_2}\ \mathbf{P_e})^+(\dot{\mathbf{e}}_\mathbf{2} - \mathbf{J_2}\ \dot{\mathbf{q}}_\mathbf{1})$$

with $\mathbf{P_e} = \mathbf{I_n} - \mathbf{J_1^+}\ \mathbf{J_1}$

- **Directional Projection $\mathbf{P_z}$**:
  Using the generic relation (II.44) for task sequencing with directional projection, the following equation is obtained:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_1 + \mathbf{P_z}\ \mathbf{J_2^+}\ \dot{\mathbf{e}}_2$$

  with $\mathbf{P_z}$ the directional projection operator defined in (II.25).

- **Minimum Norm Solution $\mathbf{P}_\eta$**:
  Using the method of minimum norm solution presented in case of several tasks in equation (II.45), and the switching strategy (II.32) gives:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_1 + (\mathbf{J_2}\ \mathbf{P}_\alpha)^+\ (\dot{\mathbf{e}}_2 - \mathbf{J_2}\ \dot{\mathbf{q}}_1)$$

- **Hybrid Control** with variable weighting matrix $\mathbf{H}$:
  The control law, defined by [KC11a], is given in (II.9):

$$\dot{\mathbf{q}} = (\mathbf{H}\ \mathbf{J_{ext}})^+\ \mathbf{H}\ \dot{\mathbf{e}}_{\mathbf{ext}}$$

  with $\mathbf{e_{ext}} = [\mathbf{e}_1\ ;\ \mathbf{e}_2]$ and $\mathbf{J_{ext}} = [\mathbf{J}_1\ ;\ \mathbf{J}_2]$ are the extended task error and the extended Jacobian respectively, and $\mathbf{H} = \mathrm{diag}(h_i)$ the weighting matrix defined in (II.10).

  In simulations part, we consider one task $\mathbf{T}_1$ and one constraint $\mathbf{T}_2$. The constraint is activated only when arriving out of a square domain from the desired point.

The control schema that will be used to execute the two desired tasks, with the defined task sequencing and redundancy resolution techniques, is represented in Fig. III.2 below:

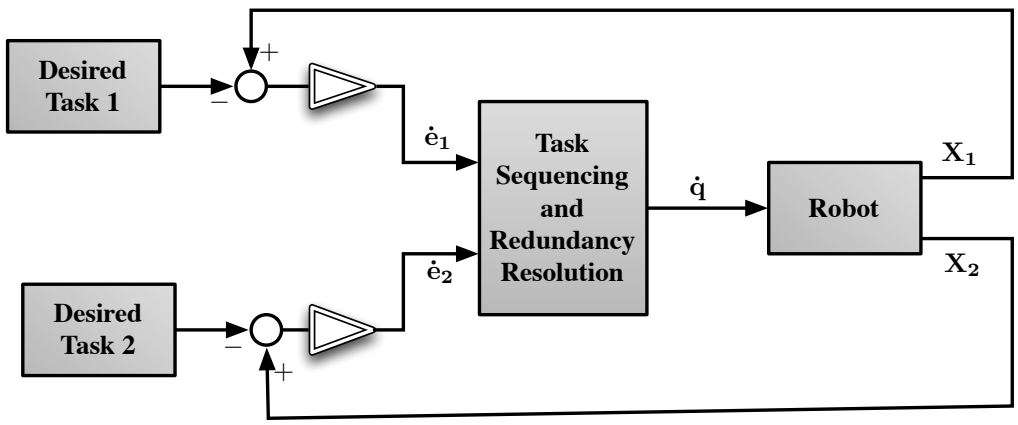


Figure III.2 – General representation of the control schema

III.2 Comparison Criteria

For the already defined structures, the following criteria are considered to compare the robot's behavior for the different control techniques.

III.2.1 Control Points Trajectory

The trajectory of the control points (CP) is studied to verify if the desired exponential decrease, of the main and secondary tasks error, is respected (linear trajectory in the plane). Furthermore, for each task, the variation of the total error norm is studied during tasks execution.

III.2.2 Rank of the Projector/Weighting Matrix

By analyzing the variation of the projection matrix rank (noted **rank**(**P**)), we can identify the number of free DOF, which can be used to execute the secondary task(s). In fact, if the number of DOF constrained by the main task increases, the projector's rank decreases and less DOF could be used by the robot to execute secondary task(s).

In case of Hybrid control, the rank of the weighting matrix (noted **rank**(**H**)) represents the number of non-zero weights. When the constraint is respected the corresponding weights are equal to zero and the matrix drops in rank. On the other hand, if the constraint is violated, the control law tries to move the corresponding CP back to the desired domain by given non-zero values for the corresponding weights, thus the rank of the weighting matrix increases.

III.2.3 Time and Order of Convergence

For the same gain values for the error regulation, the convergence time ($\mathbf{t_{conv}}$) is used to compare the speed of convergence of the different techniques, and the order of convergence of the applied main/secondary task(s).

It corresponds to the moment at which the task error becomes smaller than a desired threshold (10^{-7} in next simulations).

III.2.4 Performance Indices

One of the quantitative methods which are used to evaluate the robot's performance is the measure of the robot's manipulability in all directions, which is one of the key performance indices of redundant robotic manipulator's overall dexterity.

Therefore, several manipulability indices have been used in studying the redundant manipulator's kinematics: [SC82] proposed a condition number, [Yos84] proposed an overall dexterity index of redundant robotic manipulators, and defined the manipulability ellipsoid [Yos90] which is used to calculate the manipulability measure w_m as:

$$w_m = \sqrt{\det(\mathbf{J}\mathbf{J}^\top)} = \sigma_1.\sigma_2...\sigma_m$$

where the scalar values $\sigma_1, \sigma_2, ...\sigma_m$ are the 'm' singular values of $\mathbf{J}$ ordered from maximum to minimum.

[Yos90] defined also the condition number which is defined as the ratio of the maximum and minimum singular values:

$$w_{cond} = \frac{\sigma_1}{\sigma_m} \geq 1$$

These two indices are merged in one parameter of singularity (PS) to describe the overall kinematic behavior of the robot, it is defined as:

$$W_{ps} = \sqrt{\frac{w_{cond}}{w_m}} = \sqrt{\frac{1}{\sigma_2 ... \sigma_m^2}} \tag{III.1}$$

The greater the W_{ps} is, the more the conditioning is, or the lower the manipulability is, thus it approaches the singularity.

In addition to that, [KB87] proposed the minimum singular value index which is used to design and control the redundant robotic manipulator, [Gos90] defined a global optimization performance index for designing the redundant robotic manipulator's kinematics, [GH02] proposed the global performance index for parallel robots which includes the first-order and second-order influence coefficient matrix (Jacobian and Hessian matrices) [GLW06], and finally [SLCY05] proposed the global performance fluctuated index and suggested using it as an objective function to optimize the dimensions of robotic manipulators.

In the comparison study, the parameter of singularity W_{ps} will be used. In case of projection methods, two performance indices (W_{ps1} and W_{ps2}) for the main and secondary tasks respectively will be evaluated; and in case of hybrid control, only the parameter of singularity which corresponds to the extended Jacobian $\mathbf{J_{ext}}$ will be considered.

III.2.5 Overall Kinetic Energy

In some cases, the convergence time is not the most important index, but the consumed energy is considered as the primal criteria [Ata07]. This can be the case where the amount of available energy is scarce. Many applications, e.g rest to rest maneuvering, require a motion control system to move an object (load) starting from rest to a specified final angle/distance in a desired time duration, and to return the load to rest at the time end.

In general, to perform the desired task(s), a control law can be designed using an optimal performance index which evaluates the kinetic energy E_c used throughout task(s) execution:

$$E_c = \sum_{i=1}^{n} \frac{1}{2} \dot{\mathbf{q}}_\mathbf{i}^\top \dot{\mathbf{q}}_\mathbf{i} \tag{III.2}$$

where $\dot{\mathbf{q}}_\mathbf{i}$ is the instantaneous velocity of the i^{th} joint.

Many researchers applied the minimum energy consumption criterion for optimal trajectory selection of robot manipulators: [CS92] proposed a technique based on studying the

geometric work of a regular open chained manipulator and applied it to the path planning for minimum energy consumption. [Ma95] presented a real-time algorithm to generate quasi-minimum energy point-to-point control of robotic manipulators.

Furthermore, [FS96] presented an iterative dynamic programming technique to plan minimum-energy consumption trajectories for robotic manipulators considering joint actuator and time constraints. Finally, [SS98] considered a solution to the problem of moving a robot manipulator with minimum cost along a specified geometric path. The optimum traveling time as well as the minimum mechanical energy of the actuator are considered together to build a multi-objective function and the obtained results depend on the associated weighting factor.

III.2.6 Case of Unreachable and Incompatible Tasks

The robustness of a control law depends on its behavior and reaction to particular and irregular configurations. In fact, due to imprecisions in the system's modeling, or due to faulty inputs, the secondary task could be requested to reach a desired position which is outside of the system's workspace.

Therefore, the behavior of the system is evaluated in case the desired pose for the secondary task is unreachable, its effect on the main task and on the stability of the system is also studied.

The feasibility of applying such case in the control approach allows, for example, the possibility to consider a secondary criteria of moving the control point in a desired direction (which is defined by a pose that could be out of the application's domain of the robot).

We excpect that, for a given desired pose of the main task, if the secondary task (T_2) is unreachable, the higher priority task (T_1) is regulated to the desired value without any perturbation.

Furthermore, the reaction of the system to a faulty control of the same control point for two different desired positions is an interesting case to be studied. Therefore, the behavior of the system is also studied when two incompatible tasks are given to move the same control point to two different position/orientation. We expect that the system remains stable, and the task with higher priority (T_1) is executed.

Such case occurs due to conflicts between the defined constraints or the desired performance criteria of the system. Taking into account a large number of constraints and criteria that are necessary to be considered in the control law, could lead to the appearance of motions that force the control point to move in opposite directions. Therefore studying the system's behavior in case of incompatible tasks is necessary in the evaluation of the capability of the different redundancy resolution techniques .

III.3 Simulation on Planar Robots

The studied cases are summarized below in Table III.3. The simulation results are presented first in case of sufficient DOF for all tasks ($l < r$) with the 7R planar robot using orthogonal projection $\mathbf{P_e}$, directional projection $\mathbf{P_z}$, minimum norm solution $\mathbf{P}_\eta$ and hybrid control $\mathbf{H}$.

Then, we present simulation results in case of kinematically determinate system with the 5R planar robot ($l = r$) using the same control types, and finally in case of kinematically over-specified system (insufficient DOF) with the 4R planar robot ($l > r$). Moreover, these results are summarized in several tables for the three systems (7R, 5R and 4R planar robots), and compared at the end of this section.

Case Study	**Indeterminate System**	**Determinate System**	**Over-Specified System**	**Unreachable Secondary Task**	**Incompatible Tasks**
Robot	7R planar	5R planar	4R planar	7R planar	7R planar
Number of joints n	7	5	4	7	7
Degree of redundancy r	4	2	1	4	4
1^{st} **CP**	EF	EF	EF	EF	EF
2^{nd} **CP**	4^{th} joint	3^{rd} joint	2^{nd} joint	4^{th} joint	EF
Initial Condition $\mathbf{q}_i$	$\begin{bmatrix} -0.5 \\ -1 \\ -0.8 \\ 0.5 \\ 1.7 \\ 1.4 \\ -2.2 \end{bmatrix}$	$\begin{bmatrix} 3\pi/4 \\ \pi/6 \\ \pi/12 \\ -\pi/4 \\ \pi/6 \end{bmatrix}$	$\begin{bmatrix} \pi/4 \\ \pi/3 \\ \pi/12 \\ -\pi/4 \end{bmatrix}$	$\begin{bmatrix} \pi/3 \\ \pi/4 \\ \pi/6 \\ 0 \\ -\pi/4 \\ -\pi/6 \\ -\pi/2 \end{bmatrix}$	$\begin{bmatrix} \pi/3 \\ \pi/4 \\ \pi/6 \\ 0 \\ -\pi/4 \\ -\pi/6 \\ -\pi/2 \end{bmatrix}$
Desired Poses	$\mathbf{T}_1$ $[4; 2; \frac{\pi}{2}]$ $\mathbf{T}_2$ $[2; 1]$	$\mathbf{T}_1$ $[2.5; 1; \frac{\pi}{2}]$ $\mathbf{T}_2$ $[1; 1.5]$	$\mathbf{T}_1$ $[2; 1; \frac{\pi}{2}]$ $\mathbf{T}_2$ $[1; 0.9]$	$\mathbf{T}_1$ $[4; 2; \frac{\pi}{2}]$ $\mathbf{T}_2$ $[2; 1]$	$\mathbf{T}_1$ $[-4; 2; \frac{\pi}{2}]$ $\mathbf{T}_2$ $[2; 1]$

Table III.3 – Summary of the studied cases on the planar robots

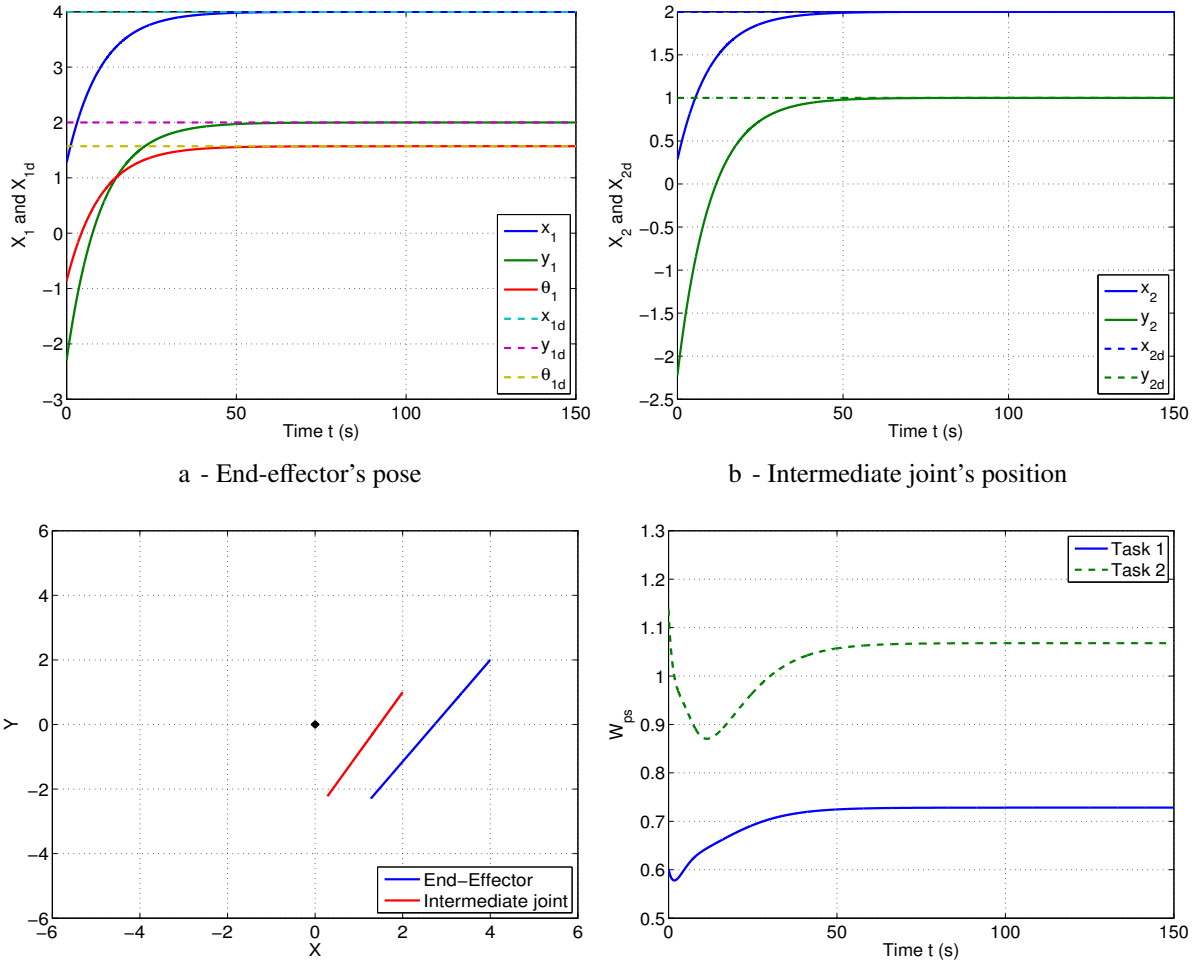
III.3.1 Kinematically Indeterminate System

III.3.1.1 Orthogonal projection

In case of feasible tasks, i.e. kinematically indeterminate system, simulation results show that, using the orthogonal projection, the two tasks are regulated to the desired pose in an acceptable convergence time (Fig. III.3a-III.3b), and the trajectory of the control points is linear (Fig. III.3c), thus this method respects the desired exponential decrease of the error.

Rank($\mathbf{P_e}$), the orthogonal projector's rank, is always constant during tasks execution, thus an invariable number of DOF is used by each task throughout the simulation. Furthermore, the variation of the performance index W_{ps} is plotted during tasks execution in Fig. III.3d. In fact, a higher value indicates that the task approaches more to system's singularity, however, an acceptable behavior is noted for this control law.

Finally, acceptable values of joint velocities is noted, which indicates that the motion given by this projection method is feasible in case of kinematically indeterminate system.



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

d - Performance index variation

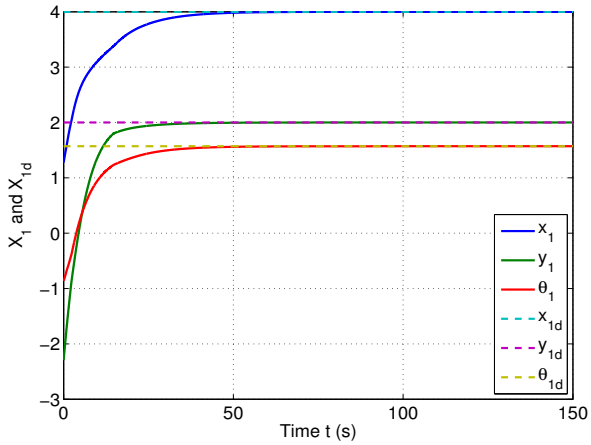
Figure III.3 – Simulation results on a 7R Planar robot with orthogonal projection $\mathbf{P_e}$

III.3.1.2 Directional projection

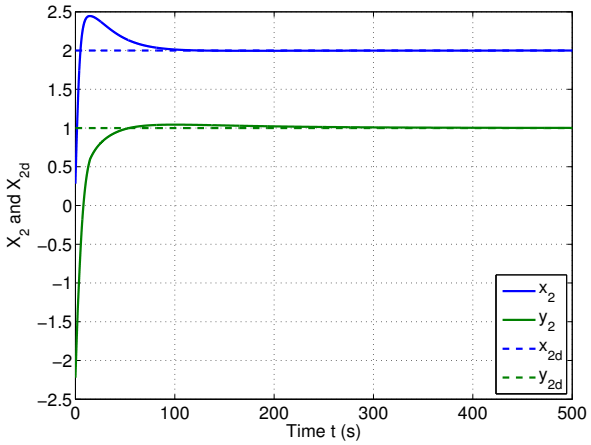
Simulation results (Fig. III.4) show a non exponential decrease of the total tasks error, and a non linear trajectory of the CP (Fig. III.4c). This behavior is expected due to the mutual effect between the tasks on the error regulation.

Rank($\mathbf{P_z}$) is always equal or higher than that of the classical projection, it varies between 7 and 4. In fact, this method allows the secondary task to use the free DOF once they move in the same direction as the main task. Thus, the variation of the rank depends on the number of components of the secondary tasks that does not impact the regulation of the main task.

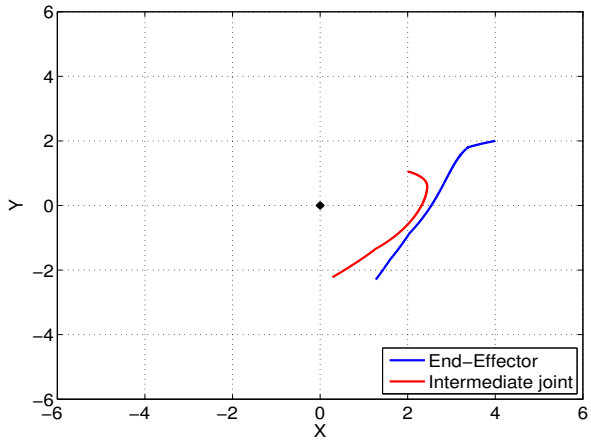
The oscillations of the projector's rank between 6 and 5, and then between 5 and 4 (between the 14 sec and 28 sec) is due to the transition between the conditions in the equation (II.25) of the directional projection operator. In fact, during this period, the main task totally uses between one and two DOF to be completed while helping the execution of the secondary task. After that, the main task is regulated to the desired value at 28 sec, and the secondary task uses only the four remaining DOF.



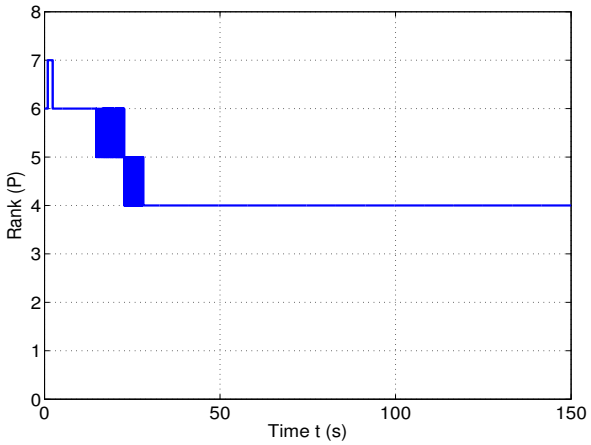
a - End-effector's pose



b - Intermediate joint's position



c - Control points trajectory in plane space



d - Projector's rank

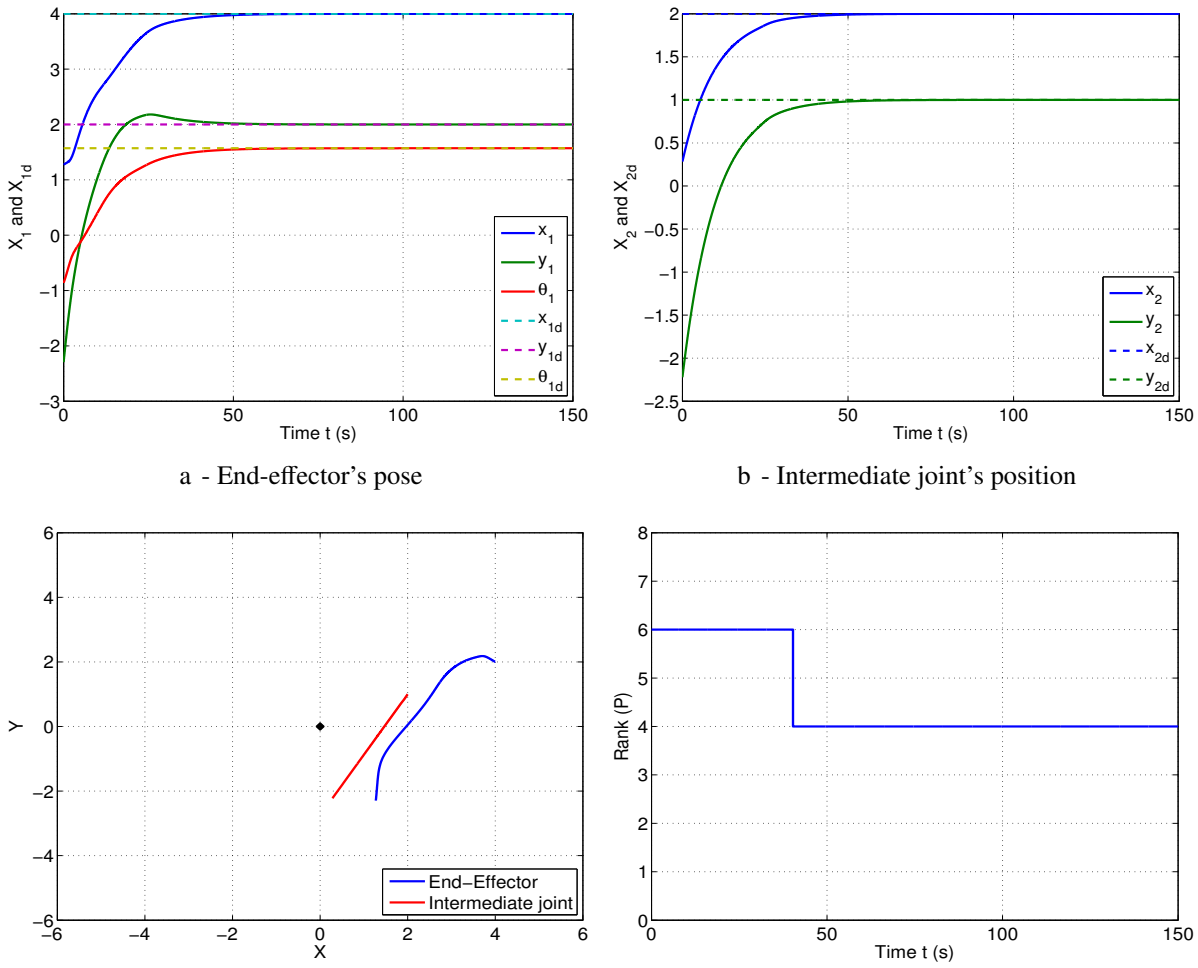
Figure III.4 – Simulation results on a 7R Planar robot with directional projection $\mathbf{P_z}$

Although the main task is executed faster than in the classical method case (as theoretically considered by this approach), the convergence time of the secondary tasks is unacceptable: the error didn't reach the desired threshold after more than 500 s (Fig. III.4b).

III.3.1.3 Minimum norm solution

For the method of minimum norm solution, simulation results show a non exponential decrease of the total error thus a non linear trajectory of the control points (Fig. III.5c). In fact, this method consists on regulating the error using an exponential decrease of the norm of the total error and not the error components $\mathbf{e_i}$.

The main improvement of this method is the high number of available DOF for the secondary task ($n-1$). Because of the switching strategy used near convergence, the projector's rank (Fig. III.5d) decreases later to that of the classical projection. Moreover, compared to directional projection, this method gives a better behavior of the projector's rank. Finally, the convergence time is acceptable for this control law as in the case of orthogonal projection.



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

d - Projector's rank

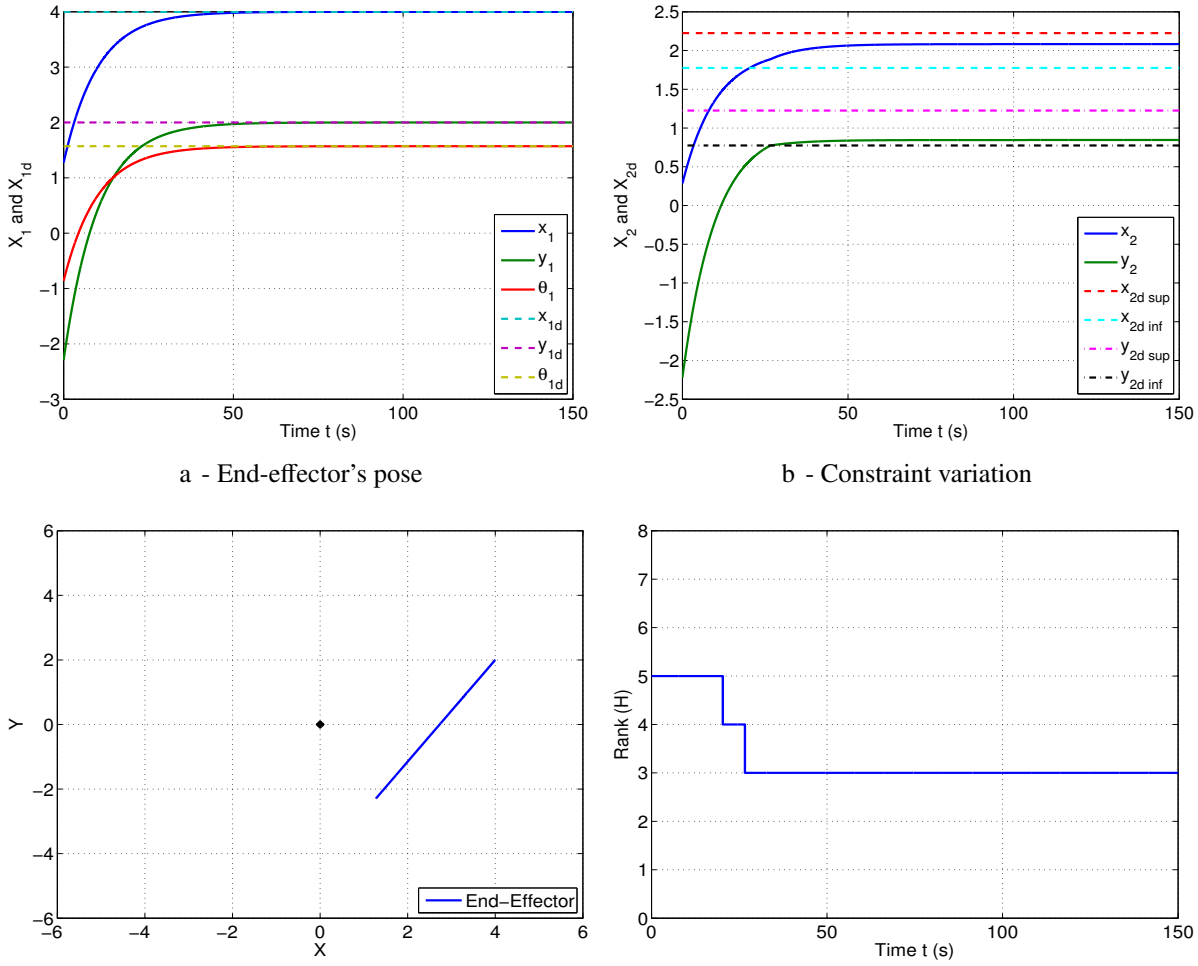
Figure III.5 – Simulation results on a 7R Planar robot with minimum norm solution $\mathbf{P}_\eta$

III.3.1.4 Hybrid control

For the hybrid control with a variable weighting matrix **H**, we used the main task T_1 and we defined one constraint which is controlled to remain in a closed domain around the desired value used above (in task T_2).

Simulation results show that the main task is regulated with a linear trajectory, and the constraint is respected (Fig. III.6a-III.6b-III.6c). During the first 21 sec, the constraint is out of the desired domain; thus, the corresponding weights value are not equal to zero (**rank**(**H**) = 5). Later on, until 26 sec, only the constraint along the x-axis is verified, thus the corresponding weight is equal to zero and the rank drops to 4. During the remaining time, the x and y components of the constraint are respected, thus the weighting matrix rank is equal to 3 (Fig. III.6d).

In this case, the performance index corresponds to the extended Jacobian $\mathbf{J_{ext}}$, and similarly to the other methods an acceptable behavior is noticed. Finally, acceptable values of joint velocities ensure the feasibility of the resultant motion when using the hybrid control on a kinematically indeterminate system.



a - End-effector's pose

b - Constraint variation

c - Control points trajectory in plane space

d - Rank of the weighting matrix

Figure III.6 – Simulation results on a 7R Planar robot with hybrid control **H**

III.3.1.5 Comparison results

The simulation results are summarized in Table III.4 below in case of kinematically indeterminate system. These results show that the directional projection method gives rise to a non linear trajectory of the CP which is closer to singularity, thus a non exponential decrease of the total error. In spite of the small improvement in the convergence time of the first task over the other methods ($\mathbf{t_{conv1}} = 80s$), a high convergence time of the secondary task is noticed ($\mathbf{t_{conv2}} >> 500s$).

On the other hand, the method of minimum norm solution yields to a linear control point trajectory of the main task, but a non linear one for the secondary ones. Thus, a non exponential decrease of the total error. Furthermore, relatively higher joint velocities are required to execute the desired tasks with such method.

For the orthogonal projection and hybrid control methods, the trajectory of the CP is linear, with an exponential decrease of the error. However, for the latter one a variable rank of the weighting matrix is noted, with a relatively high values of the performance index (calculated with the extended Jacobian).

In conclusion, the different control methods give acceptable and stable behaviors when sufficient DOF are available to execute all the desired tasks. However, the choice of the appropriate method to be used depends on the desired performance of the application (linear trajectory, low kinetic energy, exact execution of secondary tasks, ...).

	Trajectory		**Total Error Variation**	**Projector's Rank**	**Convergence Time**		$\mathbf{W_{ps}}$	$\mathbf{E_c}$
	1st CP	**2nd CP**			$\mathbf{t_{conv1}}$(s)	$\mathbf{t_{conv2}}$(s)		
$\mathbf{P_e}$	Linear	Linear	Exponential	4	86	82	0.712	10.86
				Fixed	Acceptable Time		1.036	
$\mathbf{P_z}$	Non Linear	Non Linear	Non Exponential	$7 \leftrightarrow 4$	80	>> 500	0.724	11.95
				Variable	Unacceptable Time		1.133	
$\mathbf{P_\eta}$	Linear	Non Linear	Non Exponential	$6 \rightarrow 4$	86	80	0.710	12.47
				Variable	Acceptable Time		1.039	
H	Linear	-	Exponential	$5 \rightarrow 4 \rightarrow 3$	87	–	1.563	10.82
				Variable	Acceptable Time			

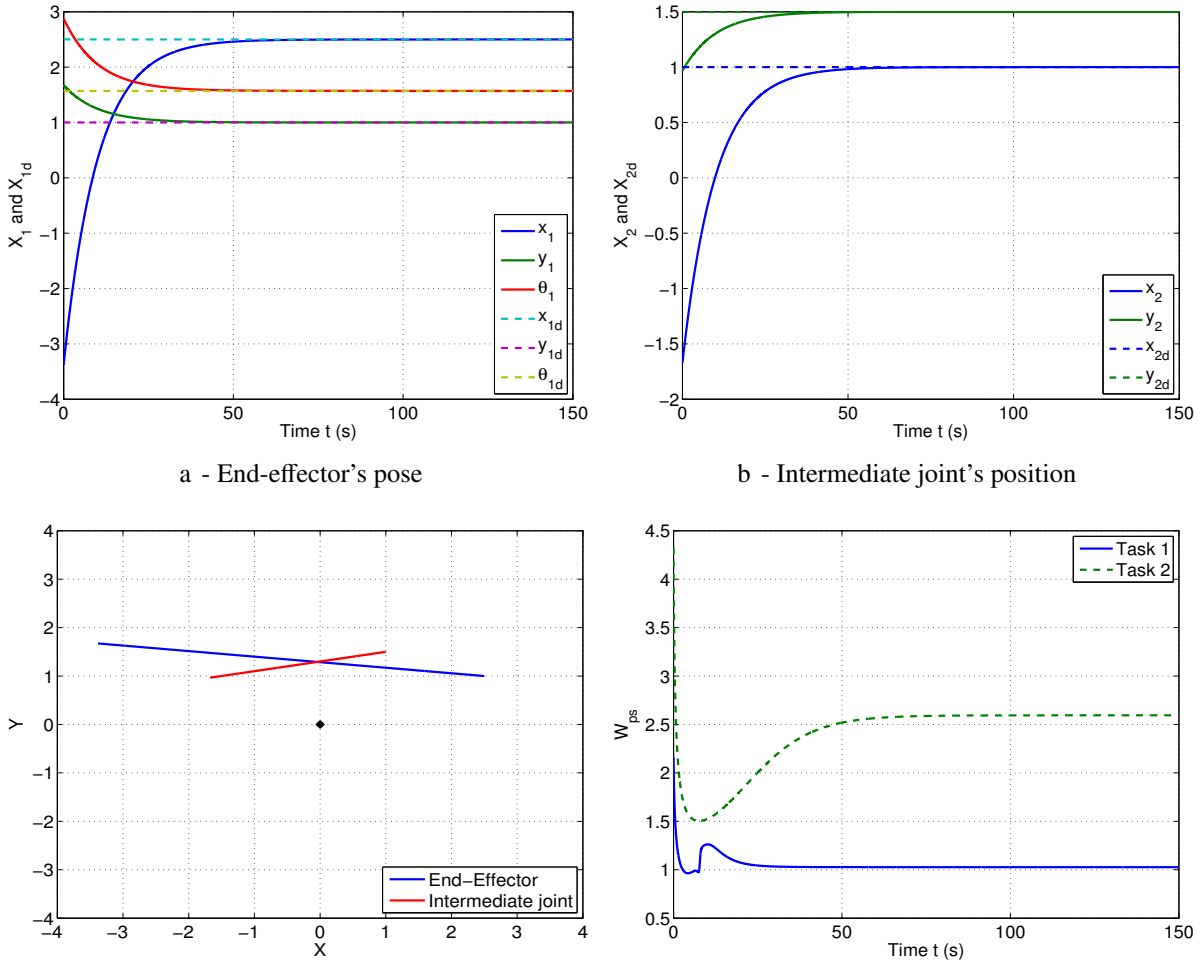
Table III.4 – Simulation results of the 7R planar robot (kinematically indeterminate system)

III.3.2 Kinematically Determinate System

In case of kinematically determinate system, an exact number of DOF is available to execute the two desired tasks using a 5R planar robot. On this part, the different control laws will be first applied on this robot and then compared between them. Results will be summarized in Table III.5 in the last paragraph.

III.3.2.1 Orthogonal projection

The comparison of the orthogonal projection behavior in this case (Fig. III.7) with respect to the case of indeterminate system (Fig. III.3) don't give a lot of differences except the relatively higher joint velocities for tasks execution, and consequently the higher values of the performance index (nearest to singularities) as shown in Fig. III.7d. Otherwise, the linear control point's trajectory (Fig. III.7c) and the fixed projector's rank ($\mathbf{Rank}(\mathbf{P_e}) = 2$) are the same as in the previous case.



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

d - Performance index

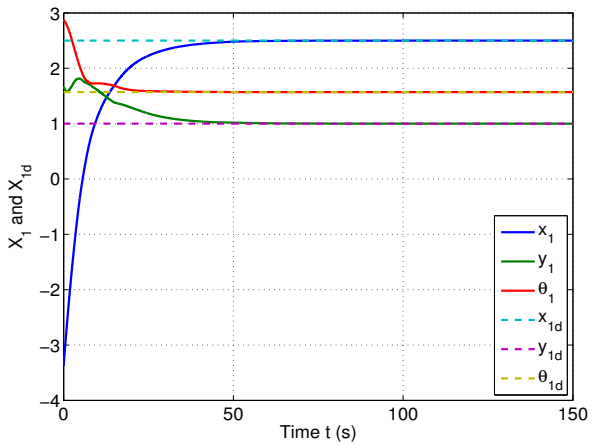
Figure III.7 – Simulation results on a 5R Planar robot with orthogonal projection $\mathbf{P_e}$

III.3.2.2 Directional projection

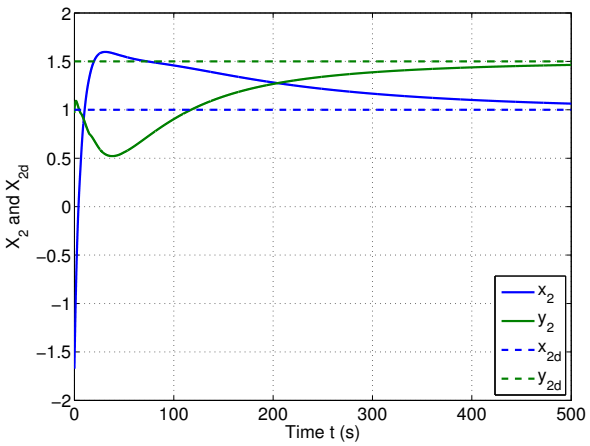
For the directional projection method (Fig. III.8), simulation results show non exponential decrease of the total tasks error, and a non linear trajectory of the CP as in the previous case (Fig. III.8c).

Furthermore, the $\mathbf{P_z}$ projector's rank is always equal or higher than that of the classical projection (where only 2 DOF are available). The rank varies between 5 and 2, thus more DOF are free to execute the secondary task than in the case of orthogonal projection. In fact, each time the secondary task components go in the same direction as that which corresponds to the main task regulation, the projector's rank increases to a higher value (Fig. III.8d).

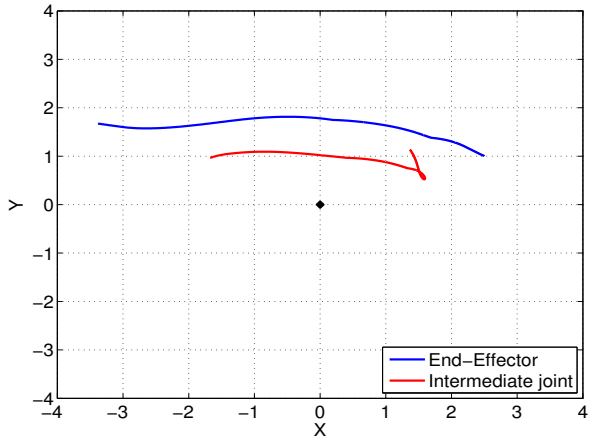
Finally, an overall acceptable behavior is noted for this case except the high value of the convergence time of the secondary task ($\mathbf{t_{conv2}}$ >> 500 sec).



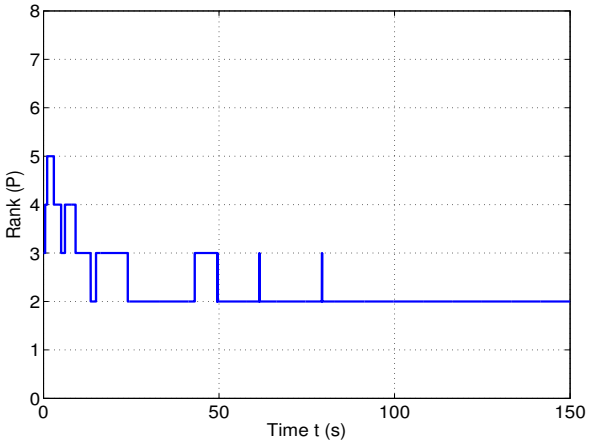
a - End-effector's pose



b - Intermediate joint's position



c - Control points trajectory in plane space



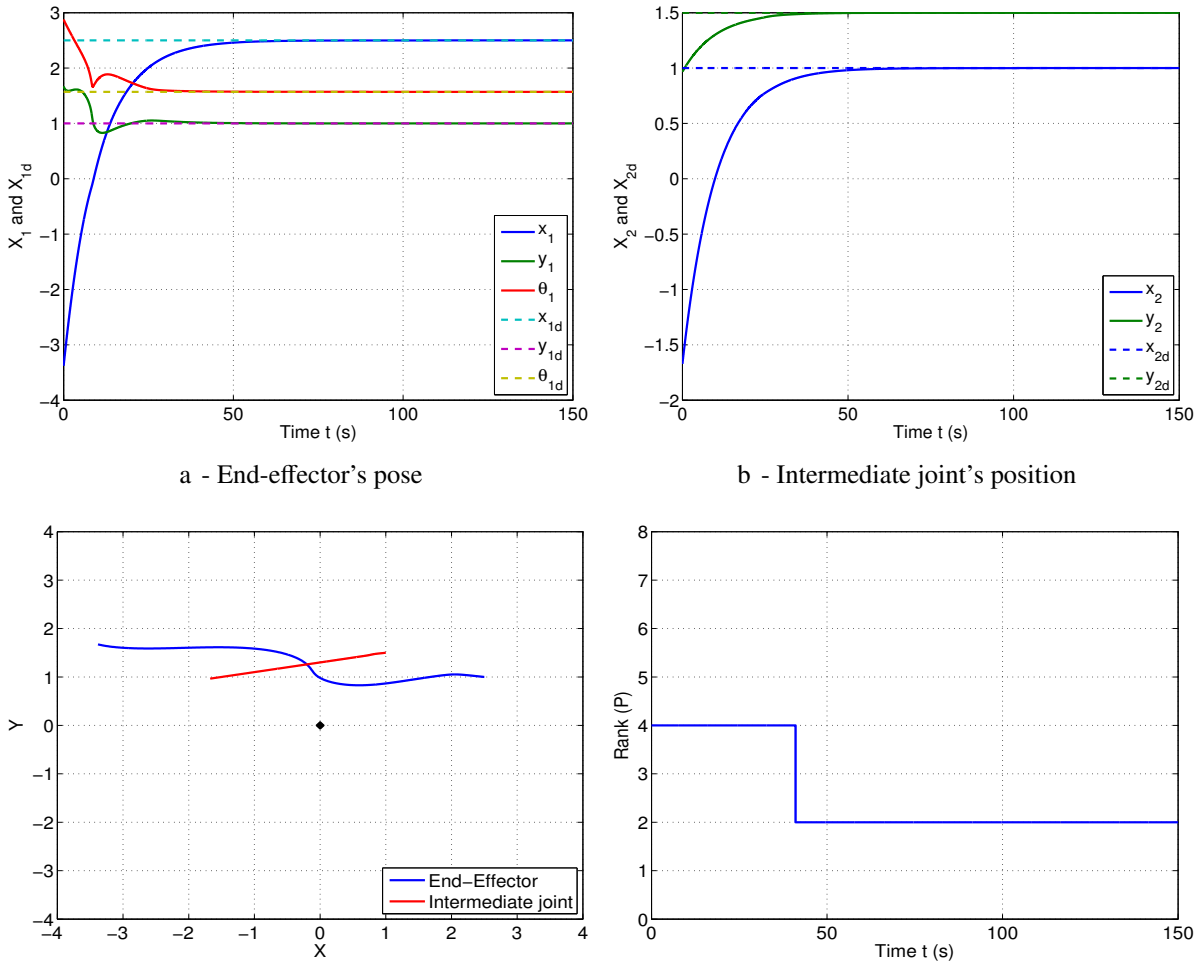
d - Projector's rank

Figure III.8 – Simulation results on a 5R Planar robot with directional projection $\mathbf{P_z}$

III.3.2.3 Minimum norm solution

For the minimum norm solution $\mathbf{P}_\eta$, simulation results (Fig. III.9) show a non linear trajectory of the control points and a non exponential decrease of the main task error, due to the control law that regulate the overall task error norm, in contrast to the classical one which regulates the components of the task error.

The projector's rank decreases from 4 (minimum norm solution) to 2 (classical projection) due to the used switching strategy (Fig. III.9d). We note also a higher performance index values, which indicates that the system approaches to a singular configuration.



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

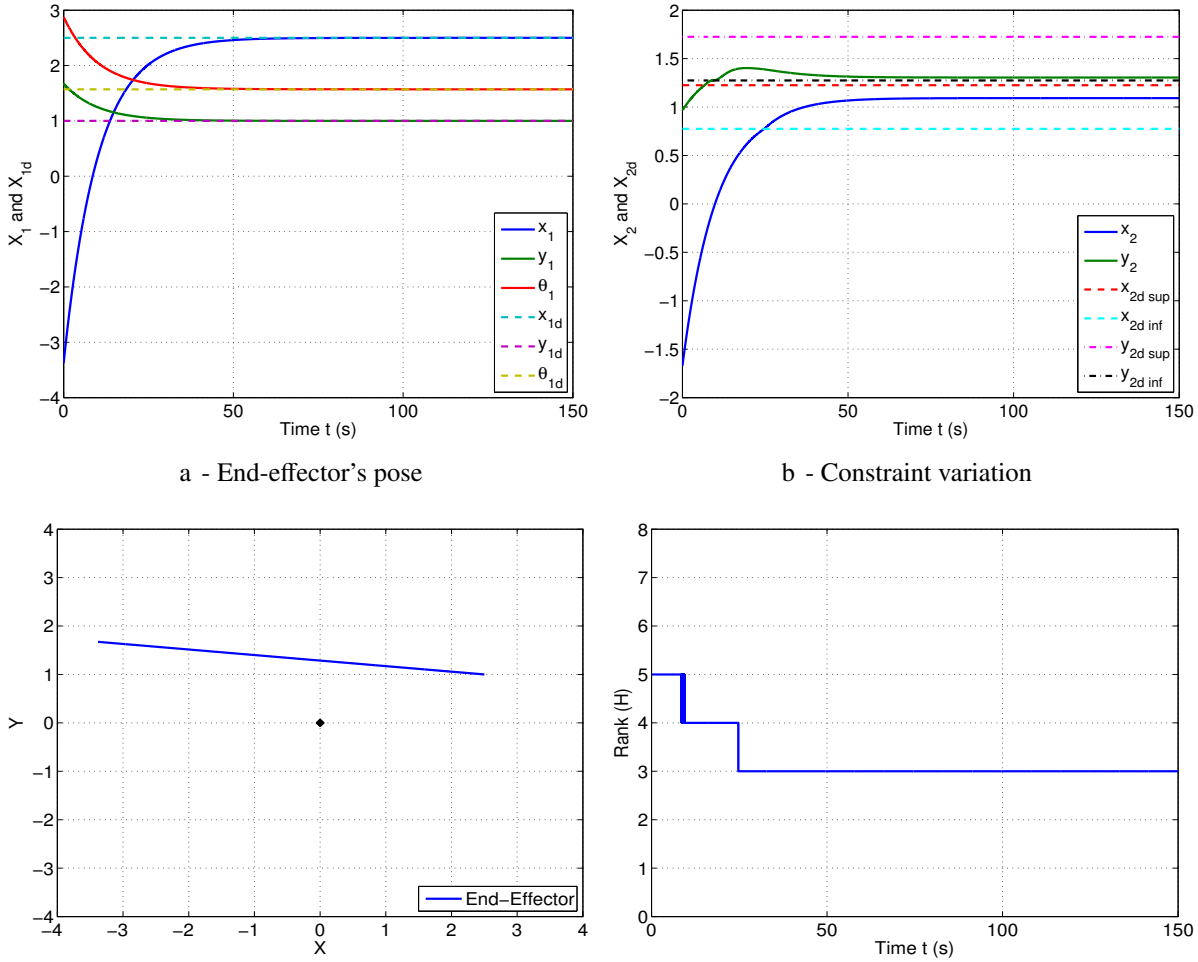
d - Projector's rank

Figure III.9 – Simulation results on a 5R Planar robot with minimum norm solution $\mathbf{P}_\eta$

III.3.2.4 Hybrid control

Using the hybrid control with a variable weighting matrix (Fig. III.10), we get an exponential decrease of the error and a linear trajectory of the control point as expected (Fig. III.10c). The x and y components of the constraint remains in the desired domain, and the same evolution of the weighting matrix rank is noticed as in the previous case (Fig. III.10d).

However, higher joint velocities with respect to the case of indeterminate system are noticed, and the performance index gives higher values.



a - End-effector's pose

b - Constraint variation

c - Control points trajectory in plane space

d - Rank of the weighting matrix

Figure III.10 – Simulation results on a 5R Planar robot with hybrid control **H**

III.3.2.5 Comparaison results

The simulation results in case of determinate system are summarized below in Table (III.5). The overall results show similar behaviors in case of determinate system ($l = r$) as in the previous case of indeterminate one ($l < r$) except for the higher joint velocities and the higher values of W_{ps} which indicates, as expected, that the system approaches more to singularities due to the lack in the free DOF.

Furthermore, the variation of the overall kinetic energy for tasks execution shows an increase of the required energy to execute the desired tasks in case of the orthogonal projection and hybrid control.

	Trajectory		**Total Error Variation**	**Projector's Rank**	**Convergence Time**		**W_{ps}**	**E_c**
	1st CP	**2nd CP**			**t_{conv1}(s)**	**t_{conv2}(s)**		
P_e	Linear	Linear	Exponential	2	87	79	1.043	165.0
				Fixed	Acceptable Time		2.042	
P_z	Non Linear	Non Linear	Non Exponential	$5 \leftrightarrow 2$	83	>> 500	1.081	40.85
				Variable	Unacceptable Time		2.099	
P_η	Non Linear	Linear	Non Exponential	$4 \rightarrow 2$	87	80	1.150	42.17
				Variable	Acceptable Time		2.403	
H	Linear	-	Exponential	$5 \rightarrow 4 \rightarrow 3$	87	–	4.190	164.0
				Variable	Acceptable Time			

Table III.5 – Simulation results of the 5R planar robot (kinematically determinate system)

Note that the different performance criteria are deeply studied for the different methods. Moreover, the system's behavior for all the studied cases is tested and validated for several initial and desired configurations. However, in sake of clearness and simplicity, only significant graphs are represented in this dissertation.

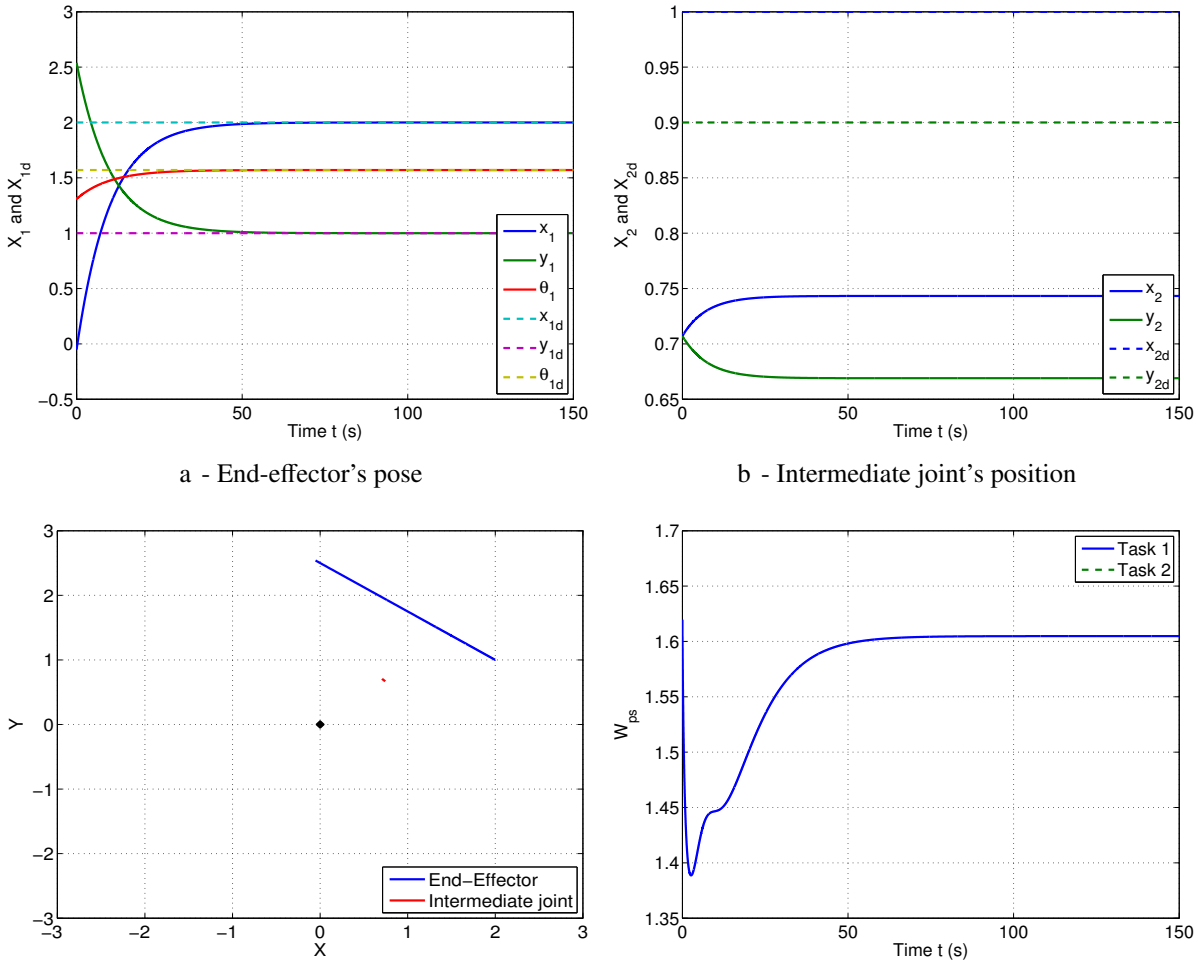
III.3.3 Kinematically Over-specified System

In this part, we study the case of over-specified system (4R planar robot), where the number of available DOF in the system is lower than the required number to execute the desired tasks. Thus, in such particular case, the applied methods should maintain the stability of the system. Furthermore, only a part of the lower priority task should be executed while regulating the main task to its desired pose.

III.3.3.1 Orthogonal projection

For the orthogonal projection method, the main task uses 3 DOF of the 4R planar robot, and 1 DOF is left to the secondary task which requires 2 DOF to control the position of an intermediate point in the plane.

Therefore, the main task is completely regulated to the desired value (Fig. III.11a), and the secondary one converges to the best reachable pose (Fig. III.11b). A linear trajectory of the controlled points is noticed (Fig. III.11c), in addition to, a fixed projector rank ($\mathbf{Rank}(\mathbf{P_e}) = 1$), and an acceptable variation of the performance index (Fig. III.11d).



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

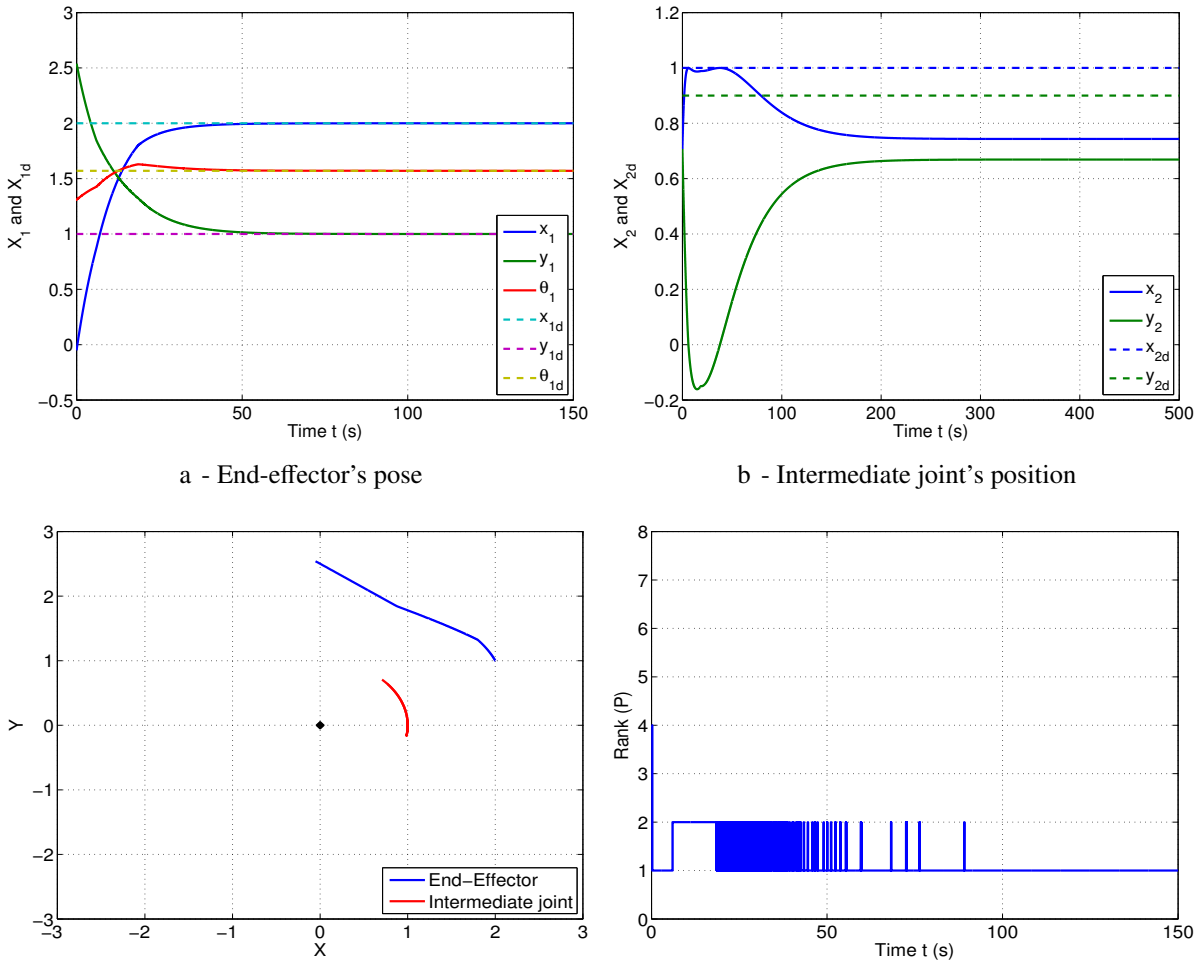
d - Projector's rank

Figure III.11 – Simulation results on a 4R Planar robot with orthogonal projection $\mathbf{P_e}$

III.3.3.2 Directional projection

Simulation results show that the directional projection method can be used even if there is no available DOF for all the tasks. In fact, the main task is regulated to the desired value, and the secondary task converges to an intermediate point which is close to the desired position. In fact, the secondary task helps the main one to be executed faster, thus the second task's error increases at the beginning before converging (Fig. III.12a-III.12b).

Once tasks execution starts, the rank of the projector varies between 2 and 1: the system uses at the beginning two free DOF to execute the secondary task, and then 1 DOF is used. After the regulation of the main task (78 sec), the 2 DOF are constrained, and only 1 DOF is free; thus, the rank of the projector stays constant ($\mathbf{Rank}(\mathbf{P_z}) = 1$). The oscillation of the projector's rank between the two values is due to the transition between the different conditions of the projector's definition (Fig. III.12d).



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

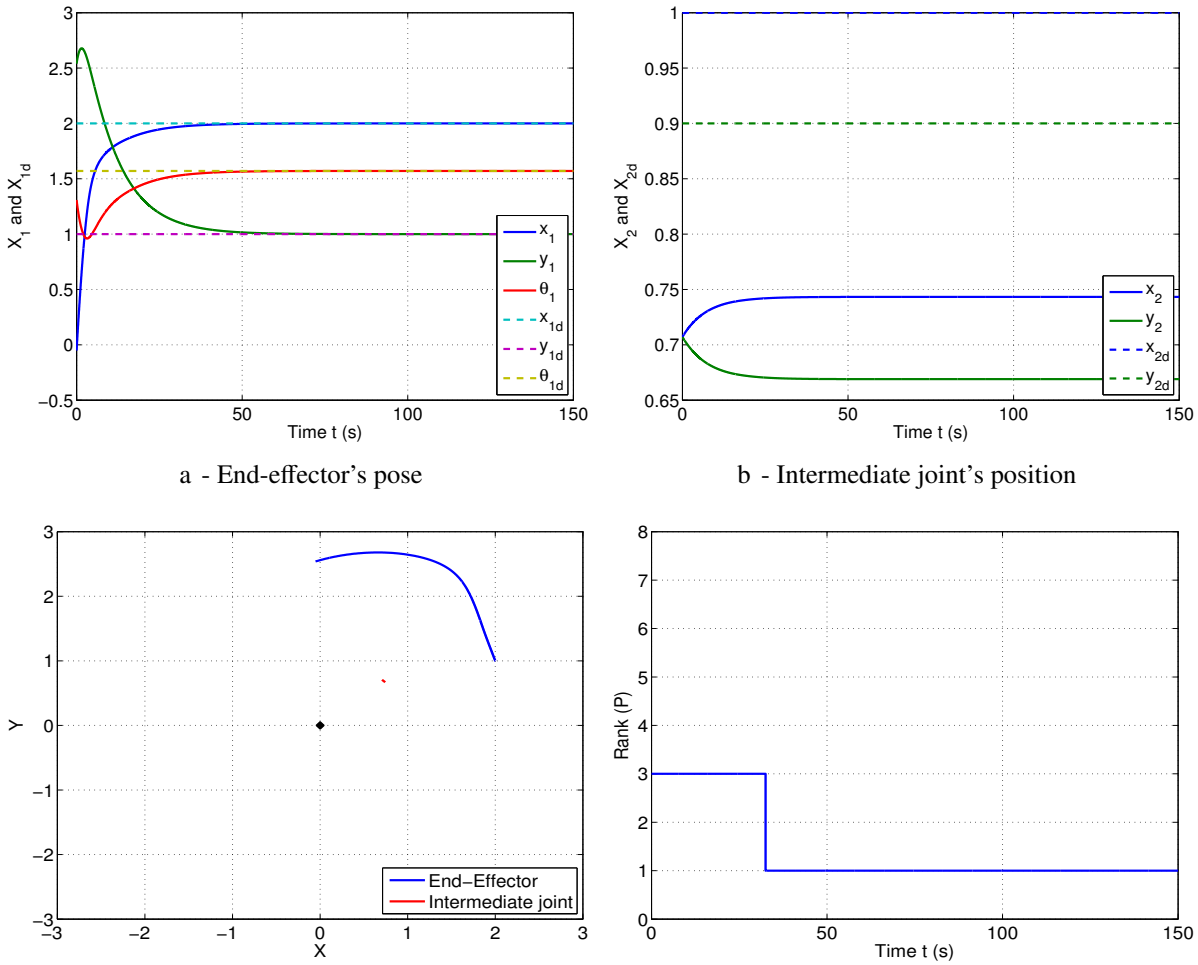
d - Projector's rank

Figure III.12 – Simulation results on a 4R Planar robot with directional projection $\mathbf{P_z}$

III.3.3.3 Minimum norm solution

For the minimum norm solution method $\mathbf{P}_\eta$, simulation results (Fig. III.13) show, as in case of orthogonal projection, a complete regulation of the high priority task and a regulation to the best reachable pose to the secondary one (Fig. III.13a-III.13b).

However, a non linear trajectory of both control points is noticed, thus a non exponential decrease of the total error. The projector's rank passes from 3 to 1 due to the switching to the orthogonal projection near convergence (Fig. III.13d). Finally, the main task is regulated in an acceptable convergence time and with reasonable joint velocities.



a - End-effector's pose

b - Intermediate joint's position

c - Control points trajectory in plane space

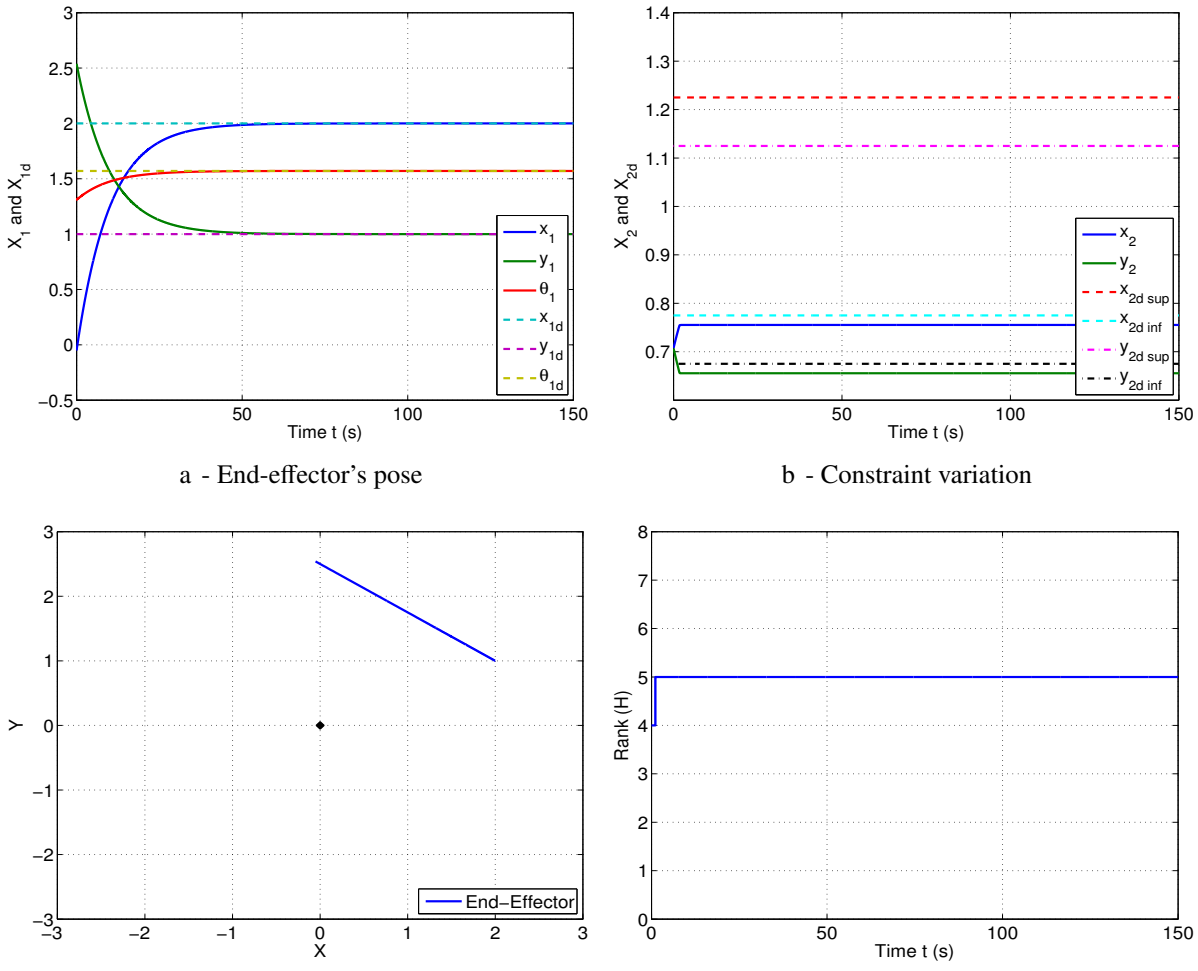
d - Projector's rank

Figure III.13 – Simulation results on a 4R Planar robot with minimum norm solution $\mathbf{P}_\eta$

III.3.3.4 Hybrid control

Using the hybrid control with a variable weighting matrix, an acceptable behavior of the system is noted in Fig. III.14: the main task and the constraint are respected using acceptable joint velocities values.

A linear trajectory of the main task's CP is observed thus the exponential decrease of the error is respected. However, we note an increase in the used kinetic energy thus higher joint velocities than previous methods. Furthermore, the performance index oscillate with high value due to the bad conditioning of the extended Jacobian in this case.



a - End-effector's pose

b - Constraint variation

c - Control points trajectory in plane space

d - Rank of the weighting matrix

Figure III.14 – Simulation results on a 4R Planar robot with hybrid control **H**

III.3.3.5 Comparison results

The simulation results in case of over-specified system are summarized below in Table III.6. As conclusion on the behavior of the different methods when there is an insufficient number of DOF to execute all tasks: the main task is completely regulated in all cases.

However, except the same issue of non linear trajectory of CP, the directional projection shows the best behavior referring to the simultaneous lowest values of the performance index, the necessary kinetic energy and the convergence time of the main task.

	Trajectory		**Total Error Variation**	**Projector's Rank**	**Convergence Time**		$\mathbf{W_{ps}}$	$\mathbf{E_c}$
	1st CP	**2nd CP**			$\mathbf{t_{conv1}}$(s)	$\mathbf{t_{conv2}}$(s)		
$\mathbf{P_e}$	Linear	Linear	Exponential	1	78	-	1.57	23.0
				Fixed	Acceptable Time		-	
$\mathbf{P_z}$	Linear	Non Linear	Non Exponential	$2 \leftrightarrow 1$	78	–	1.50	5.43
				Variable	Acceptable Time		-	
$\mathbf{P_\eta}$	Non Linear	Non Linear	Non Exponential	$3 \rightarrow 1$	78.5	–	1.65	9.63
				Variable	Acceptable Time		-	
H	Linear	-	Exponential	$4 \rightarrow 5$	78.5	–	$\approx 10^5$	20.3
				Variable	Acceptable Time			

Table III.6 – Simulation results of the 4R planar robot (kinematically over-specified system)

III.3.4 Case of Unreachable Secondary Task

After the study of the system's behavior when sufficient or insufficient DOF are available to execute all tasks, we study in this part the impact of applying unreachable secondary task on the behavior of the robot and the stability of higher priority task.

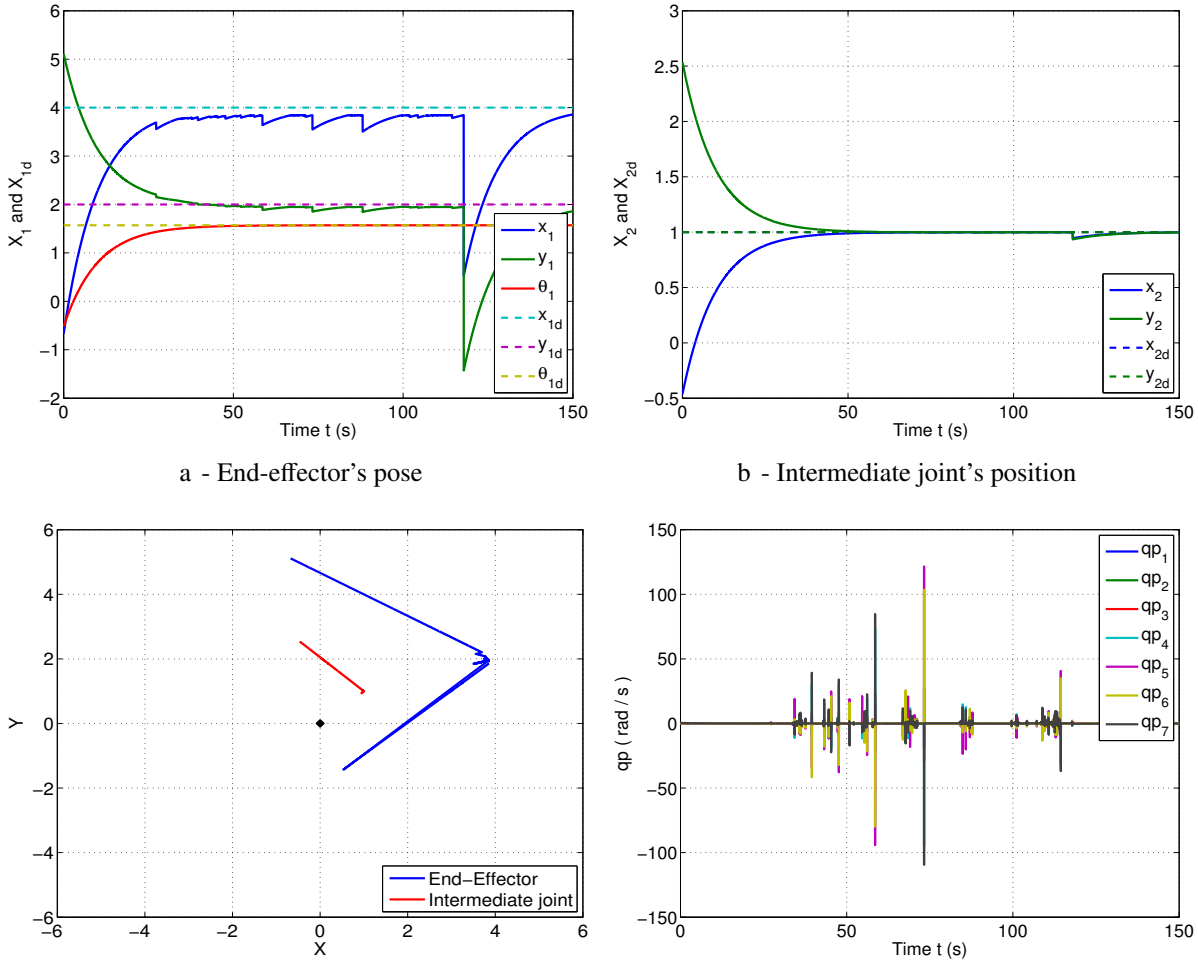
In fact, if the desired position of the lower priority task is unreachable, the main task is expected to converge normally to its desired pose, and the secondary task should be regulated to the best reachable position without disturbing the main task or destabilizing the system.

In fact, as already noted, such case may arise due to faulty inputs or imprecise desired values, which is normal in the experimental implementation of any control law due to noise and perturbation in sensor's measurements. Thus, a 7R planar robot is considered to study the system's behavior in such case; the desired poses of the applied tasks are given in Table III.3.

III.3.4.1 Orthogonal projection

In contrast to the desired behavior, simulation results show that, for the orthogonal projection, the main task is not realized: it converges to an intermediate pose and it is not totally stable (Fig. III.15a). However, the secondary task converges to its desired pose (Fig. III.15b).

Thus, the priority between tasks is not respected when using the orthogonal projection method. Furthermore, high joint velocities are used to execute the desired tasks (Fig. III.15d); thus, the robot is subject to high accelerations and the system is unstable.



a - End-effector's pose

b - Intermediate joint's position

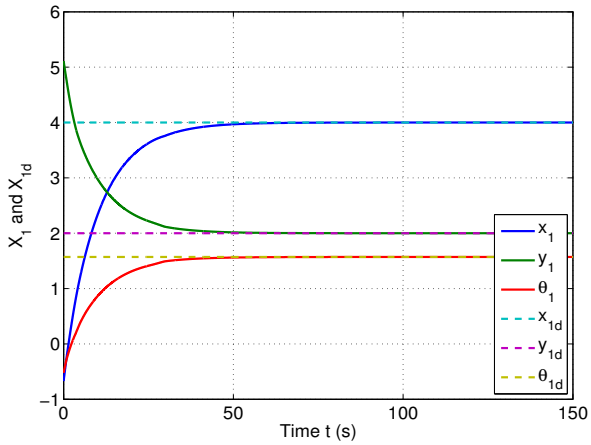
c - Control points trajectory in plane space

d - Joint velocities

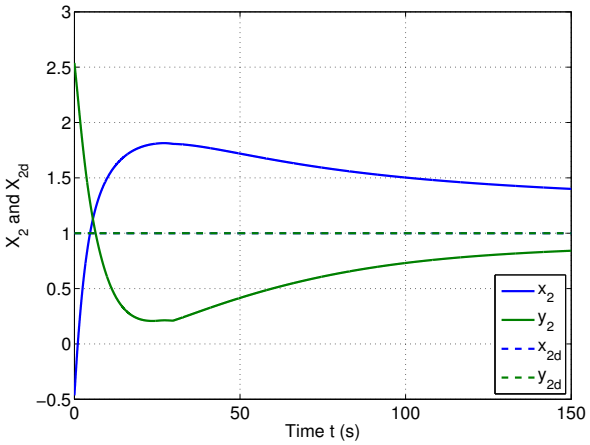
Figure III.15 – Simulation results in case of unreachable task $\mathbf{T}_2$ with orthogonal projection

III.3.4.2 Directional projection

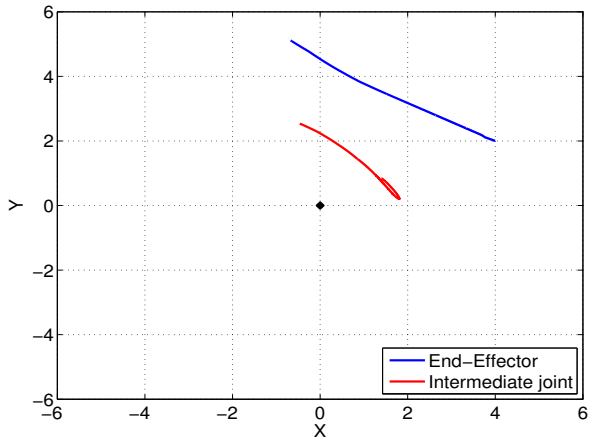
For the directional projection, if the lower priority task is unreachable, the system has an excellent behavior: the main task is regulated normally and the secondary task converges to the best reachable pose (Fig. III.16a-III.16b). Furthermore, acceptable joint velocities are detected, thus tasks motion is feasible (Fig. III.16d). Therefore, for the directional projection, the task priority is respected and this method is useful even if secondary tasks are unreachable.



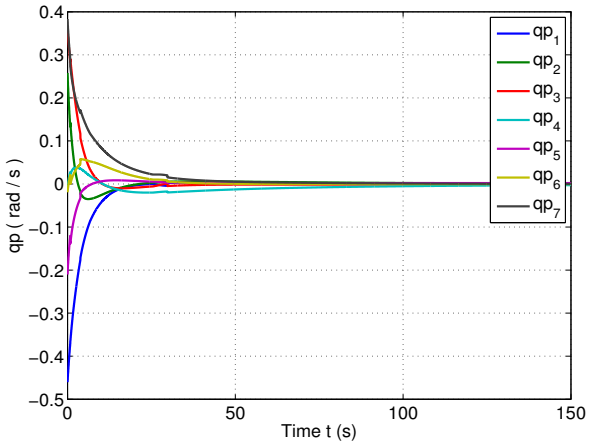
a - End-effector's pose



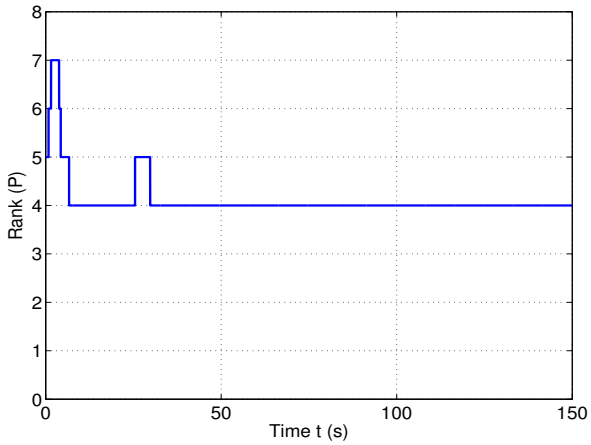
b - Intermediate joint's position



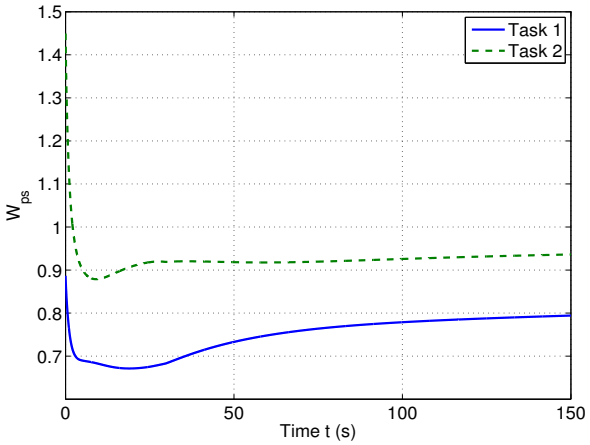
c - Control points trajectory in plane space



d - Joint velocities



e - Projector's rank



f - Performance index

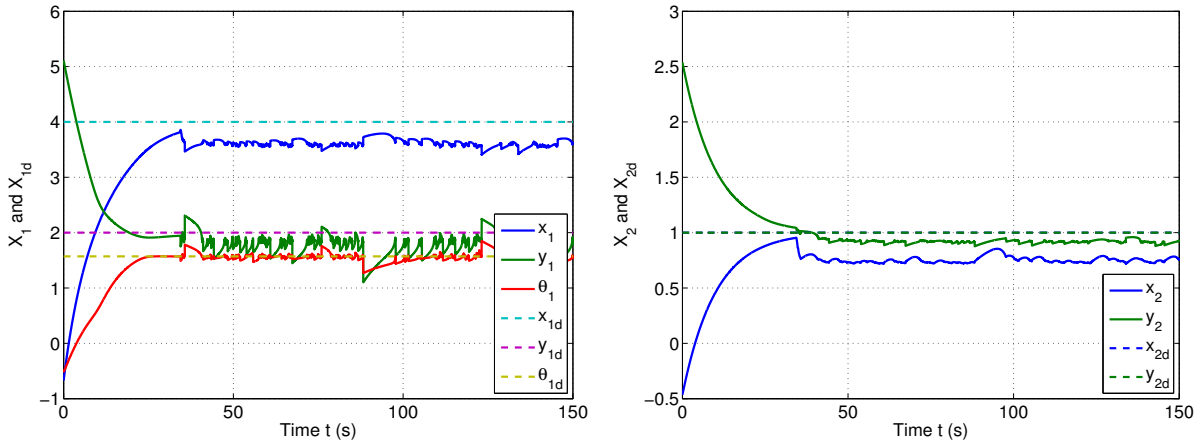
Figure III.16 – Simulation results in case of unreachable task $\mathbf{T}_2$ with directional projection

III.3.4.3 Minimum norm solution

For the minimum norm solution method $\mathbf{P}_\eta$, simulation results show a convergence of the main and secondary tasks to an intermediate pose but with oscillations (Fig. III.17a-III.17b).

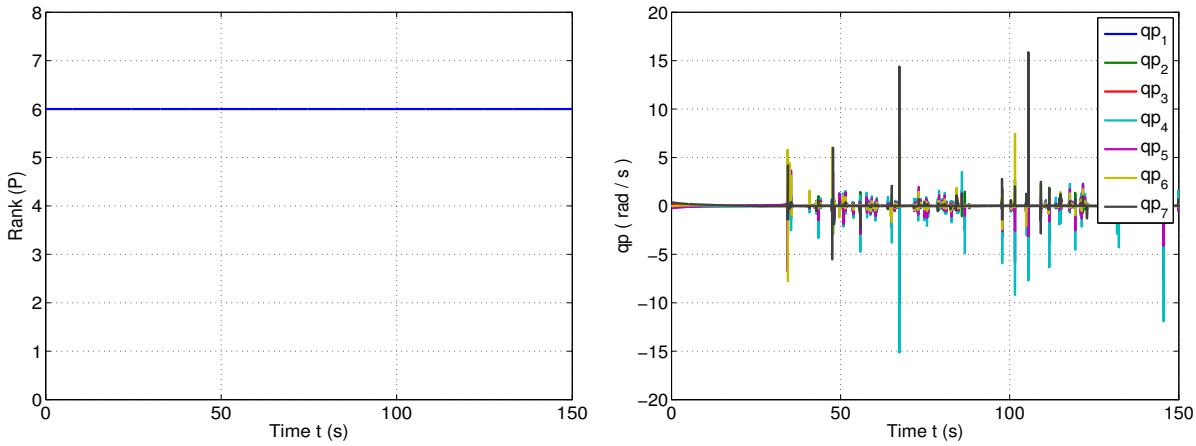
Due to the switching strategy used by this method, the system is blocked in the minimum norm solution part ($\mathbf{Rank}(\mathbf{P}_\eta) = 6$) and does not pass to the orthogonal projection, since it does not pass over the considered threshold. Furthermore, returning to the control law definition in equation (II.32), the different simulation results show that the system has smooth behavior while using the minimum norm solution, but oscillations appears when entering in the switching function (condition $e_0 \leq \|\mathbf{e}\| \leq e_1$).

This case is repeated several times while tuning the corresponding parameters, but the same behavior is noticed: oscillations and instability in the main and secondary tasks, high joint velocities (Fig. III.17d); thus, the priority between tasks is not respected, and this method is not suitable in case of unreachable secondary tasks.



a - End-effector's pose

b - Intermediate joint's position



c - Projector's rank

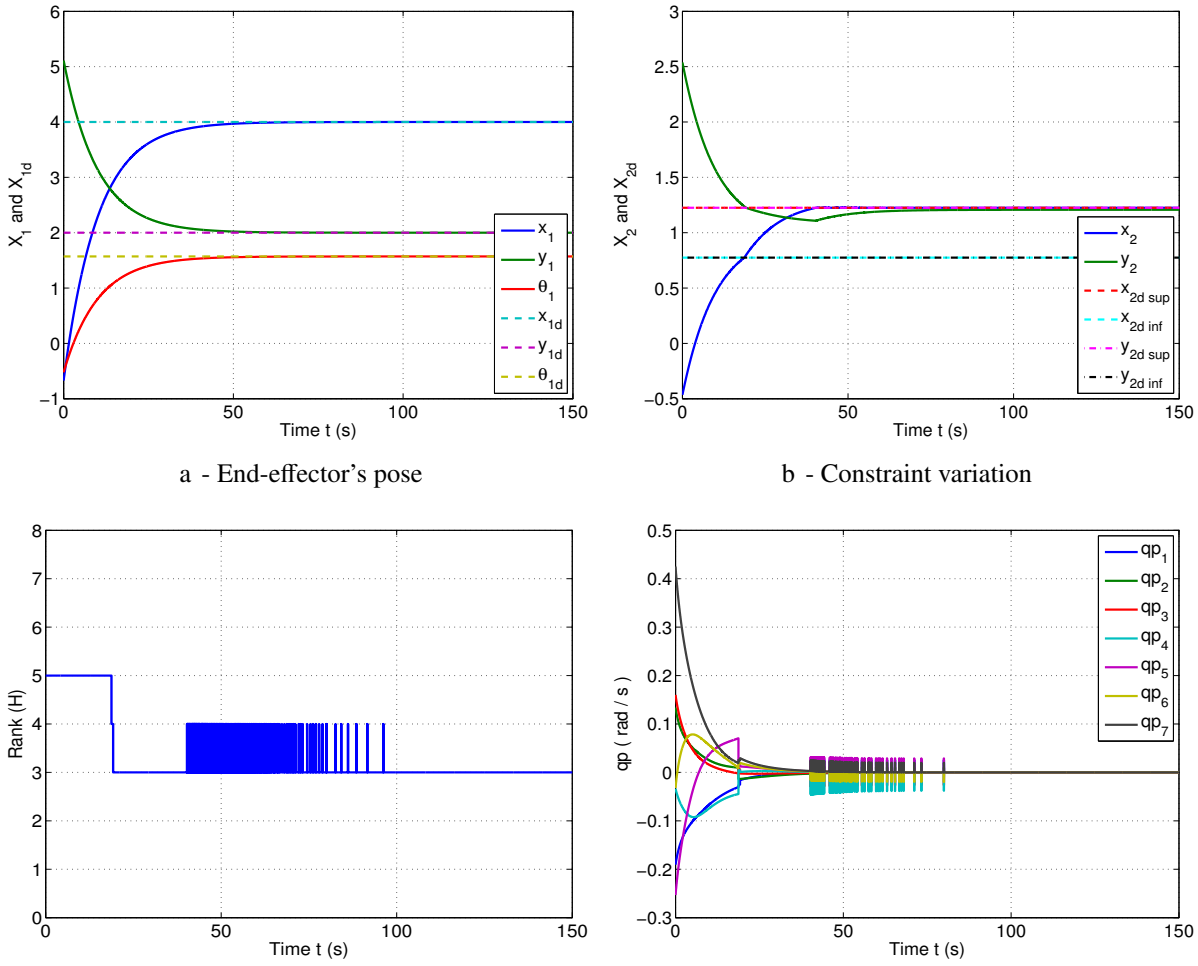
d - Joint velocities

Figure III.17 – Simulation results in case of unreachable task $\mathbf{T}_2$ with minimum norm solution

III.3.4.4 Hybrid control

Finally, for the hybrid control, the main task is regulated normally to the desired values and the constraint is maintained is the desired domain (Fig. III.18a-III.18b).

The rank of the weighting matrix begins with the value 5 (when the constraint in x and y directions is violated), then the rank drops to 3 when the constraint arrives to the desired domain, in fact each time the constraint arrives near the domain limits, the rank of the weighting matrix increases to prevent the constraint from going out of the domain. However, due to the latter case, we notice oscillations and an increase in the joint velocities (Fig. III.18d).



a - End-effector's pose

b - Constraint variation

c - Rank of the weighting matrix

d - Joint velocities

Figure III.18 – Simulation results in case of unreachable task $\mathbf{T}_2$ with hybrid control

III.3.4.5 Comparison results

In conclusion, only the directional projection method gives acceptable behavior in case of the secondary task is unreachable i.e. convergence of high priority task and best reachable pose for the unreachable task. For the other methods, this case leads to an unstable system and/or oscillations.

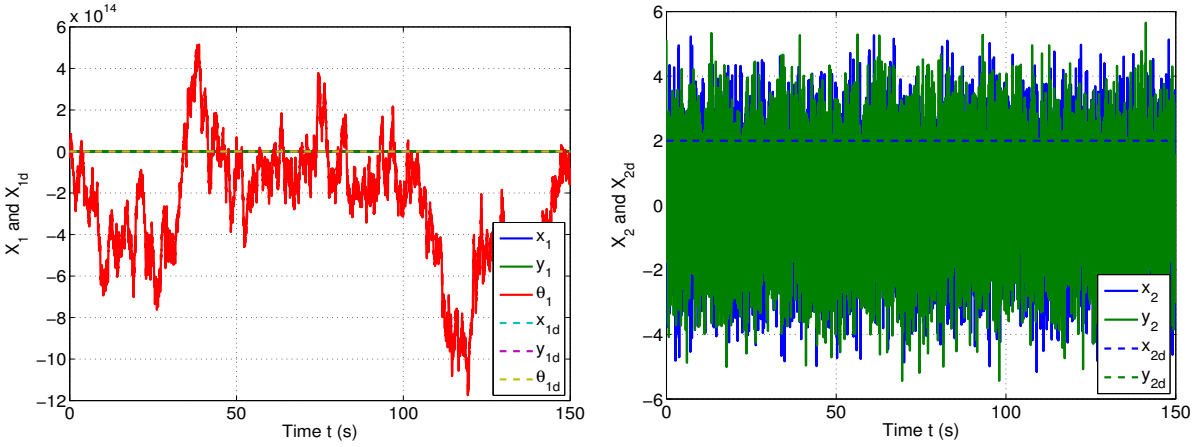
III.3.5 Case of Incompatible Tasks

In this part, we study the kinematically indeterminate system's behavior if the two tasks, which control the same control point, are incompatible. In this part, the two tasks $\mathbf{T}_1$ and $\mathbf{T}_2$ control the 7R-planar robot's end-effector to two different positions/orientations with a decreasing priority ($\mathbf{T}_1$ has more priority over $\mathbf{T}_2$).

In such case, the task priority should be respected, and the system should converge to the higher priority task $\mathbf{T}_1$ without losing its stability.

III.3.5.1 Orthogonal projection

Simulation results (Fig. III.19) show that for the orthogonal projection, tasks execution is impossible due to the opposite desired values given to the same control point, thus high joint velocities are noted, and the system is quickly destabilized.



a - End-effector's pose

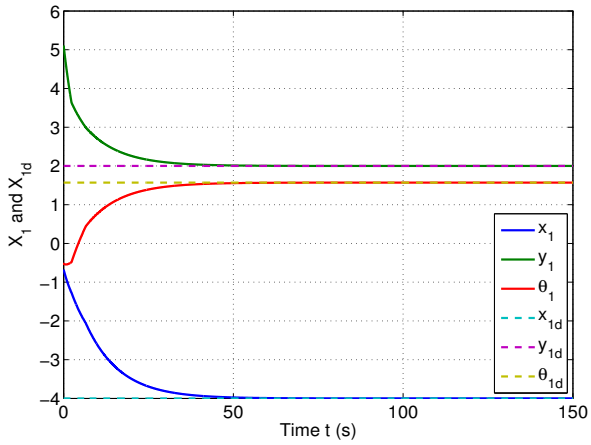
b - Intermediate joint's position

Figure III.19 – Simulation results in case of incompatible tasks with orthogonal projection

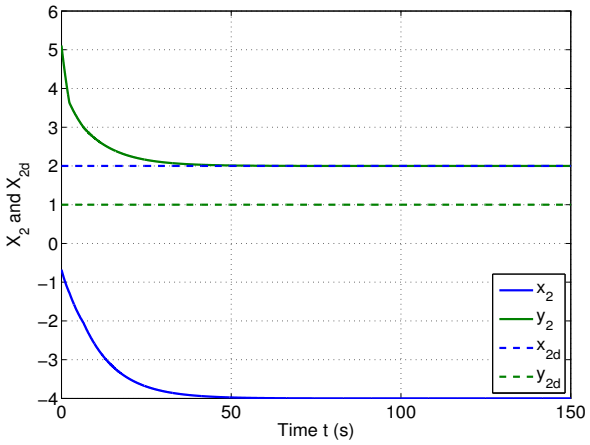
III.3.5.2 Directional projection

For the directional projection method (Fig. III.20), the system converges to the higher priority task with acceptable joint velocities (Fig. III.20d). However, a discontinuity in $\dot{\mathbf{q}}$ is noticed when the projector's rank decreases (the reason of this behavior will be discussed in the conclusion part of this section).

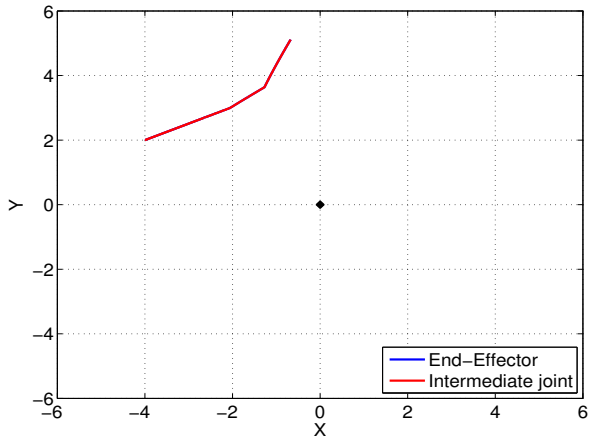
Furthermore, an acceptable trajectory of the end-effector and a reasonable variation of the performance index is observed (Fig. III.20c-III.20f). Thus, another advantage is noted for the directional projection where task priority is respected even when two incompatible tasks are considered in the same control point.



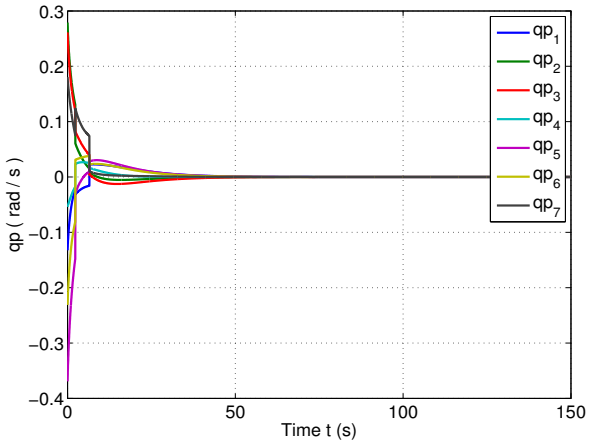
a - End-effector's pose



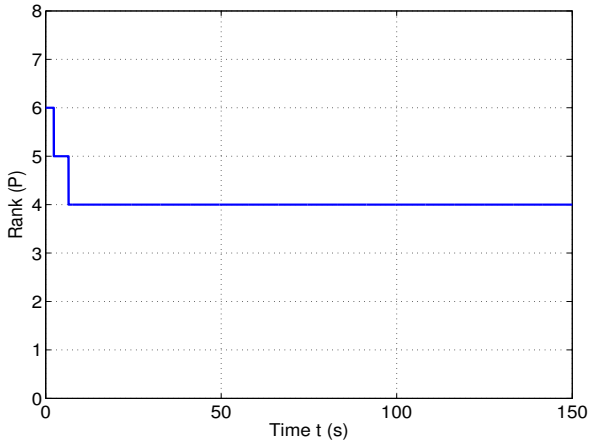
b - Intermediate joint's position



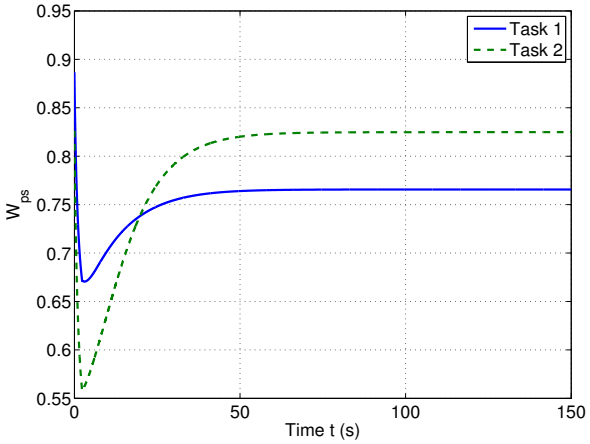
c - Control points trajectory in plane space



d - Joint velocities



e - Projector's rank

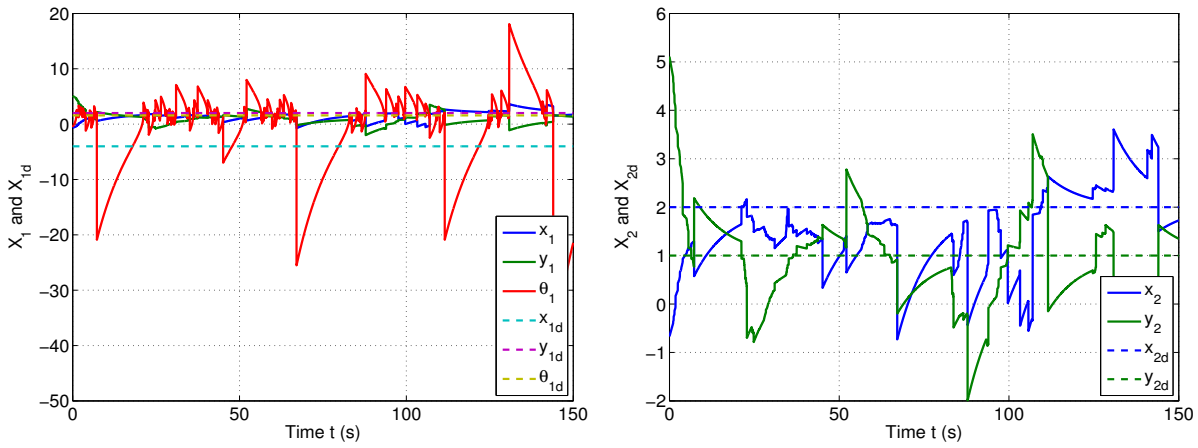


f - Performance index

Figure III.20 – Simulation results in case of incompatible tasks with directional projection

III.3.5.3 Minimum norm solution

For the minimum norm solution $\mathbf{P}_\eta$, simulation results (Fig. III.21) show the same problem at convergence due to the switching strategy: the system is blocked in the minimum norm solution part and doesn't pass to the orthogonal projection ($\mathbf{Rank}(\mathbf{P}_\eta) = 6$); this behavior leads to oscillations and instability.

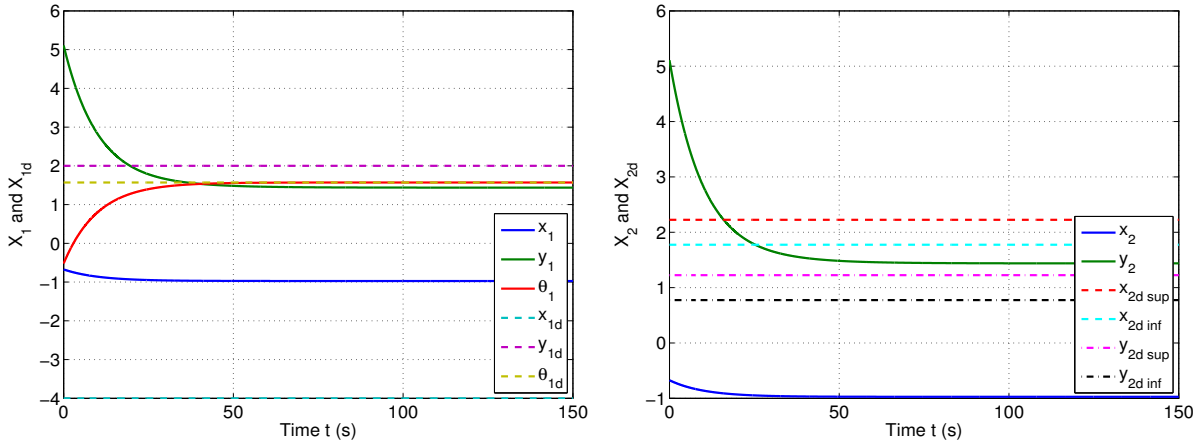


a - End-effector's pose b - Intermediate joint's position

Figure III.21 – Simulation results in case of incompatible tasks with minimum norm solution

III.3.5.4 Hybrid control

Finally, for the hybrid control (Fig. III.22), the system converges to an intermediate pose between the desired values of the main and secondary task. Thus, the system is stable but the task priority is not respected.



a - End-effector's pose b - Constraint variation

Figure III.22 – Simulation results in case of incompatible tasks with hybrid control

III.3.6 Conclusion on Simulation Results Comparison

The general comparison of the presented methods is summarized in Table III.7, where the appealing behaviors are highlighted.

Simulation results show that the classical orthogonal projection $\mathbf{P_e}$ has the best behavior with respect to the linear trajectory of the control points, the exponential decrease of the total error, the acceptable convergence time and the lack in parameters tuning. However, problems of instability and high joint velocities arise when the tasks are not well defined (in case of unreachable or incompatible tasks).

The directional projection $\mathbf{P_z}$ is applicable when the CP behavior is not essential (no desired linear trajectory of the end-effector), but more critical is that it give the best behavior if no sufficient DOF are available for all tasks ($l > r$). Moreover, it is the only method that maintain the stability of the system in case of unreachable pose of the secondary tasks, and in case of incompatibility between the controlled tasks.

	CP Trajectory	**Error Variation**	**Projector Rank**	**Convergence time**	$\mathbf{W}_{ps}$	$\mathbf{E}_c$
$\mathbf{P}_e$	Linear	Exponential	Fixed	Acceptable	Average	Low
$\mathbf{P}_z$	Non Linear	Non Exponential	Variable	Unacceptable	Average	Average
$\mathbf{P}_\eta$	Non Linear	Non Exponential	Variable	Acceptable	Average	High
H	Linear	Exponential	Variable	Acceptable	High	Low

	Parameter tuning	**Unreachable T2**	**Incompatible tasks**	**l = r**	**l > r**
$\mathbf{P}_e$	Easy	Bad	Bad	Good	Good
$\mathbf{P}_z$	Hard	Good	Good	Good	Best
$\mathbf{P}_\eta$	Average	Bad	Bad	Good	Good
H	Hard	Average	Bad	Average	Average

Table III.7 – Summary on task sequencing control laws comparison

The disadvantage of this method is that it requires rough parameter tuning for the discrete case (to remove oscillations as noted in [Man06]), especially the value of the sampling interval (Δt) in equation (II.29). In addition to that, the relatively high values of the performance index indicates that this method approaches system's singularities.

Furthermore, for the directional projection, in spite of the improvement in the convergence time of the main task of approximately 5% in the normal case (due to the condition on Lyapunov function to be faster than that of the orthogonal one), the convergence time of secondary task increases with the number of applied tasks and may explode with a large number of tasks. In fact, this task is regulated to the desired pose but with insufficient precision ($\approx 10^{-3}$), while 10^{-7} is used as a threshold for the other methods.

In fact, the higher value of the convergence time for the secondary task, is due to the choice of the task sequencing method (II.34) that does not remove the part of the secondary task which is executed by the main task from the minimum solution, which leads to imprecise regulation of the secondary task as explained in paragraph II.4.2. Recall that this choice of task sequencing is only considered for $\mathbf{P_z}$ due to the complexity of stability proofing with the optimal task sequencing method given by (II.41).

A last note on the directional method, which use the SVD decomposition of the main task Jacobian ($\mathbf{J_1} = \mathbf{U_1 \Sigma_1 V_1}^\top$) to project the secondary task in the space of singular values. In fact, the discontinuity in the singular value decomposition, that appears in the discontinuity of the columns of $\mathbf{U}_1$ and $\mathbf{V}_1$ matrices, leads to a swing between the different conditions in the projector's definitions (II.25 and II.29), and thus an oscillation in the joint velocities which may leads to high accelerations and thus vibration of the system (as noticed in Fig. III.20d for example).

In spite of the higher projector's rank, and thus the higher number of available DOF when using the method of minimum norm solution $\mathbf{P}_\eta$, the main problem of this approach lies in the switching strategy performed near convergence due to projector's singularity. The system is blocked in the switching area when applying unreachable or incompatible tasks. In contrast to the orthogonal method, the trajectory of the controlled points is not linear due to the exponential decrease of the error norm and not the error components.

Finally, the hybrid control with weighting matrix $\mathbf{H}$ requires parameter tuning especially in case of dynamic h_i and difficult to proof system stability. Acceptable behavior is noticed in case of kinematically indeterminate systems. However, when there is no sufficient DOF, the system approaches more to singularities thus highest joint velocities and accelerations are required to execute the desired motion. Moreover, problems occur in case of unreachable and incompatible tasks. Nevertheless, linear trajectory and exponential decrease of the task error is observed when applying hybrid control.

In the next section, the observed behaviors of the different methods are validated by applying different tasks on the 7 DOF LWR Kuka robot.

III.4 Simulation on LWR Kuka Robot

In the previous section, we used planar robots in several configurations to study and compare the different redundancy resolution methods. In this section, we verify the comparison results by applying several tasks on the LWR Kuka robot (see Fig. III.23 and Table III.8).

Joint	σ	α	d	θ	r
1	0	0	0	θ_1	0.3105
2	0	$\pi/2$	0	θ_2	0
3	0	-$\pi/2$	0	θ_3	0.400
4	0	-$\pi/2$	0	θ_4	0
5	0	$\pi/2$	0	θ_5	0.390
6	0	$\pi/2$	0	θ_6	0
7	0	-$\pi/2$	0	θ_7	0.078

Table III.8 – MD-H parameters of the LWR4+ robot

This robot has 7 DOF ($n = 7$), the end-effector has a mobility of 6 DOF ($m = 6$); thus, the degree of redundancy is always $r = 1$. To study the system's behavior, we define the following tasks:

— The first task $\mathbf{T}_1$ to control the robot's end-effector position and orientation (6 DOF).

— A secondary task $\mathbf{T}_2$ to control the 3D position of the origin of the 4^{th} frame (3 DOF).

In the first part, we apply only the first task to verify the possibility to use the chosen position/orientation parametrization. Thus the task $\mathbf{T}_1$ of 6 DOF is applied in case of kinematically indeterminate system (sufficient DOF are available to execute the task). In the second part, the two tasks are applied simultaneously; thus, the system is kinematically over-specified in this case. The same comparison criteria are considered as previously (section III.2).

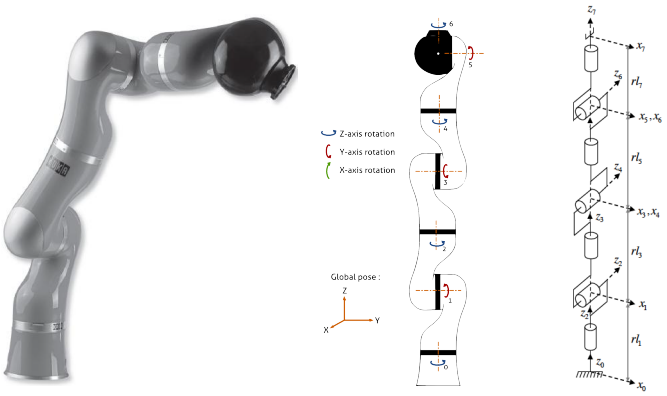


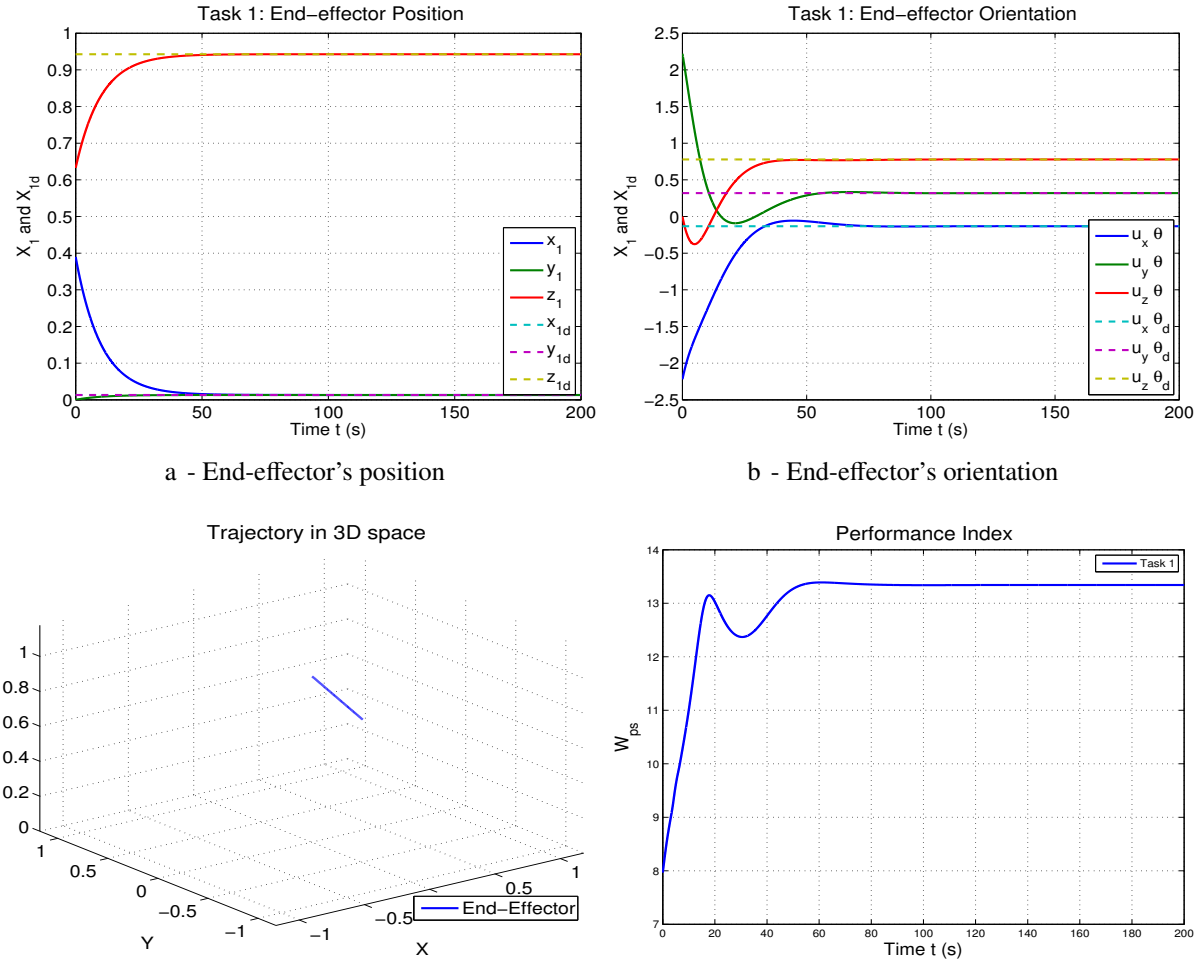
Figure III.23 – General schema of the LWR kuka robot

III.4.1 Kinematically Indeterminate System

In this paragraph, only one task of 6 DOF is applied to the 7 DOF LWR Kuka robot. In the orientation part, we use the angle-axis parametrization ($u\theta$), and no secondary task is applied.

Fig. III.24a-III.24b show the successful regulation of the position and orientation errors. In Fig. III.24c the linear trajectory of the controlled point in the 3D space is shown. Acceptable joint velocities and performance index values are also observed (Fig. III.24d).

Simulation results show the possibility to apply the end-effector task, using the axis-angle parametrization, when sufficient DOF are available in the system.



a - End-effector's position

b - End-effector's orientation

c - Control points trajectory in 3D space

d - Performance index

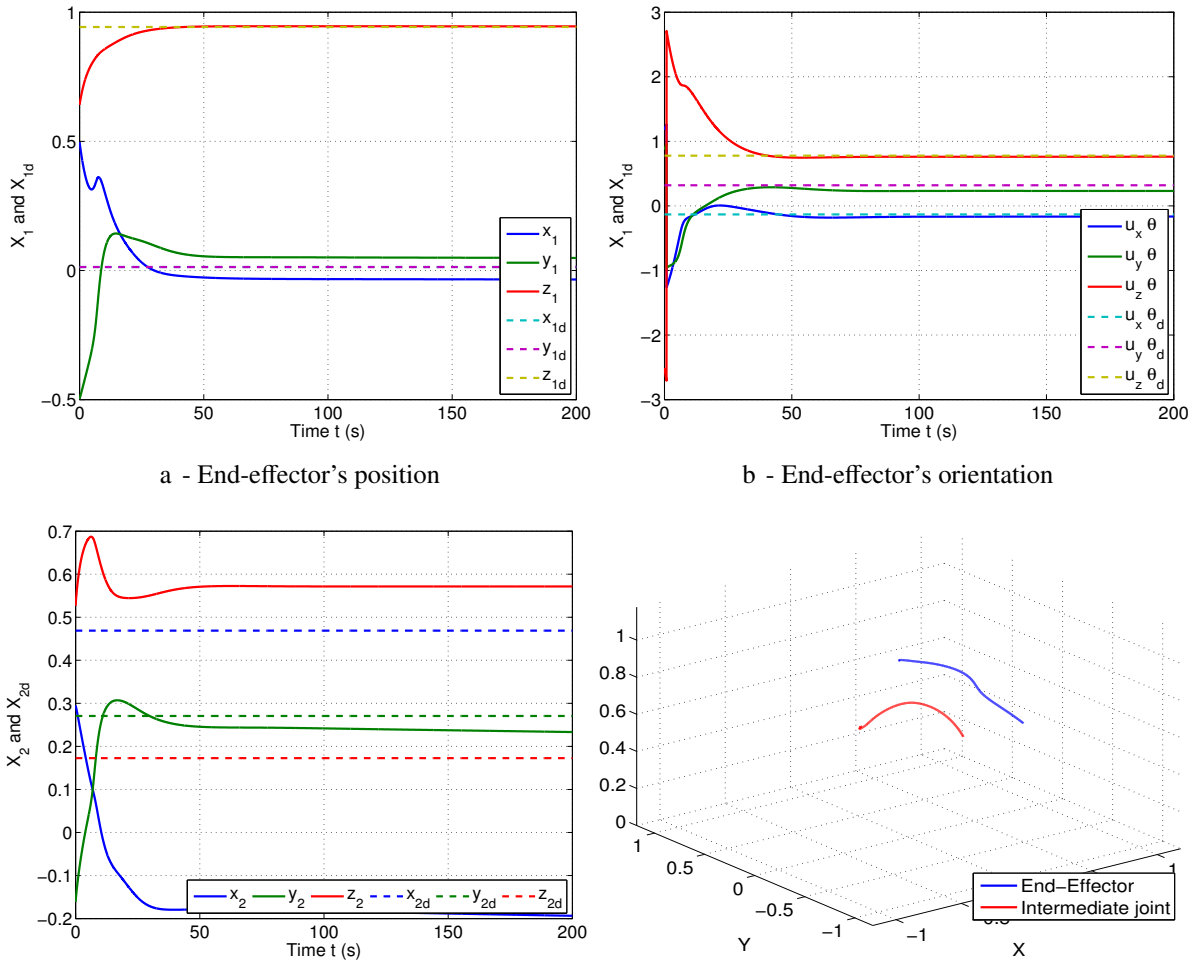
Figure III.24 – Simulation results in case of one task $\mathbf{T}_1$ on the LWR Kuka robot

III.4.2 Kinematically Over-specified System

In this part, we apply the secondary task $\mathbf{T_2}$ in addition to the main task $\mathbf{T_1}$. However, in this case, the required number of DOF is higher than those of the robot. The behavior of the system is thus studied when applying the different redundancy resolution methods.

III.4.2.1 Orthogonal projection

In Fig. III.25, using the orthogonal projection method, the first task is regulated to a pose near the desired value in position and orientation, and not to the exact one. The second task converges to the best reachable position without disturbing the main task or destabilizing the system. Acceptable joint velocities are used to apply these tasks, however a non linear trajectory of the controlled points is shown in Fig. III.25d. Therefore, the desired behavior, in case of over-specified system, is not reached when using the orthogonal projection method.



a - End-effector's position

b - End-effector's orientation

c - End-effector's orientation

d - Control points trajectory in 3D space

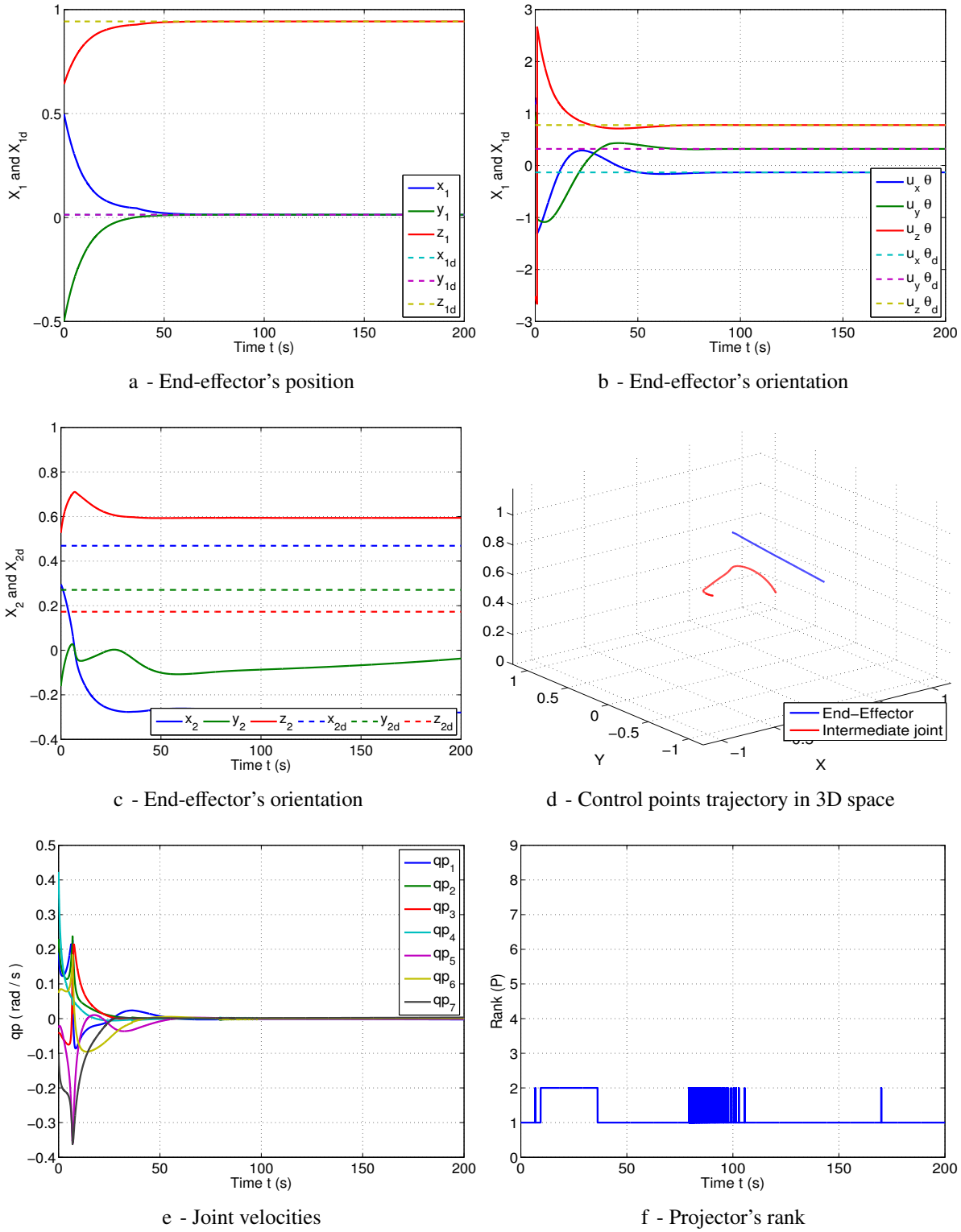
Figure III.25 – Simulation results on the Kuka robot with orthogonal projection

III.4.2.2 Directional projection

The directional projection method $\mathbf{P_z}$ is used in Fig. III.26 to apply the desired tasks. In this case, the main task converges to the desired values in position and orientation, without being disturbed by the secondary task (Fig. III.26a-III.26b). The latter converges to most reachable position because of the lack in the system's DOF (Fig. III.26c).

As expected, a non linear trajectory of the considered CP in the 3D space is shown in

Fig. III.26d, it can be performed with acceptable joint velocities; thus, this motion is feasible.



a - End-effector's position

b - End-effector's orientation

c - End-effector's orientation

d - Control points trajectory in 3D space

e - Joint velocities

f - Projector's rank

Figure III.26 – Simulation results on the LWR Kuka robot with directional projection

III.4.2.3 Minimum norm solution

For the method of minimum norm solution $\mathbf{P}_\eta$, the first task converges to an intermediate pose (different from the desired one), and the secondary one converges also to a different position than the desired one. However, the small oscillations in the robots control points and the high joint velocities indicate that the system is unstable, and the desired tasks can not be applied using this method. In fact, the system is blocked (as noted in previous cases) in the first part before switching to the orthogonal method that is usually performed near convergence ($\mathbf{Rank}(\mathbf{P}_\eta) = 6$ in Fig. III.27d).

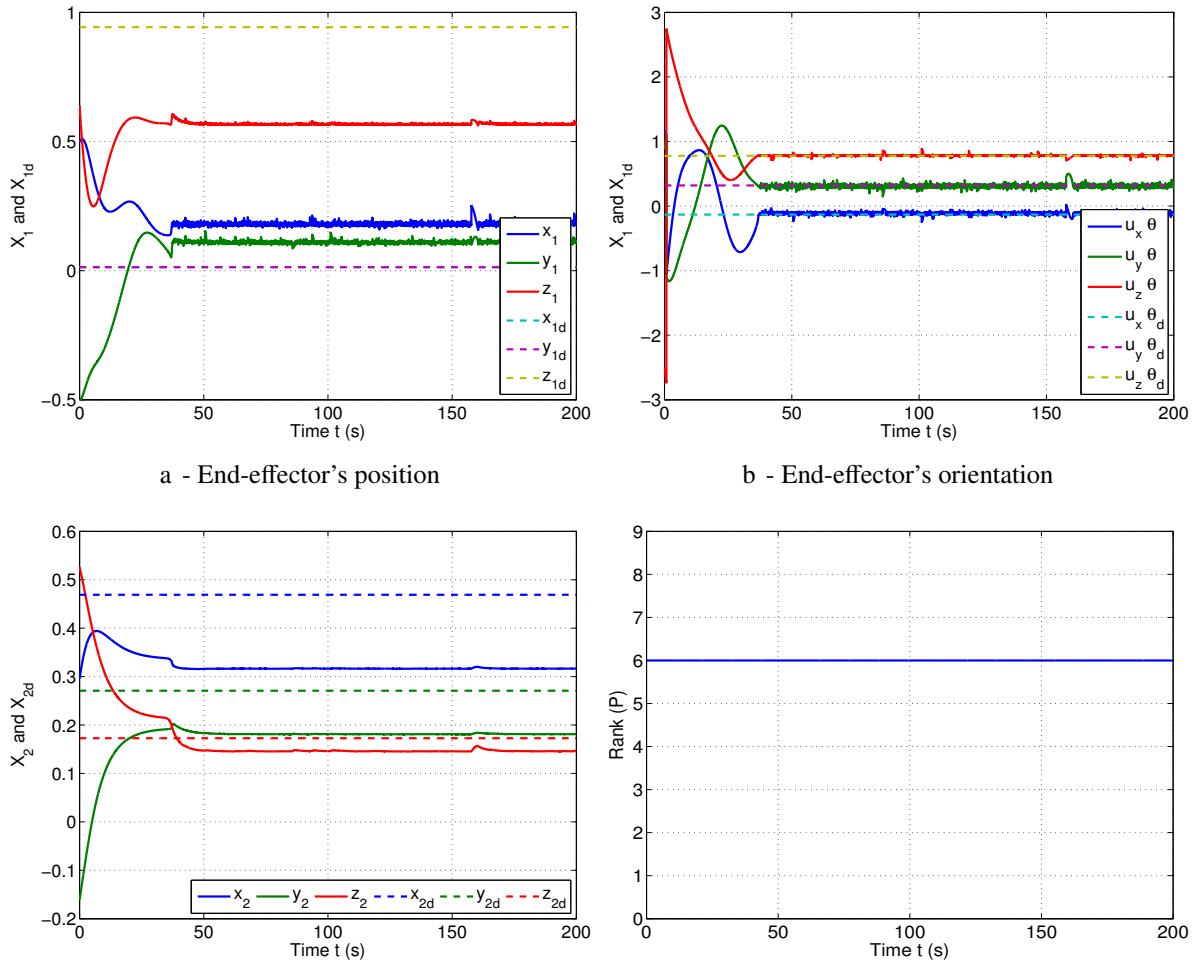
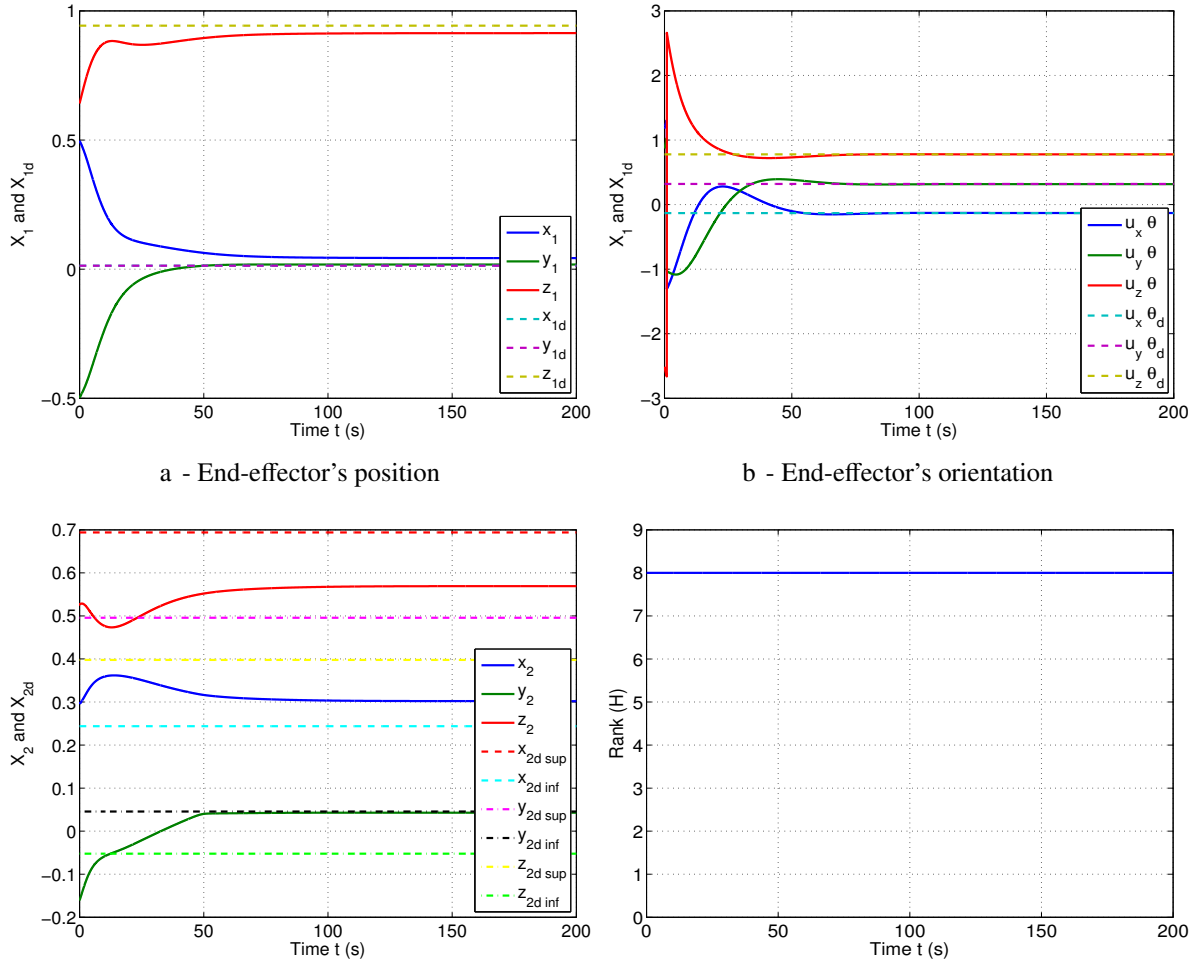


Figure III.27 – Simulation results on LWR Kuka robot with minimum norm solution $\mathbf{P}_\eta$

III.4.2.4 Hybrid control

In this case, the $\mathbf{T_1}$ is considered as task and $\mathbf{T_2}$ as constraint which is controlled to stay around the desired value. Using the hybrid control method (Fig. III.28), the system converges to a position near the desired one, however the end-effector's orientation is normally regulated because it's independent from the position part. Although the constraint is partially respected, the main task is not regulated correctly.

On the other hand, the x_2 and y_2 components of the constraint are always outside the desired domain, thus the hybrid control uses non-zero weights to try regulating these components, in addition to the 6 DOF of the main task; thus the considered weighting matrix has 8 non-zero elements and the rank is always equal to $\mathbf{Rank}(\mathbf{H}) = 8$ as represented in Fig. III.28-d.



a - End-effector's position

b - End-effector's orientation

c - Constraint variation

d - Rank of the weighting matrix

Figure III.28 – Simulation results on the LWR Kuka robot with hybrid method **H**

III.4.2.5 Conclusion on Simulation Results

Finally, these simulations on the LWR robot verify the advantage of the directional projection method over the other techniques when there is an insufficient number of DOF in the system. In fact, the hybrid control and the orthogonal projection methods does not allow the exact regulation of the main in this case, and the method minimum norm solution leads to oscillations and instability of the system. However, the directional projection method ensure the stability of the system, the execution of the main and regulates the secondary tasks to the best reachable pose.

III.5 New Unified Projector for Orthogonal and Directional Methods

III.5.1 Discussion on Projection Operators

The comparison and simulation results, that are discussed in the previous sections and summarized in Table III.7, show that there is no one redundancy resolution method that has perfect behavior in all the studied cases. However, it can be noticed that the orthogonal and the directional projection methods have good performance in complementary cases. In fact, the first one is easily applicable when sufficient DOF are available, and the latter one in the special case of insufficient DOF, and unreachable or incompatible tasks.

Returning to the theoretical construction of the directional projection approach, it consists on allowing secondary task components which goes into the same direction of the main task and does not perturb the system's stability. Thus, if none of the components of the secondary tasks is enabled, this approach performs only the main task and thus returns to the classical orthogonal projection approach. Therefore, these two approaches could be merged into one general formalism.

In addition to that, in several cases, it is not necessary to fully consider the secondary task in the control law, such as when it corresponds to a constraint which is not always present. In such cases, these secondary tasks/constraints can be fully activated or partially activated (in a desired direction) or even deactivated and projected in the null space of the main task (as in the classical orthogonal projection). Therefore, we suggest adding a weighting term on the projector of the secondary task which is tuned depending on the desired task/constraint to be executed.

Furthermore, in case of directional projection, we can notice in several graphs of the joint velocities that there is a discontinuity in the velocity which appears when there is a change in the projector's rank. In fact, this behavior is due to the variation in the projector's elements that change between two values: 0 and 1, depending on the sign of the main task error and secondary task joint velocities (projected in the space of singular values). Thus, a smoother transition between these values can improve the robot's behavior.

Moreover, using such transition function, allows not only to pass from the directional projection to the orthogonal one near convergence, but also to perform the exact regulation of the secondary task by using the control law defined in (II.41) in contrast to the directional method that gives high convergence time (since it does not consider the main task effect on the secondary one as already explained in section II.4.3).

III.5.2 Definition of the Unified Projection Operator

The general idea of the new projection operator is not only to enable the motions produced by the secondary control law that help the main task to be completed faster, but also to control these motions and thus to control the direction of the projection.

We consider the same goal as in the directional case: the secondary control law should

not increase the error of the main task (to preserve the stability of the system), that is when the secondary task goes in the same direction than the main task. Thus, a non-linear projection operator $\mathbf{P}_w$ is proposed to enlarge the free space on which the secondary task is projected:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}} + \mathbf{P}_w\,\mathbf{z} \tag{III.3}$$

Using the classical Lyapunov function on the main task error $\left(\mathcal{L} = \frac{1}{2}\mathbf{e}^{\top}\mathbf{e}\right)$, the projection operator will be built to have a faster convergence of the main task than in case of the orthogonal method; thus, the second part the control law ($\dot{\mathbf{q}}_\mathbf{2} = \mathbf{P}_w\,\mathbf{z}$) should respect the following condition:

$$\widetilde{\nabla\mathcal{L}}^{\top}\,\mathbf{\Sigma}\,\widetilde{\dot{\mathbf{q}}}_\mathbf{2} \leq \mathbf{0} \tag{III.4}$$

where $\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ is the singular values decomposition of $\mathbf{J}$, $\nabla\mathcal{L} = (\frac{\partial\mathcal{L}}{\partial\mathbf{e}})^{\top}$, $\widetilde{\nabla\mathcal{L}} = \mathbf{U}^{\top}\nabla\mathcal{L}$ and $\widetilde{\dot{\mathbf{q}}}_\mathbf{2} = \mathbf{V}^{\top}\dot{\mathbf{q}}_\mathbf{2}$.

To simplify the construction of the projection operator, and by considering the elements $\tilde{l}_i$ of the diagonal matrix $\widetilde{\nabla\mathcal{L}} = (\tilde{l}_1, ..., \tilde{l}_m)$, this condition is restricted to the following:

$$\forall i \in [1...m],\;\; \tilde{l}_i\,\sigma_i\,\widetilde{\dot{q}}_{2_i} \leq 0$$

Since σ_i are the m positive singular values of $\mathbf{J}$, this condition can be simply written as:

$$\forall i \in [1...m],\;\; \tilde{l}_i\,\widetilde{\dot{q}}_{2_i} \leq 0 \tag{III.5}$$

As previously noted, this condition is more restrictive than the previous one given by (III.4). However, it ensures the convergence of each singular components of the error separately, thus it ensures that each component of the main task error will be faster than the required motion $\dot{\mathbf{e}}$.

For a given secondary control law $\mathbf{z}$, the secondary term $\dot{\mathbf{q}}_\mathbf{2}$ that respect (III.5) is built by keeping and weighting the components that respect this condition and by nullifying the other components:

$$\widetilde{\dot{\mathbf{q}}}_{\mathbf{2_i}} = \begin{cases} 0 & \text{if} \quad \tilde{z}_i\,\tilde{l}_i > 0 \\ \tilde{z}_i & \text{if} \quad i > m \;\text{ or }\; \tilde{z}_i = 0 \\ w\,\tilde{z}_i & \text{if} \quad \tilde{z}_i\,\tilde{l}_i < 0 \end{cases}$$

where w is a weighting term such that $w \in [0, 1]$, and m is the number of non-zero singular values of the Jacobian of the main task.

If $w = 0$, the secondary task is projected into the null space of the main task, and it corresponds to the orthogonal projection. Otherwise, if $w = 1$, thus the secondary task is fully considered in the control law, and in this case it corresponds to the classical directional projection ($\mathbf{P}_\mathbf{z}$).

This equation can be represented in a matrix form:

$$\widetilde{\dot{\mathbf{q}}_2} = \mathbf{V}^\top \dot{\mathbf{q}}_2 = \mathbf{V}^\top \mathbf{P}_{w(\tilde{\mathbf{z}})}\, \mathbf{z} = \mathbf{P}_{w(\tilde{\mathbf{z}})} \mathbf{V}^\top\, \mathbf{z} = \mathbf{P}_{w(\tilde{\mathbf{z}})}\, \tilde{\mathbf{z}} = \begin{bmatrix} p_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & p_n(\tilde{z}) \end{bmatrix} \tilde{\mathbf{z}} \tag{III.6}$$

where $\tilde{\mathbf{z}} = \mathbf{V}^\top \mathbf{z}$ and the components $p_i(\tilde{\mathbf{z}})$ of $\mathbf{P}_{w(\tilde{\mathbf{z}})}$ are defined by:

$$p_i(\tilde{\mathbf{z}}) = \begin{cases} 0 & \text{if} \quad \tilde{z}_i\, \tilde{l}_i > 0 \\ 1 & \text{if} \quad i > m \ \text{ or } \ \tilde{z}_i = 0 \\ w & \text{if} \quad \tilde{z}_i\, \tilde{l}_i < 0 \end{cases}$$

Finally, the control law that realizes the main task $\mathbf{e}$ and ensures that the secondary control law $\mathbf{z}$ respects condition (III.5) can finally be written:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \dot{\mathbf{e}} + \mathbf{P}_w\, \mathbf{z} \tag{III.7}$$

where $\mathbf{P}_w = \mathbf{V}\, \mathbf{P}_{w(\tilde{\mathbf{z}})} \mathbf{V}^\top$.

Note that the presented projector is defined in the space of singular values of the main task, and not in the classical joint or cartesian spaces.

Using the same representation as for the orthogonal and directional projections, we show in Fig. III.29 the main and secondary tasks with their projections. If the secondary task is in the null space (non-hashed half plan), the vector is modified by the projection depending on the weighting value w (such as vectors $\mathbf{z}_1$ and $\mathbf{z}_2$). If the secondary task is out of this space, it is projected into the $Ker(\mathbf{J})$ line (such as vector $\mathbf{z}_3$).

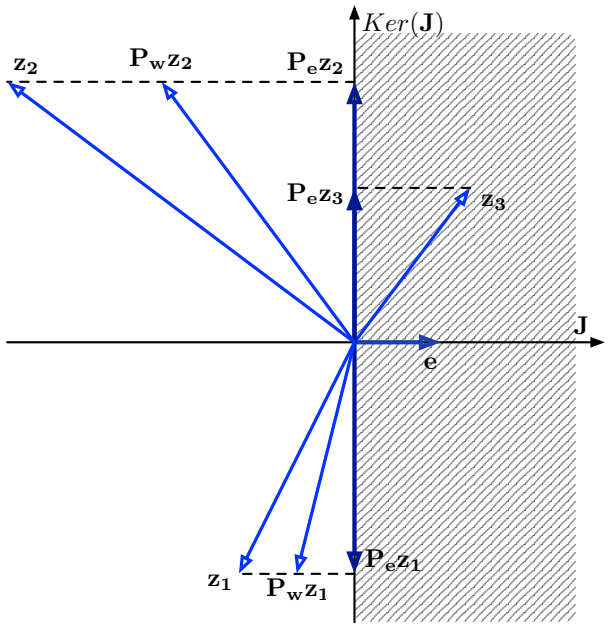


Figure III.29 – Example of the new directional projection of simple tasks using the control law (III.7)

III.5.3 Choice of the Weighting Value

Depending on the executed secondary task and its compatibility with the main one, the value of the weighting parameters changes. In fact, if sufficient DOF are available to execute the main and secondary task, $w = 0$ corresponds to the orthogonal projection which gives the best behavior in this case. If there is insufficient DOF to the execute both tasks, the second task is unreachable or the two tasks are incompatible $w = 1$ is used to maintain the stability of the system, it corresponds to the directional projection method.

Furthermore, if the secondary task is far from its desired value, a high weighting value w allows a faster regulation of this task, thus a variable weighting $w \in [0, 1]$ could be used such as it approaches to 1 when the secondary task error is sufficiently high, and smoothly passes to 0 when it is regulated, and finally arrives to $w = 0$ for the orthogonal projection.

Therefore, a general variation of the w value could be considered by using a sigmoid function which is applied on the norm of the secondary task error $\|\mathbf{e}_2\|$:

$$w\left(\|\mathbf{e}_2\|\right) = \begin{cases} 1 & \text{if} \quad \|\mathbf{e}_2\| \geq \alpha_1 \\ \frac{1}{1+\exp\left(20\frac{\|\mathbf{e}_2\|-\alpha_1}{\alpha_0-\alpha_1}-10\right)} & \text{if} \quad \alpha_0 < \|\mathbf{e}_2\| < \alpha_1 \\ 0 & \text{if} \quad \|\mathbf{e}_2\| \leq \alpha_0 \end{cases} \tag{III.8}$$

where α_0, α_1 are two threshold values that define the starting and the ending conditions for the switching period, they should be selected such that the system does not converge too fast during the interval. w is thus a continuous monotonically decreasing function as represented in Fig. III.30.

Note that to allow a complete regulation of the secondary task, the orthogonal projection uses equation (II.41) to consider the effect of the main task on the secondary one. Thus, a transition to this control law is performed near convergence to reach the desired values of the executed tasks.

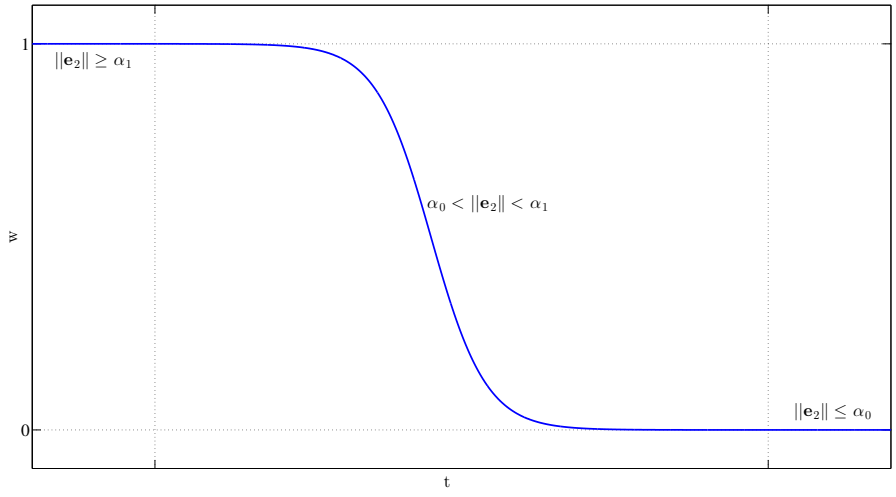


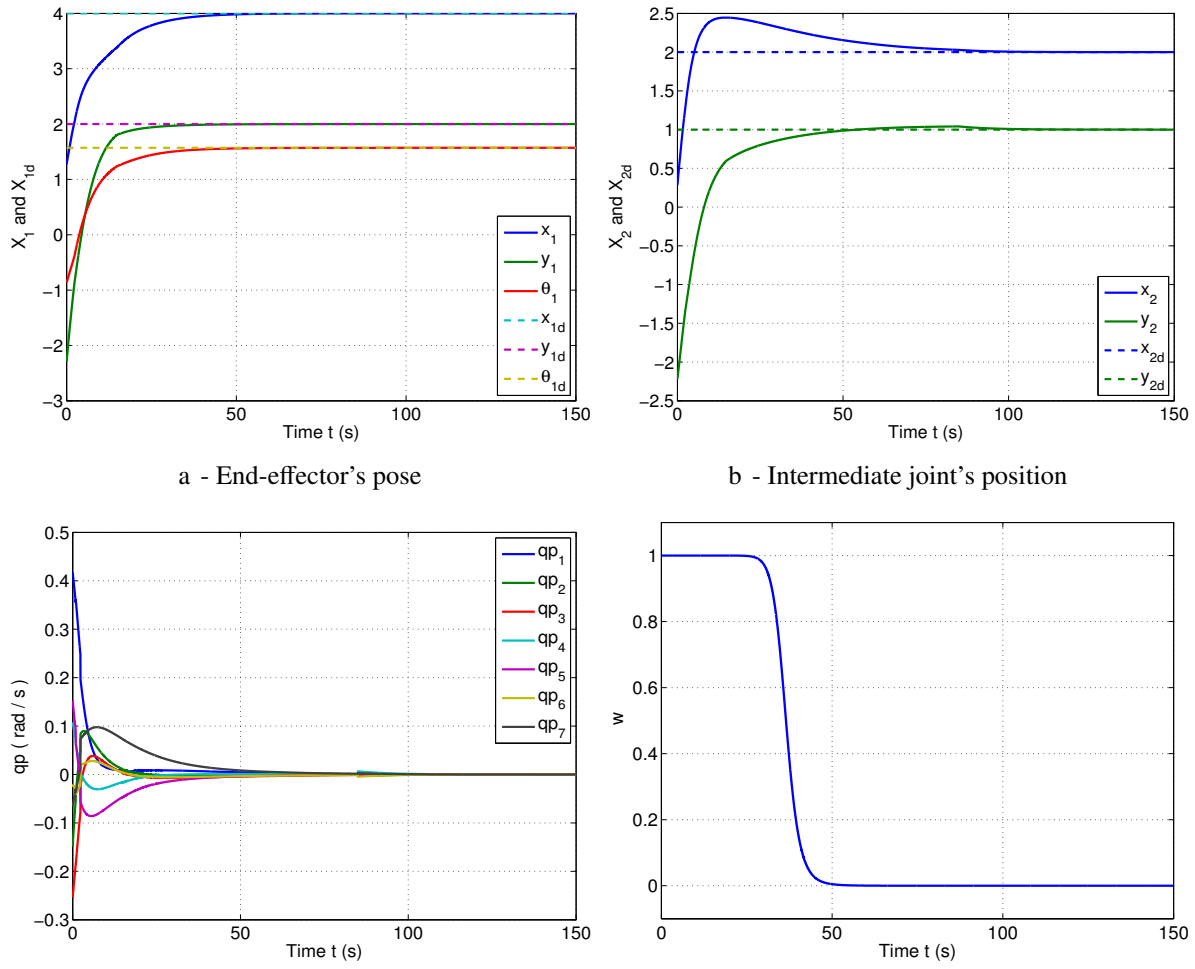
Figure III.30 – Variation of weighting value w with respect to $\|\mathbf{e}_2\|$

III.5.4 Simulations on Planar Robots

The new projector is applied in simulation for the same tasks, initial and desired conditions, and for the different case studies as conducted in the previous sections for the other projection methods. Using the already presented performance criteria, this projector's behavior will be compared to the other techniques.

In case of kinematically indeterminate system, the same acceptable behavior as that of the directional method is noted: non linear trajectory and same variation of projector's rank. However, the secondary task converges to the desired position in $\mathbf{t_{conv2}} = 124\ s$, in addition to the improvement in the convergence time of the main task $\mathbf{t_{conv1}} = 80.5\ s$ over the orthogonal method (Fig. III.31 and Table III.9).

A smooth switching from the directional to orthogonal projection is observed in the other cases (determinate and over-specified systems) also with an improvement of the convergence time of the secondary task.



a - End-effector's pose

b - Intermediate joint's position

c - Joint velocities

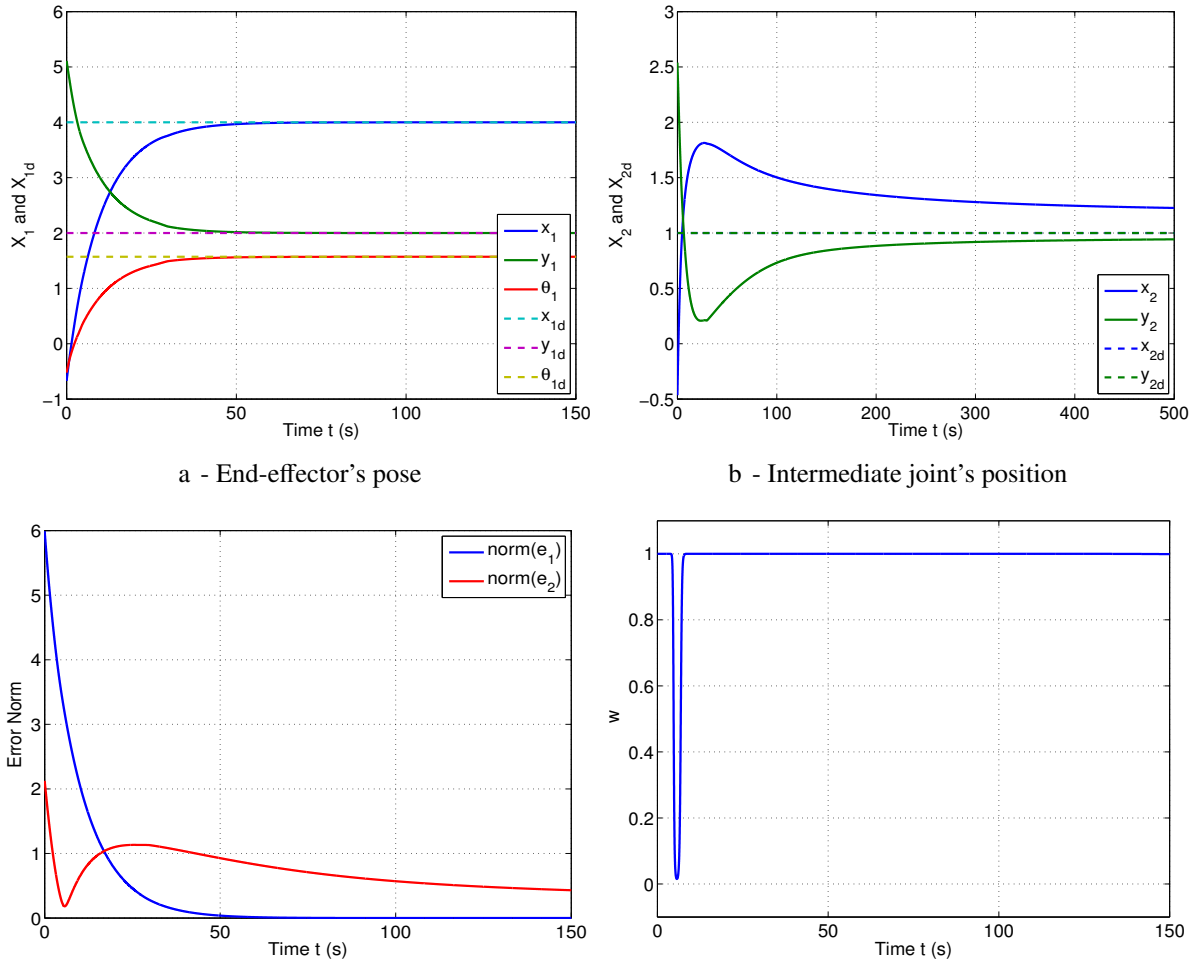
d - Weighting value

Figure III.31 – Simulation results on a 7R Planar robot with the new projection method $\mathbf{P}_w$

	Trajectory		Total Error Variation	Projector's Rank	Convergence Time		W_{ps}	E_c
	1st CP	2nd CP			t_{conv1}(s)	t_{conv2}(s)		
P_w	Non Linear	Non Linear	Non Exponential	7 ↔ 4	80.5	124	0.724	11.9
				Variable	Acceptable Time		1.128	

Table III.9 – Simulation results with the unified projector P_w on the 7R planar robot (kinematically indeterminate system)

More critical is the study of the projector's behavior in the defined particular cases. Simulation results show that in case of unreachable and incompatible secondary tasks, acceptable behavior is noted for this projector. In fact, since $\|e_2\|$ will not arrive near zero, thus at the



a - End-effector's pose

b - Intermediate joint's position

c - Task error norm

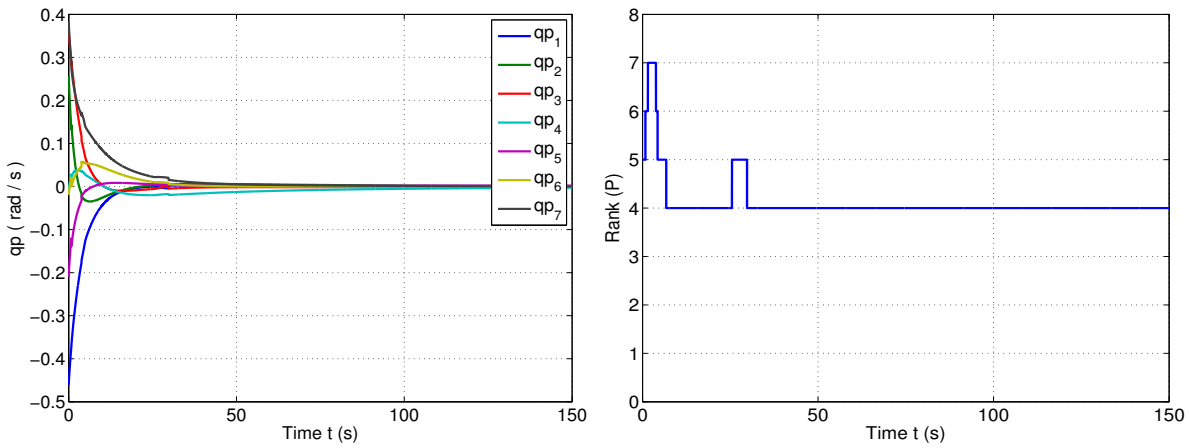
d - Weighting value

Figure III.32 – Simulation results in case of unreachable secondary task with the new projection method P_w

convergence the weighting value will remain $w = 1$ and the robot will stay stable and behave as in the case of directional projection.

Fig. III.32, shows the case of unreachable secondary task for example. $\|\mathbf{e}_2\|$ decreases at the beginning of the simulation and passes below the considered threshold thus to value of w is smaller than 1, but to respect the priority between task this error increase later to allow the regulation of the main task (III.32a-III.32b) and thus the value of w passes to $w = 1$ and remain at this value till the end of the simulation (III.32d).

Figures III.33a-III.33b show respectively the variation of joint velocities and projector's rank while performing these tasks. A smoother variation of the joint velocities is noticed, and for the projector's rank there are no oscillations as seen in the previous cases. The behavior of these elements ensures the feasibility and the stability of the defined projector.



a - Joint velocities

b - Projector's rank

Figure III.33 – (Cntl) Simulation results in case of unreachable secondary task with new projection $\mathbf{P}_w$

III.5.5 Conclusion on the Unified Projector

As a conclusion, the defined generalized projector $\mathbf{P}_w$ could be used to combine between the directional and orthogonal projectors and thus to benefit from the advantages of these methods in the different cases while maintaining the stability of the system.

This methods allows the exact convergence of the two tasks in an acceptable time, it removes also the oscillations in the projector's rank and gives a smoother variation of joint velocities.

Simulation results on the different cases show a better performance of this method over the classical directional and the orthogonal projection methods. Finally, the use of this generalized control law gives more freedom in the choice of the projection method, depending on the desired behavior. For example, if all conditions (sufficient DOF, well defined tasks,...) are present for the application of the orthogonal projection method, choosing a fixed value of $w = 0$ leads to the desired behavior.

III.6 Generalized Projection Operator

III.6.1 Definition of the Projection Operator

To overcome the problem of discontinuity in the SVD decomposition of the main task Jacobian that is mentioned in paragraph III.3.6, which leads to discontinuities in joint velocities and oscillations of the system, we define in this section a simple projection operator that benefit from the secondary task to apply faster the main task while maintaining the stability of the system and exactly regulating the two tasks (when sufficient DOF are available).

The control law will be defined by:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + \mathbf{P}_\beta \, \mathbf{z} = \mathbf{J}^+ \, \dot{\mathbf{e}} + (\mathbf{I_n} - \beta \, \mathbf{J}^+ \, \mathbf{J}) \, \mathbf{z} \tag{III.9}$$

where $\beta \in [0, 1]$ is a tuning parameter that is used to ensure the stability of the system while applying the desired behavior presented by the conditions above.

To ensure the main task regulation (with an exponential error decrease), we consider the classical Lyapunov function $\mathcal{L}_{(\mathbf{P}_\beta)}$ which minimize the norm of the main task error ($\|\mathbf{e}\|$):

$$\mathcal{L}_{(\mathbf{P}_\beta)} = \frac{1}{2} \|\mathbf{e}\|^2 = \frac{1}{2} \mathbf{e}^\top \mathbf{e} \tag{III.10}$$

Thus, the derivative of this function is given by:

$$\begin{aligned} \dot{\mathcal{L}}_{(\mathbf{P}_\beta)} &= \tfrac{\partial \mathcal{L}}{\partial \mathbf{e}} \, \dot{\mathbf{e}} = \tfrac{\partial \mathcal{L}}{\partial \mathbf{e}} \, \mathbf{J} \, \dot{\mathbf{q}} = \tfrac{\partial \mathcal{L}}{\partial \mathbf{e}} \left(\mathbf{J} \, \mathbf{J}^+ \, \dot{\mathbf{e}} + \mathbf{J} \, \mathbf{P}_\beta \, \mathbf{z} \right) \\ &= \tfrac{\partial \mathcal{L}}{\partial \mathbf{e}} \, \dot{\mathbf{e}} + \tfrac{\partial \mathcal{L}}{\partial \mathbf{e}} \, \mathbf{J} \, \mathbf{P}_\beta \, \mathbf{z} = \dot{\mathcal{L}}_{(\mathbf{P_e})} + \mathbf{e}^\top \, \mathbf{J} \, \mathbf{P}_\beta \, \mathbf{z} \end{aligned} \tag{III.11}$$

where $\dot{\mathcal{L}}_{(\mathbf{P_e})}$ is the derivative of this Lyapunov function when considering the classical orthogonal projector ($\mathbf{P_e}$).

The new projection operator defined in (III.9) is built such that it respects the desired condition of enlarging the free space on which the secondary task is projected in addition to a faster execution of the main task, which can be represented by the following inequality:

$$\mathbf{e}^\top \, \mathbf{J} \, (\mathbf{I_n} - \beta \, \mathbf{J}^+ \, \mathbf{J}) \, \mathbf{z} \leq \mathbf{0}$$

This condition can be written also as:

$$(1 - \beta) \, \mathbf{e}^\top \, \mathbf{J} \, \mathbf{z} \leq \mathbf{0} \tag{III.12}$$

For $\beta = 1$, this condition is equal to zero, and the new projector acts with the same behavior as the classical orthogonal projector ($\mathbf{P}_\beta = \mathbf{I_n} - \mathbf{J}^+ \, \mathbf{J} = \mathbf{P_e}$). For $\beta = 0$, the projector fully consider the secondary task into the control law (if $\mathbf{e}^\top \, \mathbf{J} \, \mathbf{z}$ is negative). In fact, when the secondary task does not perturb the execution of the main one, i.e. the latter inequality is valid, the control law (III.9) performs a hybrid control of the two tasks ($\mathbf{P}_\beta = \mathbf{I_n}$). A variable value of β, from 1 to 0, allows the switch from the orthogonal projection to this hybrid control.

III.6.2 Choice of the Weighting Value

If the condition ($\mathbf{e}^\top \mathbf{J}\,\mathbf{z} \leq \mathbf{0}$) is not verified, $\beta = 1$ is considered to perform the orthogonal projection, otherwise, a general variation of the β value could be considered by using a sigmoid function which is applied on the norm of the secondary task error $\|\mathbf{e}_2\|$:

$$\beta(\|\mathbf{e}_2\|) = \begin{cases} 0 & \text{if} \quad \mathbf{e}^\top \mathbf{J}\,\mathbf{z} \leq \alpha_0 \\ \frac{1}{1+\exp\left(-20\frac{\|\mathbf{e}_2\|-a}{b-a}+10\right)} & \text{if} \quad \alpha_0 < \mathbf{e}^\top \mathbf{J}\,\mathbf{z} < 0 \\ 1 & \text{if} \quad \mathbf{e}^\top \mathbf{J}\,\mathbf{z} \geq 0 \end{cases} \tag{III.13}$$

β is thus a continuous monotonically increasing function where parameters a and b are tuned to control the behavior and velocity of the transition, and α_0 is a threshold value that define the starting of the switching period, it should be selected such that the system does not pass too fast to the orthogonal projection (Fig. III.34).

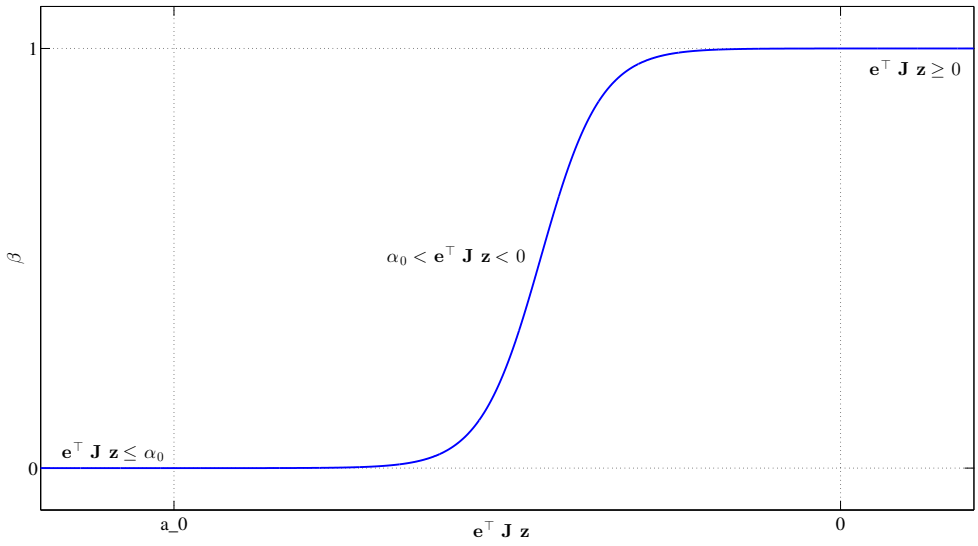


Figure III.34 – Variation of weighting value β with respect to $\|\mathbf{e}_2\|$

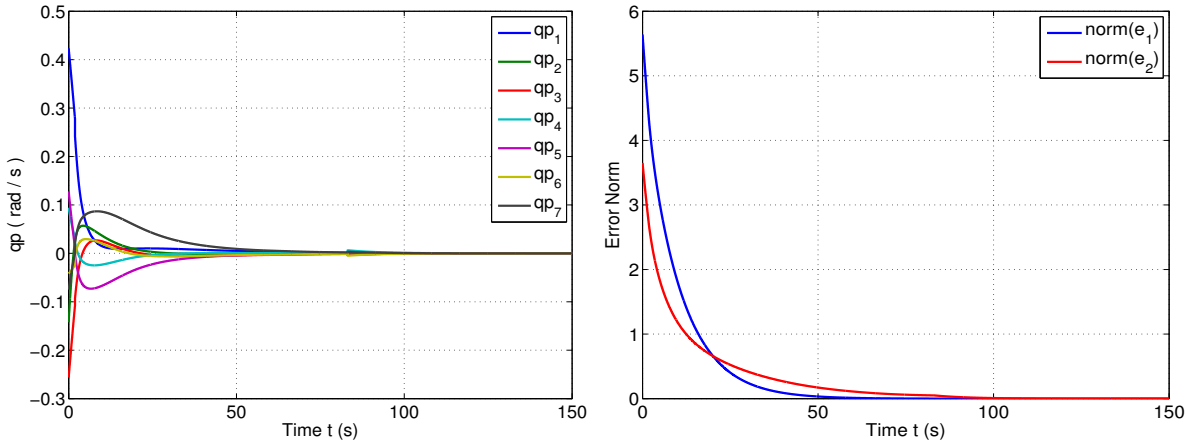
The advantage of this projection operator, with respect to the previously defined one, is that there is no projection into the singular values space which shows some discontinuities in several cases, in addition to that fewer parameters tuning are required for this method. Furthermore, the stability condition, of this control law, is more general and non-restrictive. In fact, it is considered for the overall secondary task error in contrast to the directional projection which consider this condition on the error components.

III.6.3 Simulation on Planar Robots

The new projector is applied in simulation for the same tasks, initial and desired conditions, and for the different case studies as conducted in the previous section for the other projection methods.

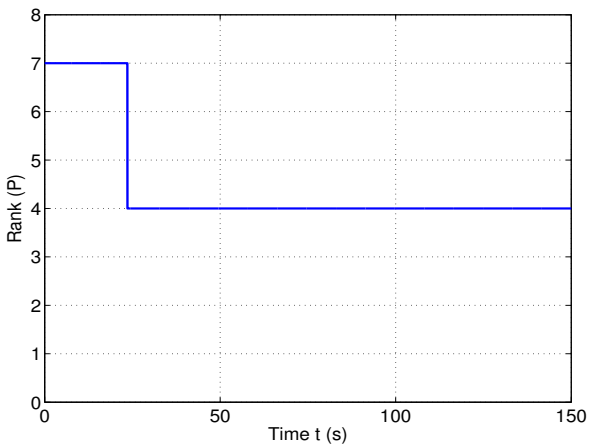
In case of kinematically indeterminate system, an acceptable behavior is noted with a non linear trajectory of the control points. However, the secondary task converges to the desired position in $\mathbf{t_{conv2}} = 122\ s$ (Fig. III.35 and Table III.10).

An acceptable variation of joint velocities and error norm of the two tasks in noted in Fig. III.35a-III.35b. The variation of the projector's rank is shown in Fig. III.35c, in fact the rank passes from 7, when the secondary task is fully or partially considered in the control law, to 4 when the secondary task is projected into the null space of the main one to maintain the stability of the system. Finally, the smooth variation of the weighting β from 0 to 1 is given in Fig. III.35d.

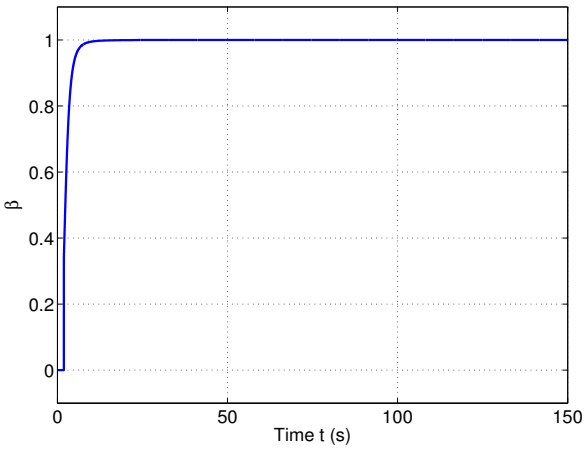


a - Joint velocities

b - Task error norm



c - Projector's rank



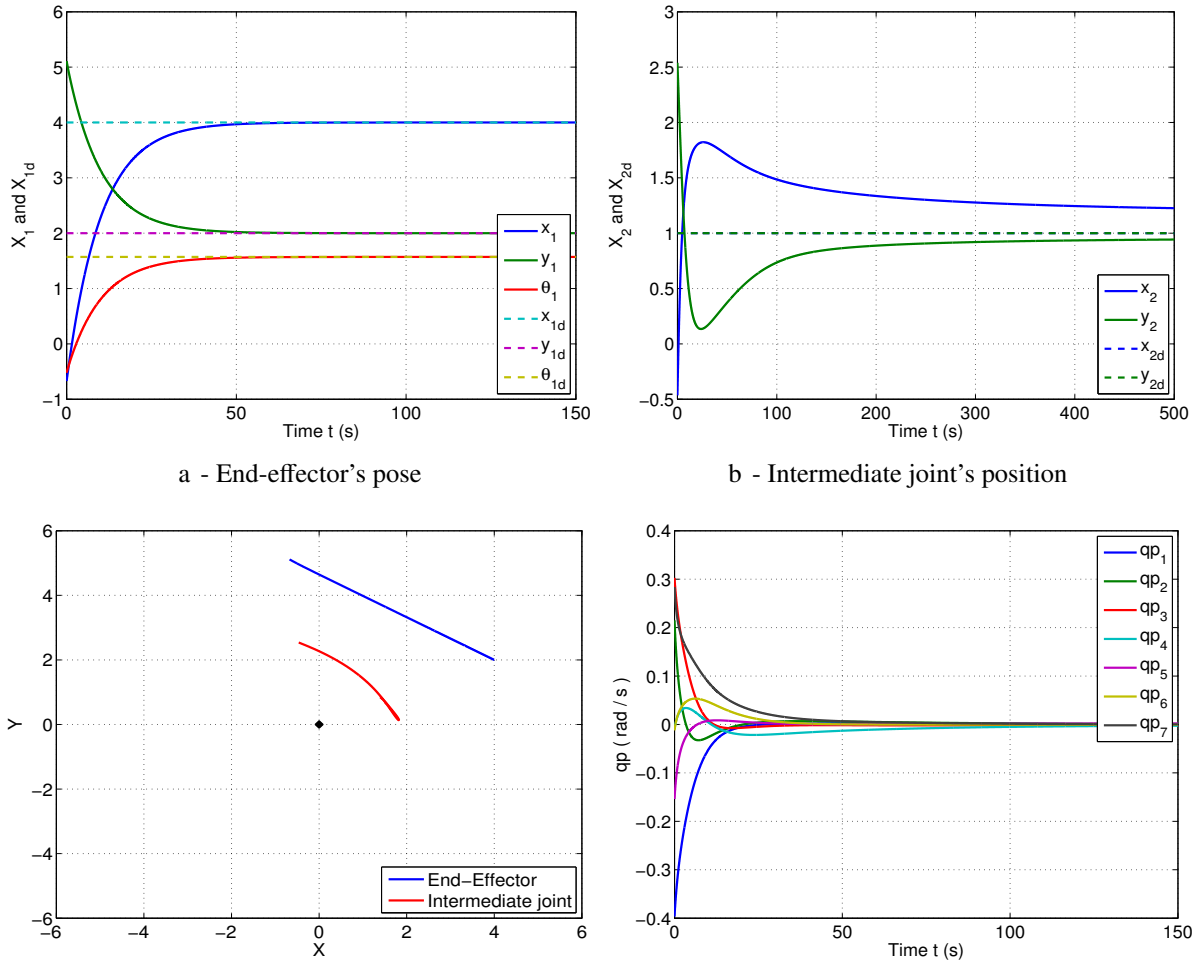
d - Weighting value

Figure III.35 – Simulation results on a 7R Planar robot with new projection $\mathbf{P}_\beta$

	Trajectory		Total Error Variation	Projector's Rank	Convergence Time		W_{ps}	E_c
	1st CP	2nd CP			t_{conv1}(s)	t_{conv2}(s)		
P_β	Non Linear	Non Linear	Non Exponential	$7 \rightarrow 4$	85	122	0.724	11.1
				Variable	Acceptable Time		1.116	

Table III.10 – Simulation results with the generalized projector P_β on the 7R planar robot (kinematically indeterminate system)

The same acceptable behavior is observed in the other cases (determinate and over-specified systems) also with an improvement of the convergence time of the secondary task. Furthermore, in case of unreachable and incompatible secondary tasks, acceptable behavior is noted for this projector.



a - End-effector's pose

b - Intermediate joint's position

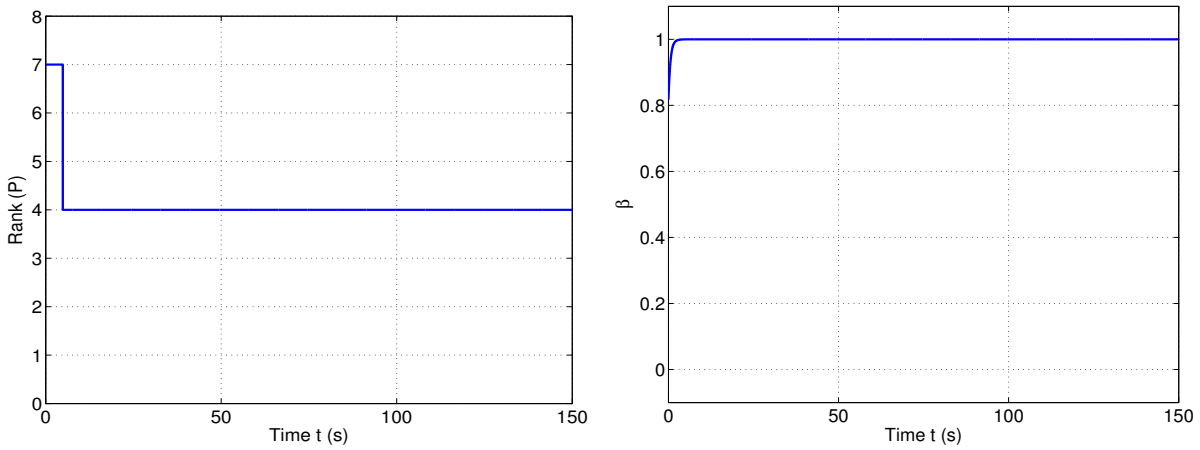
c - Control points trajectory in plane space

d - Joint velocities

Figure III.36 – Simulation results in case of unreachable secondary task with new projection P_β

Fig. III.36, shows the case of unreachable secondary task for example. The main task converges normally to the desired value and the secondary converges to the best reachable position with a stable behavior (Fig. III.36a-III.36b). An acceptable variation of joint velocities and trajectory of the control points is shown respectively in Fig. III.36c and Fig. III.36d.

The variation of projector's rank while performing these tasks from 7 to 4, and the variation of β value is shown also in Fig. III.37a-III.37b. The behavior of these elements ensures the feasibility and the stability of the defined projector in this case.



a - Projector's rank

b - Weighting value

Figure III.37 – (Cntl) Simulation results in case of unreachable secondary task with new projection $\mathbf{P}_\beta$

III.6.4 Conclusion on the Generalized Projector

The redundancy resolution method, defined in this section, overcomes the discontinuity problem which arise due to the projection into the singular values space. Thus, this generalized method gives a unique operator that integrates several existing techniques.

In addition, this projector allows the system to benefit from the advantages of the orthogonal projection method (by taking $\beta = 1$), especially the simplicity and exact regulation of the secondary task near convergence. Furthermore, it contains the hybrid control (for $\beta = 0$) by taking into account the secondary task. Finally, its a kind of directional projection of the secondary task with less restrictive constraints than the classical directional projection.

Simulation results, on the different cases, show an acceptable behavior of the generalized projector $\mathbf{P}_\beta$ and ensure the feasibility of the defined control law.

III.7 Conclusion

After the classification of the different types of redundancies and the presentation of several methods for kinematic redundancy resolution and task sequencing in the previous chapter, a comparison of these techniques is performed in this chapter using different types of planar robots and the 7 DOF LWR Kuka robot. Multiple configurations of the applied tasks are discussed using well defined comparison and performance criteria.

The theoretical study and the simulations results of the comparison shows the advantages and drawbacks of each method, in addition to the applications where each redundancy resolution technique can be suitably applied (paragraph III.3.6).

In addition, the directional projection method is useful in all cases except the non linear trajectory of the control point (although the desired exponential decrease of the error given in the control law), and except in case of a precise execution of the secondary task(s) is needed. Otherwise, the significance of this technique appears especially with over-specified systems, i.e there is an insufficient number of DOF to apply all tasks, and in case of incompatible tasks or unreachable lower priority tasks, in these cases the system converges normally and it is not disturbed. However, this method has several small inconveniences such as the difficulty of tuning projector's parameters, and the larger convergence time with respect to other methods.

Thus, the orthogonal projection method is more appropriate when the system has sufficient DOF to apply all the desired tasks, as long as they are well defined, reachable and compatible. This is due to the absence of parameters tuning, the linear control points trajectories and the acceptable convergence time for both main and secondary tasks.

To benefit from the advantages of these two methods in the different cases, a new projection operator was defined to control the contribution of the motions produced by the secondary task in the control law of the system; thus, it helps the main task to be completed faster by controlling the projection's direction of the additional task. Moreover, this projector uses a smooth transition function to pass between the directional projection (where all secondary task components are considered) and the classical orthogonal one near convergence to perform the exact regulation of the secondary task.

A problem in the directional projection method, and consequently in the defined projector, was noticed at the joint velocities level. A discontinuity appears in several cases due to the discontinuity of the space where the tasks errors are projected (space of singular values). This discontinuity leads to oscillations of the projector when passing between the different conditions in the control law, and thus vibration of the system.

Accordingly, a new simple projector was defined by considering the stability condition on the overall error vector without projection of its components into that space. Therefore, it allows a decrease in the number of parameters which should be tuned, and a perfect execution of the desired tasks in a acceptable time. The acceptable behavior of the defined projectors in simulation on the planar and LWR robots should be later confirmed on real robotic platforms.

The possibility to define other projection operators, such that the execution of the lower priority tasks will not perturb the higher priority tasks execution, is based on the use of other

Lyapunov functions to prove the system's stability. In addition to the classical function based on error norm, other functions could be defined to optimize different parameters, as the minimization of the error norm of several tasks simultaneously, but in all cases a continuous projection space should be defined to prevent the system's oscillations.

Other alternatives for redundancy resolution control are based on replacing the classical Moore-Penrose pseudo-inverse by other generalized inverse techniques using the dampest-least square inverse, matrix transpose, or other weighted pseudo-inverse techniques. Although these methods may be simpler and computationally cheaper than the classical pseudo-inverse, their use in redundancy resolution and task sequencing may lead to imprecise execution of the desired tasks.

IV

Kinematic and Dynamic Control of Multi-Arm Systems

Contents

As stated in [XTB96], the intelligence of a robotic system is characterized by three functional abilities. **Task Level control**: the robotic system should take task level commands directly without any planning or decomposition in joints level. **Generalization of the task**: the control systems should be designed for a large class of tasks that could be applied on various robotic systems. **Flexibility to environment changes**: the robotic system should be able to handle and integrate some unexpected or uncertain events.

According to that, a generic technique is necessary to control multi-arm robotic systems. Therefore, we develop, in this chapter, the multi control points approach at the kinematic and dynamic levels. In fact, it consists on controlling the robotic system by studying and monitoring the robot/environment interaction on several points using various types of sensors mounted in different configurations.

To manage the different types of redundancies that may arise in such systems, we will use the comparison results of redundancy resolution techniques from the previous chapters. On the other hand, a presentation and classification of the different types of sensors will be also addressed, especially the use of vision sensors in the control schema (visual servoing) where the selection of visual features and the calculation of the interaction matrices are discussed.

Furthermore, regardless the considered application, the desired scenario is decomposed into several simple tasks which are defined in a generic form to guarantee their ease of integration in all robotic systems. Therefore, several elementary tasks are discussed in this chapter as the positioning, cooperative and visibility tasks.

An extension of the kinematic multi control points approach to the dynamic level is also addressed in this chapter, where different control laws for redundancy resolution and task sequencing at the acceleration and torque level are discussed, in addition to the generic dynamic task definition.

The presented approaches will be used in the next chapter to perform various applications into several platforms as the robot's localization and object's manipulation.

IV.1 Kinematic Multi Control Points Approach

Since the applied tasks are normally specified in the operational space and require precise control of the end-effector motion, joint space control schemes are not suitable in these situations as presented in paragraph (I.3.2). This fact motivated different control approaches which are based on the definition of a task objective, such as the task function approach [SLBE91] and the operational space formulation [LWP80, Kha87].

In particular, these methods are essential when controlling the interaction between the robot and the environment. They also enable us to address the control problem directly in the sensor space, which closes the control loop and improves the control law robustness and accuracy [CH06]. Moreover, since the same task space is valid for a large set of robots, a control scheme based on the defined task is very portable and easy to modify and maintain. Therefore, it could be easily adapted from one robot structure to another and can be considered in advanced robotic architectures.

Furthermore, sensor-based control has long been recognized to provide precise manipulator control in the presence of environmental uncertainties, it controls the robot in a task-referenced space utilizing task specific sensors as part of the closed-loop manipulator control system. It overcomes many of the difficulties of uncertain models and unknown environments which limit the domain of application of robots used without external sensory feedback.

On the other hand, to control a redundant robot several approaches use the classical control laws which are based on the minimization of a task function, and tend to constrain all the degrees of freedom of a robot during the task execution. However, these approaches neither benefit from the robot redundancy in their control laws, nor took into account the robot-environment interaction. To overcome these drawbacks, we present the multi control points approach which is used for controlling such systems.

This approach is based on the definition of multiple control points, distributed on appropriate parts of the robot. For each control point, we study and control its interaction with the environment/robot using different proprioceptive and/or exteroceptive sensors data. From the study of these interactions, we define the possible task functions which control the robot behavior during the execution of the desired tasks. These task functions are then pushed into a structure which manages the priority and the compatibility between them using, for example, one of the methods presented in section II.4 and compared in the previous chapter.

The multi control points approach uses only the interaction, with the environment or the robot's body, of several control points on the robot, to manage and control the redundant systems. The first step is then to choose the appropriate control points before defining the constraints and the pertinent tasks on the robotic system using data from proprioceptive and exteroceptive sensors, for which we have to study as well their optimal number and position on the robotic system and the environment.

In the rest of this section, we define in paragraph (IV.1.1) the commonly used control points for redundant systems (especially for the humanoid robots); then we develop the control law which is applied to control such systems at the kinematic level (paragraph IV.1.2). Later on, we present the commonly used proprioceptive and exteroceptive sensors (paragraph IV.1.3). Finally we define in paragraph (IV.1.4) some generic tasks which are widely used in various robotic applications.

IV.1.1 Control Point Definition

Control points are chosen with respect to the robot's architecture and the desired tasks to be executed. The most common ones are the Center of Mass (CoM), the end-effectors points, and the contact points with the environment. The CoM control point is usually used to control the equilibrium and the posture of the robot. Those on the end-effectors of the system are used to define interaction tasks between the robot and the environment objects.

Furthermore, some control points on the robot's body can be used to avoid the self collision, or the collision with the environment. The control points which may be used to control an example of general multi-arm robot are marked by a dashed circle in Fig. IV.1.

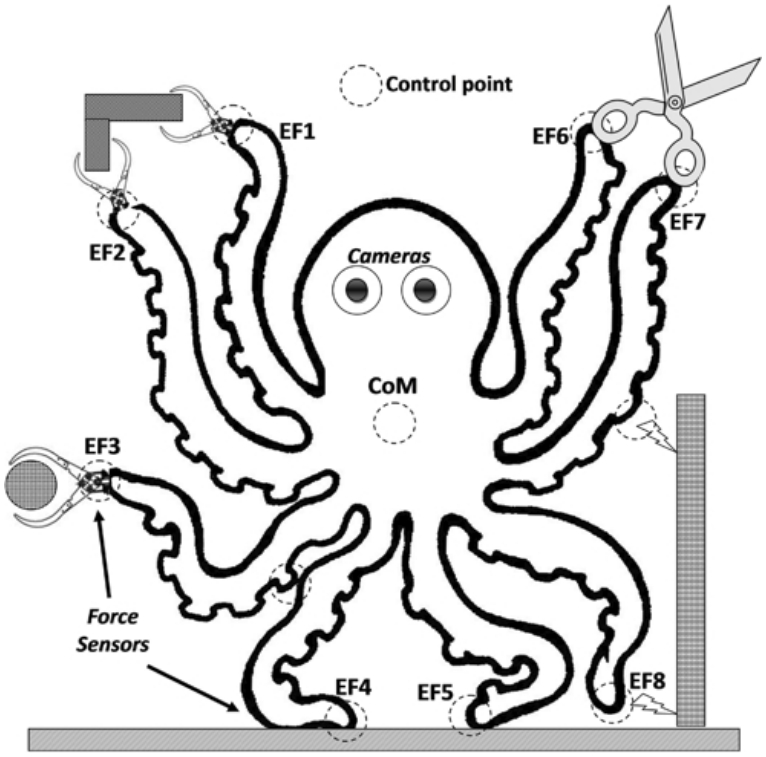


Figure IV.1 – Representation of a multi-arm redundant system (of 8 end-effectors) with the useful control points and exteroceptive sensors for multi-tasks execution

In case of humanoid robots, the most used control points are the robot's gripper's (especially in case of manipulation applications). A control point in the robot's head is also used when applying service tasks which need to control the robot's field of vision. Furthermore, control points on the robot's legs are extensively used for tasks which includes robot's locomotion, for example it is used to obtain a robust walking with active interaction between foot and irregular ground, or for the study of the Zero Moment Point (ZMP) effect on the stability of robot's standing and stepping. A control point can be also used to control the embedded exteroceptive sensors position, for example we can control, through a control point, the cameras positions in case of active vision applications.

IV.1.2 Control Technique

Regardless the used sensors, we define $\mathbf{s} \in \mathbb{R}^m$ the vector of the collected information. It can be a data given by a scalar sensor (as distance), or a data vector in case of more complex sensors as cameras and force sensors.

The desired value of this vector is $\mathbf{s}^*$, thus the error can be directly calculated by:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{IV.1}$$

Note that the error is expressed in the sensor's space $\mathcal{F}_s$ (not in the joint nor the operational space).

To control the system, we calculate the variation of the data vector using the kinematic relation below:

$$\dot{\mathbf{s}} = \mathbf{L_s}\ (\mathbf{v_s} - \mathbf{v_o}) \tag{IV.2}$$

where $\mathbf{L_s} \in \mathbb{R}^{m\times 6}$ is the sensor interaction matrix relative to sensor data $\mathbf{s}$,
$\mathbf{v_s}$ is the velocity tensor (linear and angular velocity) at the sensor's point expressed in $\mathcal{F}_\mathbf{s}$,
$\mathbf{v_o}$ is the velocity tensor at the target point.

If we consider a motionless target, thus $\mathbf{v_o} = 0$, and equation (IV.2) becomes:

$$\dot{\mathbf{s}} = \mathbf{L_s}\ \mathbf{v_s} \tag{IV.3}$$

To use the multi control points approach, it will be necessary to determine the variation of $\mathbf{s}$ with respect to the velocity at the control point $\mathbf{v_{cp}}$. Therefore, we consider that $\mathbf{v_s}$ and $\mathbf{v_{cp}}$ are related by:

$$\mathbf{v_s} = {}^{\mathbf{s}}\mathbf{W_{cp}}\ \mathbf{v_{cp}} \tag{IV.4}$$

where ${}^{\mathbf{s}}\mathbf{W_{cp}} \in \mathbb{R}^{6\times 6}$ is the twist transformation matrix between the control point frame $\mathcal{F}_\mathbf{cp}$ and the sensor frame $\mathcal{F}_\mathbf{s}$, it is given by:

$${}^{\mathbf{s}}\mathbf{W_{cp}} = \begin{bmatrix} {}^{\mathbf{s}}\mathbf{R_{cp}} & [{}^{\mathbf{s}}\mathbf{t_{cp}}]_\times\ {}^{\mathbf{s}}\mathbf{R_{cp}} \\ \mathbf{0_{3\times 3}} & {}^{\mathbf{s}}\mathbf{R_{cp}} \end{bmatrix} \tag{IV.5}$$

with ${}^{\mathbf{s}}\mathbf{R_{cp}} \in SO(3)$ and ${}^{\mathbf{s}}\mathbf{t_{cp}} \in \mathbb{R}^3$ are the rotation matrix and the translation vector from the control point frame $\mathcal{F}_\mathbf{cp}$ to the sensor frame $\mathcal{F}_\mathbf{s}$. The matrix ${}^{\mathbf{s}}\mathbf{W_{cp}}$ remains constant for sensor's fixed/embedded on the robot, while it has to be estimated at each iteration for external configurations.

Using (IV.3), (IV.4) and ${}^{\mathbf{cp}}\mathbf{J_q}$ the Jacobian of the robot expressed in the control point frame $\mathcal{F}_\mathbf{cp}$ given by the kinematic equation $\mathbf{v_{cp}} = {}^{\mathbf{cp}}\mathbf{J_q}\ \dot{\mathbf{q}}$, we can write:

$$\dot{\mathbf{s}} = \mathbf{L_s}\ {}^{\mathbf{s}}\mathbf{W_{cp}}\ {}^{\mathbf{cp}}\mathbf{J_q}\ \dot{\mathbf{q}} \tag{IV.6}$$

To reduce (IV.6), we define the sensor Jacobian $\mathbf{J_s}$ expressed as:

$$\mathbf{J_s} = \mathbf{L_s}\ {}^{\mathbf{s}}\mathbf{W_{cp}}\ {}^{\mathbf{cp}}\mathbf{J_q} \tag{IV.7}$$

Thus the feature variation will be given by:

$$\dot{\mathbf{s}} = \ \mathbf{J_s}\ \dot{\mathbf{q}} \tag{IV.8}$$

The motion is usually constrained to follow a differential equation: $\dot{\mathbf{e}} = -\lambda\ \mathbf{e}$. By inverting (IV.8) and using (IV.1), we can calculate the joint velocity for a motionless target ($\dot{\mathbf{e}} = \dot{\mathbf{s}}$). Thus, the general control law for sensor-based control used in multi-control points approach is given by:

$$\dot{\mathbf{q}} = -\lambda\ \mathbf{J_s^+}\ \mathbf{e}\ \ = \ -\lambda\ \mathbf{J_s^+}\ (\mathbf{s} - \mathbf{s}^*) \tag{IV.9}$$

and using (IV.7), it can be represented by:

$$\dot{\mathbf{q}} = -\lambda \, (\mathbf{L_s} \, {}^{\mathbf{s}}\mathbf{W_{cp}} \, {}^{\mathbf{cp}}\mathbf{J_q})^{+} \, (\mathbf{s} - \mathbf{s}^{*}) \tag{IV.10}$$

This general control law can be used for all types of sensors. Thus, to apply a desired task, the robot joint velocity can be calculated using (IV.10) after determining the following information:

— The actual ($\mathbf{s}$) and desired ($\mathbf{s}^{*}$) values of feature vector given by the sensor.

— The robot Jacobian (${}^{\mathbf{cp}}\mathbf{J_q}$) at the control point in $\mathcal{F}_{cp}$ frame.

— The twist transformation matrix (${}^{\mathbf{s}}\mathbf{W_{cp}}$) between the control point $\mathcal{F}_{cp}$ and sensor's frame $\mathcal{F}_{s}$.

— The sensor interaction matrix $\mathbf{L_s}$ corresponding to $\mathbf{s}$. It is given by equation (IV.3) which relates the variation of $\mathbf{s}$ to velocity tensor $\mathbf{v_s}$ at sensor point.

Finally, the different tasks are applied simultaneously using the most appropriate task sequencing formalism between the defined ones in the previous chapters. As discussed in section (III.3.6), the choice of the suitable method depends on the robot's architecture and the desired scenario to apply (which require a determinate number of DOF).

IV.1.3 Useful Sensors

In this part, we present a general classification of the classical sensors that are used in robotic systems. They could be divided into two main categories: **Proprioceptive sensors**, for the measurement of the robot's (internal) parameters; and **Exteroceptive sensors**, for the measurement of its environmental (external, from the robot point of view) parameters. In addition to that, the sensor-robot configurations are discussed in the third paragraph where the embedded and external sensor configurations are explored.

IV.1.3.1 Proprioceptive sensors

From a mechanical point of view, a robot appears as an articulated structure consisting of a series of links interconnected by joints. Each joint is driven by an actuator that can change the relative position of the two links connected by that joint. Proprioceptive sensors measure both kinematic and dynamic parameters of the robot. Based on these measurements the control system activates the actuators to exert torques so that the articulated mechanical structure performs the desired motion.

The usual kinematic parameters are the joint positions, velocities, and accelerations. Dynamic parameters as forces, torques and inertia are also important to monitor the proper control of the robotic manipulators. In addition to that, inertial units are present in several robots (especially humanoids), it uses gyrometers and accelerometers to estimate the robot speed and altitude.

IV.1.3.2 Exteroceptive sensors

Exteroceptive sensors measure the positional or force-type interaction of the robot with its environment. They can be classified according to their range as follows:

(a) Contact Sensors

The interaction forces and torques that appear at the robot end-effectors can be measured by force/torque sensors mounted on the joints or on the manipulator wrist. The first solution is not attractive since it requires a conversion of the measured joint torques to equivalent forces and torques at the hand level. The forces and torque measured by a wrist sensor can be converted quite directly at the end-effector level. The latter sensors are sensitive, small, compact and not too heavy, which is suitable for force controlled robotic applications.

Another type of contact sensors is the tactile one which measure a multitude of parameters of the touched object surface. It is defined as the continuous sensing of variable contact forces over an area within which there is a spatial resolution. Tactile sensing is more complex than touch sensing which usually is a simple vectorial force/torque measurement at a single point. Tactile sensors mounted on the fingers of the hand allow the robot to measure contact force profile and slippage, or to grope and identify object shape.

(b) Proximity Sensors

Proximity sensors detect nearby objects but without touching them. These sensors are used for near-field robotic operations (object approaching or avoidance). They are classified according to their operating principle: inductive, hall effect, capacitive, ultrasonic and optical sensors.

(c) "Far Away" Sensing

Two types of "far away" sensors are used in robotics: range sensors and vision. Range sensors measure the distance to objects in their operation area. They are used for robot navigation, obstacle avoidance or to recover the third dimension for monocular vision. Laser range finders and sonar are the best known sensors of this type.

Robot vision is a complex sensing process. It involves extracting, characterizing and interpreting information from images in order to identify or describe objects in the environment. A vision sensor converts the visual information to electrical signals which are then sampled and quantized by a special computer interface electronics yielding a digital image. CCDs and CMOS image sensors are increasingly used in robotics: CMOS image sensors consist of an array of pixels built from transistors and a photodiode. However, solid state CCD image sensors use a lens to direct light onto an array of capacitors which develop a charge that is proportional to the intensity of the incoming light.

Both image sensors can be used for most robotic applications, but there are advantages to both. The small lightweight CCD sensor gives higher quality images with little noise, typically more pixels, and higher sensitivity to light. However, it is easier to build a CMOS sensor which has a lower cost and a lower power consumption.

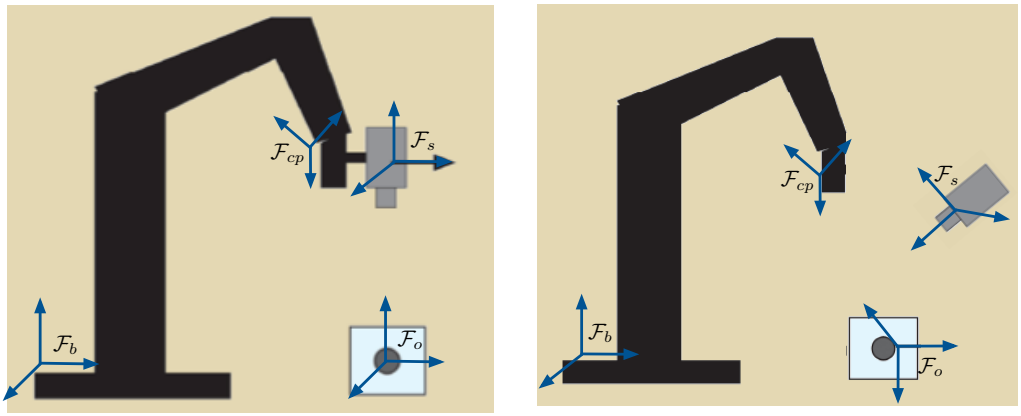
IV.1.3.3 Sensor-Robot configurations

There are two main configurations to combine the used sensor(s) with the robotic system for sensor based applications. The first configuration, called embedded (Fig. IV.2-a), consists in sensor(s) mounted on the robot's body (EF, head, foot, ...), it is attached to a control point of the robot. Control schemes are designed such that the velocity vectors $\mathbf{v_s}$ or $\dot{\mathbf{q}}$ are defined to control the required motion in the sensor space or in the robot's joints space respectively. This velocity vector is sent to the robot controller in order to perform the required motion. Thus sensor calibration, robot calibration and control point to sensor calibration are first needed in order to have the set of sensor intrinsic parameters, the robot Jacobian ${}^{\mathbf{cp}}\mathbf{J_q}$ and the constant transformation matrix ${}^{\mathbf{s}}\mathbf{W_{cp}}$ between the sensor frame $\mathcal{F}_s$ and the control point frame $\mathcal{F}_{cp}$.

In the second configuration, which is called external (Fig. IV.2-b), one or several sensors are placed in the workspace to monitor the desired control point (target object, end effector or both). Unlike the embedded configuration, where the sensor's value changes based on both the motion of the target and the motion of the attached control point; in external configuration, feature value changes based on the motion of the target only. In fact, if the robotic system is fixed, the transformation matrix ${}^{\mathbf{s}}\mathbf{W_b}$ between the sensor frame $\mathcal{F}_s$ and the base coordinate system of the robot frame $\mathcal{F}_b$ is constant and is computed once. Thus, this configuration needs to compute the transformation ${}^{\mathbf{s}}\mathbf{W_{cp}}$ between sensor's and control point's frames at each iteration.

The choice between the two configurations depends on the desired tasks to be executed. For example in case of humanoid robots, an appropriate use of the presented sensors is to fix them on the moving parts of the robotic system (end-effectors, foot ...) to obtain the maximum of information on the environment as perceived by the robot. The second configuration can also be used to obtain data about the position of the mobile parts of the robot from an external sensor mounted on the robot's environment.

Furthermore, the configuration influences the type of used sensors. For example, in case of using external configuration, fixed remote wide-angle cameras should be used to have a general view of the tasks scene, in contrary, for embedded configurations mobile narrow angle cameras should be used to distinguish the details of the task to be accomplished.



a - Embedded sensor configuration b - External sensor configuration

Figure IV.2 – Sensor-Robot configurations

IV.1.4 Task Definition

Each robotic scenario can be divided into a set of Generic Robotic Tasks that can be applied to several systems. Usually a desired task is defined at a specific point of the robot or the environment with a definite priority over the other tasks. Thus, each generic task can be defined as a triplet composed of: **task/constraint function, control point** and **priority level**. Our goal is to have a set of these predefined entities which are defined in a generic form to make this approach more portable and adaptable on a large number of redundant systems.

When applying a desired scenario, several tasks should be executed simultaneously on different parts of the system. One of the critical tasks is to maintain a stable equilibrium and/or a desired posture of the robot (especially humanoids). Another important task is the positioning of a control point to a desired location; this generic task can be used to apply several grasping and moving motions, or other interactive tasks between human/robot/environment. Additional tasks could be implemented to control some physical quantities due to the reaction between the robot and its environment (such as applied/received forces).

These generic tasks will be detailed later in section IV.3 after representing the case of using the vision sensors to control the system, known as visual servoing, in section IV.2.

IV.2 Visual Servoing

Since vision sensors are widely used in robotic systems to apply sensor-based control, we present in this section the application of the presented approach in case of vision based control. Thus, the different camera-robot configurations are presented, in addition to the used control laws with their corresponding interaction matrices.

Visual servoing is a well known flexible and robust technique to increase the accuracy and the versatility of a vision-based robotic system [WSN87, HHC96, CH06]. It consists in using vision in the feedback loop of a robotic system control with fast image processing to provide reactive behavior. Single or multiple cameras can be used to track visual information in order to control a robot with respect to a target [WSN87, HHC96, KC02b].

IV.2.1 Vision Sensor Configurations

Vision sensor can be conventional camera (as usually used in visual servoing), 2-D ultrasound camera [MKC10] or omni-directional cameras that is motivated in robotics applications to avoid visibility problems due to the restricted field of view of conventional camera [OJGJ02, HAMM07].

In the past few years, several 3D sensors showed up as the tilting laser scanner used on the PR2 robot to acquire high-quality 3D representations of the world point clouds. A cheap 3D sensor, which is widely used nowadays in several applications is the RGB-D Kinect sensor which provides real time point clouds as well as 2D images. In fact, it combines RGB color information with per-pixel depth information provided by an infrared sensor which projects an infrared speckle pattern.

The generic robotic task in visual servoing is thus specified in terms of image features extracted from a target object and their use for controlling the robot/camera motion through the

scene. Indeed, the visual features, which are extracted in real time from the image, are used to define a task function that depends on the robot's configuration and the time [SLBE91, ECR92, Has93].

In case of visual servoing, the two sensors configuration presented previously in paragraph (IV.1.3) are known as eye-in-hand (embedded) and eye-to-hand (external) configurations. The former configuration is commonly used in visual servoing since it allows us to keep the target(s) in the field of view (for example when a grasping task requires to ensure monitoring the object and the grasping tool attached to the end effector). Hybrid configurations can be constructed when eye-to-hand and eye-in-hand configurations are used [FCM00]. More details concerning camera/robot configurations can be found in [KC02b, CH07].

IV.2.2 PBVS and IBVS Approaches

Two basic approaches have been proposed namely position-based visual servoing (PBVS) and image-based visual servoing (IBVS) [WSN87, HHC96]. Both approaches have been used as bases for developing other schemes such as $2D^1/_2$ visual servoing [MCB99].

In position-based visual servoing (PBVS or 3D VS), the image measures (Fig. IV.3) are processed in order to estimate the relative 3D pose between the camera and the target, which is then used as an error signal for controlling the motion of the robot/camera system toward its desired goal [WWHB96, MDGD97]. 3D coordinates of points can also be used [MGK+96]. In image-based visual servoing (IBVS or 2D VS), the error is directly computed in terms of features expressed in image [WSN87, ECR92] (Fig. IV.4).

In [JSDW11], a comprehensive discussion of IBVS and PBVS is presented by comparing system stability and dynamic performance in the cartesian and image spaces on a common framework using both predefined and taught references. The robustness and sensitivity analyses are investigated with respect to all the camera, target, and robot modeling errors. Furthermore, other fundamental characteristics of the two methods, such as sensory task space singularity and local minima are also compared.

Two main aspects have a great impact on the behavior of any visual servoing scheme: the selection of the visual features used as input of the control law and the designed form of the control scheme. On one hand, the same feature set gives different behavior when employed in

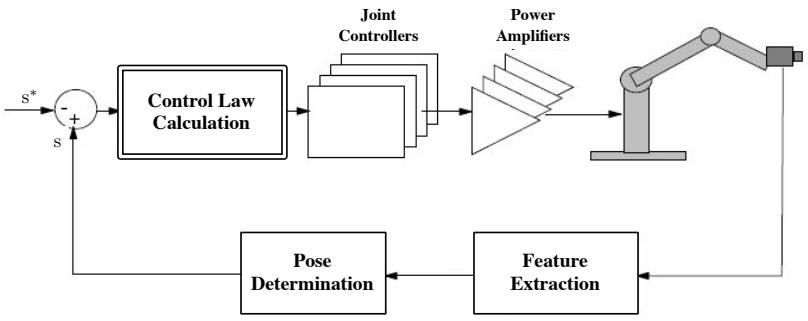


Figure IV.3 – Position based visual servoing schema

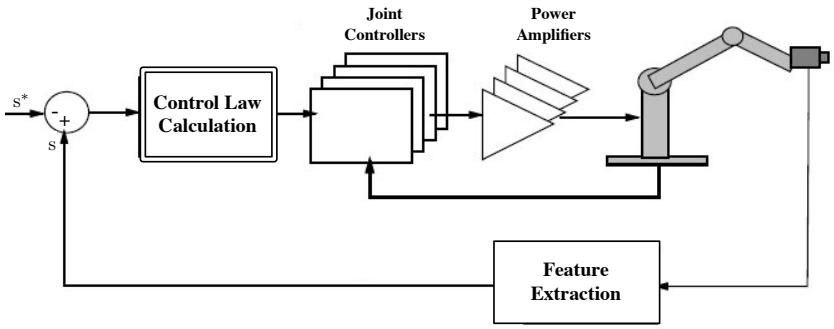


Figure IV.4 – Image based visual servoing schema

different control schemes and, on the other hand, the same control law gives different behavior when it considers different feature sets. The behavior obtained is not always the required one: selecting a specific feature set or a specific control scheme may lead to some stability and convergence problems.

IV.2.3 Selection of Visual Features

Visual features observed by the vision sensor define the inputs to the control scheme. A feature can be any property that represents a part of the scene and can be extracted from the image. Selection of good visual features is a crucial aspect of visual servoing as it is necessary for achieving optimal speed, accuracy, and reliability of image measurements, consequently, affects performance and robustness of visual servoing [FLM91, JSW97, Cha98].

As already noted, image measurements are either used directly in the control loop or used for relative pose estimation of a workpiece with respect to a camera. Thus, the number of DOF to be controlled by the employed control scheme determines the minimum number of independent features required. Therefore, it is desirable to choose features which will be highly correlated with movements of the camera. Image features are classified into three classes named geometric features, photometric features, and velocity field features.

Geometric features are defined to describe geometrically contents involved in a scene (2D visual features) or to relate a frame attached to a robot system with a frame attached to an object in a scene (3D visual features). Both 2D and 3D visual features can be used simultaneously in a hybrid form.

Recently, several studies focus on the utilization of photometric features computed from pixel intensities. Their utilization does not rely on complex image processing such as feature extraction, matching, and tracking process, contrary to utilizing geometric visual features such as points, straight lines, pose, homography, etc. Moreover, it is not very sensitive to partial occlusions and to coarse approximations of the depths required to compute the interaction matrix.

In case of velocity field features, the error is defined as the difference between the desired velocity field and the image feature velocity. Thus, instead of requiring the image feature to be at a specific location at each instant time as it is imposed in trajectory tracking control, in velocity field control the image feature is guided towards the desired flow defined by the desired velocity field i.e. the image feature will match with the flow lines of the desired velocity field.

IV.2.4 Interaction Matrices

The analytical form of the interaction matrix is based on the type of the used camera (conventional camera, 2D or 3D ultrasound cameras or omni-directional camera) and the used projection model [HHC96, FC09]. The most common geometric model for usual cameras is the perspective projection model [Hor86] (Fig. IV.5). In this model, the center of projection is considered at the origin of the camera frame $\mathcal{F}_c$, and the image plane is at $Z = f$, where f is the camera focal length.

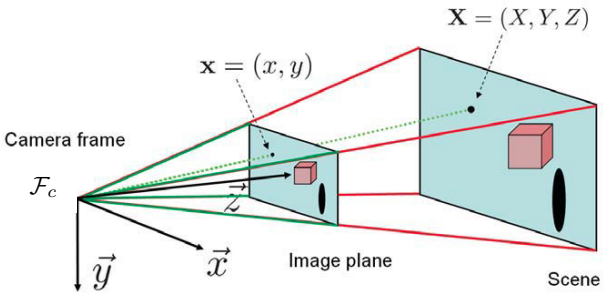


Figure IV.5 – Perspective projection camera model

By considering a 3D point with coordinates $\mathbf{X} = (X, Y, Z)^\top$ in the camera frame and using a perspective projection model [FP03], the point $\mathbf{X}$ is projected on a 2D point $\mathbf{x}$ of coordinates $(x, y)^\top$ in the normalized image plane such that:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} = \begin{bmatrix} (u - c_u)/f\alpha \\ (v - c_v)/f \end{bmatrix} \tag{IV.11}$$

where $\mathbf{x_p} = (u, v)^\top$ is the image point coordinates in pixel unit, $\mathbf{c} = (c_u, c_v)^\top$ is the coordinates of the principal point, f is the focal length of the camera lens and α is the ratio of pixel dimension.

IV.2.4.1 Interaction matrix of image feature points (IBVS)

The interaction matrix $\mathbf{L_s}$ of an image feature point can be obtained by taking the time derivative of (IV.11):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dot{X}/Z - X\dot{Z}/Z^2 \\ \dot{Y}/Z - Y\dot{Z}/Z^2 \end{bmatrix} = \begin{bmatrix} 1/Z & 0 & -X/Z^2 \\ 0 & 1/Z & -Y/Z^2 \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{IV.12}$$

If the spatial velocity of the camera is given by $\mathbf{v_c} = (v, \omega)^\top$ where $v = (v_x, v_y, v_z)^\top$ and $\omega = (\omega_x, \omega_y, \omega_z)^\top$ are the instantaneous linear and angular velocities of the origin of the camera frame both expressed in $\mathcal{F}_c$. Then the velocity of the 3D point $\mathbf{X}$ related to the camera velocity is defined using the fundamental kinematic equation $\dot{\mathbf{X}} = -v - \omega \times \mathbf{X}$ such that:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -v_x - \omega_y Z + \omega_z Y \\ -v_y - \omega_z X + \omega_x Z \\ -v_z - \omega_x Y + \omega_y X \end{bmatrix} = \begin{bmatrix} -\mathbf{I_3} & [\mathbf{X}]_\times \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{IV.13}$$

where $[\mathbf{X}]_\times$ is the skew symmetric matrix associated with $\mathbf{X}$ vector.

By injecting the values of $\dot{X}, \dot{Y}$ and $\dot{Z}$ from (IV.13) in (IV.12) and grouping for v, and ω we get the classical result [WSN87]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{IV.14}$$

which can be written as:

$$\dot{\mathbf{x}} = \mathbf{L_s}\, \mathbf{v_c} \tag{IV.15}$$

where $\mathbf{L_s}$ is the interaction matrix related to $\mathbf{s} = \mathbf{x} = (x, y)^\top$.

If there is a set of k feature points $\mathbf{S} = (\mathbf{s_1}, ..., \mathbf{s_k})$, the interaction matrix $\mathbf{L_S}$ of the set $\mathbf{S}$ is obtained by stacking $\mathbf{L_{s_i}}$ for all $\mathbf{s_i} \in \mathbf{S}$ to get:

$$\mathbf{L_S} = \begin{bmatrix} \mathbf{L_{s_1}} \\ \vdots \\ \mathbf{L_{s_k}} \end{bmatrix} \tag{IV.16}$$

In the matrix $\mathbf{L_s}$ given by (IV.14), Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of Z. Similarly, the camera intrinsic parameters are involved in the computation of x and y. Thus, $\mathbf{L_s}$ cannot be directly used, and an estimation or an approximation $\widehat{\mathbf{L_s}}$ must be used.

In general, there are several choices available for constructing the estimate $\widehat{\mathbf{L_s^+}}$ to be used in the control law (IV.10):

(a) Use of $\widehat{\mathbf{L_s^+}} = \mathbf{L_s^+}$
If $\mathbf{L_s}$ is known, that is, if the current depth Z of each point is available, these parameters can be estimated at each iteration of the control scheme. Referring to the experimental results of [CH06], trajectories of the points in the image are almost straight lines, but the behavior induced in 3D is not satisfactory (far from a straight line). The large camera velocities at the beginning of the servo indicate that the condition number of $\widehat{\mathbf{L_s^+}}$ is high at the start of the trajectory, and the camera trajectory is far from a straight line.

(b) Use of $\widehat{\mathbf{L_s^+}} = \mathbf{L_{s^*}^+}$
In this case, we consider $\mathbf{L_{s^*}}$ which is the value of $\mathbf{L_s}$ for the desired position $\mathbf{s}^*$, thus $\widehat{\mathbf{L_s^+}}$ is constant and only the desired depth of each point has to be set, which means no varying 3D parameters have to be estimated during the visual servo. Despite the large displacement that is required, the system converges. However, neither the behavior in the image nor the computed camera velocity components nor the 3D trajectory of the camera present desirable properties far from the convergence. Note that if the required displacement increases, the system converges more slowly.

(c) Use of $\widehat{\mathbf{L_s^+}} = 1/2\ (\mathbf{L_s} + \mathbf{L_{s^*}})^+$
Finally, this choice has been proposed in [Mal04]. Since $\mathbf{L_s}$ is involved in this method, the current depth of each point must also be available. Referring to the same simulations, this choice provides good performance. In fact, the camera velocity components do not include large oscillations and provide a smooth trajectory both in the image and in 3D space.

IV.2.4.2 Interaction matrix with uθ parameterization (PBVS)

We can define $\mathbf{s}$ to be $(\mathbf{t}, \mathbf{u}\theta)$, in which $\mathbf{t}$ is a translation vector, and $\mathbf{u}\theta$ gives the angle/axis parameterization for the rotation.

If the rotation matrix ${}^{c^*}\mathbf{R}_c \in SO(3)$ relating the two views of the camera is considered, thus $\mathbf{s}$ is defined to be $\mathbf{s} = [{}^{\mathbf{c}^*}\mathbf{t}_\mathbf{c}, \mathbf{u}\theta]$, in this case we have $\mathbf{s}^* = \mathbf{0}$, and the error function is given by $\mathbf{e} = \mathbf{s}$. The camera velocity related to the pose error is given by [MCB99]:

$$\dot{\mathbf{e}} = \begin{bmatrix} {}^{c^*}\mathbf{R}_c & 0 \\ 0 & {}^{c^*}\mathbf{R}_c \, \mathbf{L}_{\mathbf{u}\theta} \end{bmatrix} \mathbf{v}_c \tag{IV.17}$$

where $\mathbf{L}_{\mathbf{u}\theta} = \mathbf{I_3} - \frac{\theta}{\mathbf{2}}[\mathbf{u}]_\times + \left(\mathbf{1} - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^{\mathbf{2}}(\theta/\mathbf{2})}\right)[\mathbf{u}]_\times^2$ and $[\mathbf{u}]_\times$ is the skew symmetric matrix associated with $\mathbf{u}$.

The decoupling between translational and rotational motions, allows us to obtain a simple control scheme:

$$\begin{cases} v = -\lambda \, ({}^{\mathbf{c}^*}\mathbf{R}_\mathbf{c})^\top \, ({}^{\mathbf{c}^*}\mathbf{t}_\mathbf{c}) \\ \omega = -\lambda \, u\theta \end{cases} \tag{IV.18}$$

In this case, if the pose parameters involved in the last system of equations are perfectly estimated, the camera trajectory is a pure straight line, while the image trajectories are less satisfactory. Some particular configurations can even be found so that some points leave the camera field of view.

IV.2.4.3 Which one to use PBVS or IBVS?

As for the 3D parameters involved, a correct estimation is important in IBVS, since they appear in $\mathbf{L_s}$ and thus in the stability condition. Poor estimation can thus make the system unstable, but coarse estimations will only imply perturbations in the trajectory performed by the robot to reach its desired pose and will have no effect on the accuracy of the pose reached.

A correct estimation of the pose is crucial in PBVS, since it appears both in the error $\mathbf{e}$ to be regulated to $\mathbf{0}$ and in $\mathbf{L_s}$. Coarse estimations will thus induce perturbations on the trajectory realized but will have also an effect on the accuracy of the pose reached after convergence.

In fact, in PBVS, the vision sensor is considered as a 3D sensor. Since the control scheme imposes a behavior of $\mathbf{s}$, which is here expressed in the cartesian space, it allows the camera to follow theoretically an optimal trajectory in that space but generally not in the image space. When only one image is used, even small errors in the image measurements can lead to errors in the pose that can significantly impact the accuracy of the system.

On the other hand, in IBVS, the vision sensor is considered as a 2D sensor since the features are directly expressed in the image space. That is more realistic when a monocular camera is used, and this allows IBVS to be remarkably robust to errors in calibration and image noise.

However, IBVS is not without its shortcomings: when the displacement to realize is large, the camera may reach a local minimum or may cross a singularity of the interaction matrix. Furthermore, the camera motion may follow unpredictable, often suboptimal cartesian trajectories.

IV.3 Generic Robotic Tasks

As previously presented in paragraph IV.1.4, decomposing the desired application into several generic robotic tasks is a primary step of robot's control. Furthermore, each generic task can be defined as a triplet composed of: **task function, control point** and **priority level**. Thus, we develop in this section several essential generic tasks with their possible applications in the robotic manipulation framework.

The presented generic tasks will be used, in the experimental part (next chapter), to implement several service and assistive tasks on various robotic platforms.

IV.3.1 Positioning Task

The simplest generic task, that could be generalized to a large number of robotic tasks, is to move a control point from its actual pose to a desired position and/or orientation. The desired pose could be a fixed one (positioning task) or a point in motion (target following task).

Several examples can be cited as the case of grasping a fixed/mobile object by the robot's end-effector (gripper), and the robot shaping (i.e. moving the robot's body to a desired pose). It can be also used for navigation of mobile robots or humanoids, to move the entire system to a desired location.

For a simple grasping task, one control point on the end-effector is considered. The number of constrained DOF due to this task may vary in function of the gripper's and object's geometry: for example, only the position of the control point (3 DOF) is constrained in case of grasping a spherical ball, 4 DOF are constrained in the case of a cylindrical body, and generally 6 DOF (position and orientation) are constrained for grasping non-regular objects.

To remove the sensor/robot calibration problem and inverse kinematics imprecision, it is advisable to acquire 3D pose data ($\mathbf{s}$ and $\mathbf{s}^*$) about the end-effector and target object poses from exteroceptive sensors. Thus, as in (IV.10), the following control law can be applied using the articular Jacobian (${}^{\mathbf{cp}}\mathbf{J_q}$) at the gripper control point given by the robot's proprioceptive sensors, and the interaction and transformation matrices ($\mathbf{L_s}$, ${}^{\mathbf{s}}\mathbf{W_{cp}}$) corresponding to the used sensor:

$$\dot{\mathbf{q}} = -\lambda \, (\mathbf{L_s} \, {}^{\mathbf{s}}\mathbf{W_{cp}} \, {}^{\mathbf{cp}}\mathbf{J_q})^{+} \, (\mathbf{s} - \mathbf{s}^*) \tag{IV.19}$$

IV.3.2 Cooperative Task

An important issue in the redundant robots manipulation process is the cooperation between different end-effectors to execute a desired task, as for example long object displacement using the two robot's grippers. In this case, the task function is defined using two control points in order to control the relative configuration between them and maintain a desired relative position/orientation.

We define for this task the pose of the two control points $\mathbf{s_1}$ and $\mathbf{s_2}$ for $\mathbf{CP_1}$ and $\mathbf{CP_2}$ respectively. Thus, the relative pose of the second control point with respect to the first one is considered as controlled feature and is given by $\mathbf{s} = (\mathbf{s_2} - \mathbf{s_1})$, and the desired configuration is noted $\mathbf{s}^*$. Furthermore, since we control generally the relative pose, thus 6 DOF are constrained by this task.

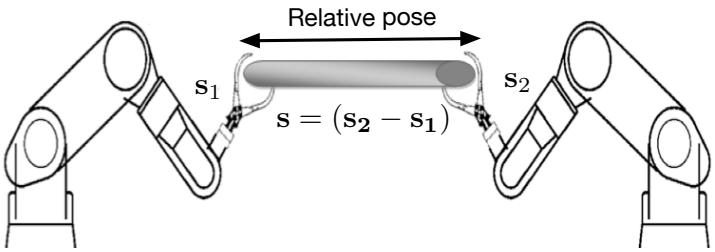


Figure IV.6 – Representation of the generic cooperative task

Let's consider the interaction matrices $\mathbf{L}_{s_1}$ and $\mathbf{L}_{s_2}$ which respectively relates the feature variation $\dot{\mathbf{s}}_1$ and $\dot{\mathbf{s}}_2$ with the velocity at the control points $\mathbf{CP_1}$ and $\mathbf{CP_2}$:

$$\dot{\mathbf{s}} = \dot{\mathbf{s}}_2 - \dot{\mathbf{s}}_1 = \mathbf{L}_{s_2}\,{}^{s_2}\mathbf{W}_{cp_2}\,\mathbf{v}_{cp_2} - \mathbf{L}_{s_1}\,{}^{s_1}\mathbf{W}_{cp_1}\,\mathbf{v}_{cp_1} = \begin{bmatrix} -\mathbf{L}_{s_1}\,{}^{s_1}\mathbf{W}_{cp_1} & \mathbf{L}_{s_2}\,{}^{s_2}\mathbf{W}_{cp_2} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{cp_1} \\ \mathbf{v}_{cp_2} \end{bmatrix}$$

Thus, the interaction matrix which corresponds to $\mathbf{s}$ is given by:

$$\mathbf{L}_s = \begin{bmatrix} -\mathbf{L}_{s_1}\,{}^{s_1}\mathbf{W}_{cp_1} & \mathbf{L}_{s_2}\,{}^{s_2}\mathbf{W}_{cp_2} \end{bmatrix} \tag{IV.20}$$

From the other side, the jacobian at the control points can be derived from $\mathbf{v}_{cp_1} = {}^{cp_1}\mathbf{J}_q\,\dot{\mathbf{q}}$ and $\mathbf{v}_{cp_2} = {}^{cp_2}\mathbf{J}_q\,\dot{\mathbf{q}}$, thus:

$$\mathbf{v}_{cp} = \begin{bmatrix} \mathbf{v}_{cp_1} \\ \mathbf{v}_{cp_2} \end{bmatrix} = \begin{bmatrix} {}^{cp_1}\mathbf{J}_q \\ {}^{cp_2}\mathbf{J}_q \end{bmatrix} \dot{\mathbf{q}} \tag{IV.21}$$

Therefore, using the presented features $\mathbf{s}$ and $\mathbf{s}^*$ with the interaction matrix in (IV.20) and the jacobian from (IV.21), we can calculate the control law for the cooperative task using the generic relation presented in (IV.10):

$$\dot{\mathbf{q}} = -\lambda \begin{bmatrix} -\mathbf{L}_{s_1}\,{}^{s_1}\mathbf{W}_{cp_1}\,{}^{cp_1}\mathbf{J}_q & \mathbf{L}_{s_2}\,{}^{s_2}\mathbf{W}_{cp_2}\,{}^{cp_2}\mathbf{J}_q \end{bmatrix}^+ (\mathbf{s} - \mathbf{s}^*) \tag{IV.22}$$

IV.3.3 Visibility Task

The problem of features visibility was widely addressed in the literature. Using classical 2D and 3D visual servoing and assuming a bad calibration with a large initial camera displacement, the target may leave the camera field of view [MCB99, CHPV04]. That is why, it is desirable to have servoing controls that are able to keep features in the camera field of view to obtain reliable feedback signal during the servoing process.

IV.3.3.1 State of the art

Several works attacked this problem: [MLS+00] developed the 2D½ visual servoing and used as visual information the center and the radius of the circle containing all points. In practice, when the object is large in the image, several points may be lost, and a commutative strategy is implemented. In [CH01] a partitioned strategy isolates the optical axis in the control

law. If a point approaches the edge of the image, the camera back automatically to keep it in the field of view. Unfortunately, this technique leads to many motions which are not optimal in terms of 3D path.

In [CHPV04], a commutative approach deactivates the linear or angular components of the PBVS controller. If the target approaches the edge of the image, the control law is partially applied to ensure the visibility. Another commutative approach is proposed by [GH07], where IBVS and PBVS schemas are used, and the control law switches between them as soon as the corresponding error becomes too large.

The switching criterion is based on the measurements error in the 2D and 3D spaces. However, this error is not directly related to the visibility constraint, it doesn't depend on the size of the image. Furthermore, to minimize the probability that the object leaves the field of view, other techniques can be adopted as the path planning strategy [MC02], the use of structured light [XLTP09], and the predictive control [ACC10].

IV.3.3.2 Task definition

In this paragraph, we address this problem as a task function: the visibility task. In fact, when applying a manipulation process on a robotic system that uses vision sensors; an important issue that should be considered is the visibility issue: the object and end-effector should stay in the field of view of the camera to not lose their tracking for example.

Generally, the used control point is the joint on which the camera is mounted. In case of humanoid robots, the head's joint is considered. The articular Jacobian ${}^{cp}\mathbf{J_q}$ at the considered control point is given by proprioceptive sensors.

Since the goal of this task is to keep some objects in the field of view of the vision system, the used feature vector $\mathbf{s}$ may change during tasks execution. In addition to the number of possible used DOF in this task, several modes of visibility can be considered in function of the number of objects which are in interaction. For example, when grasping an object by two hands; visibility task can be considered for one body (the object, gripper's center, ...), two bodies (midpoint of object-gripper or gripper-obstacle, ...) or even three bodies (point between object and the two grippers).

In general, the feature vector $\mathbf{s}$ is considered as the 3D pose of one of these points (noted m) and is obtained from the robot model and/or the exteroceptive sensors.

In the following, we consider the case of a humanoid robot, and we discuss the number of constrained DOF and the choice of the feature vectors in several cases that varies in function of the desired complexity and purpose of the visibility task.

(a) 2 DOF task using 2D point and depth visual feature

In this case, we control the robot's head so that the camera's optical axis passes through the desired point m. This case is the simplest one, it can be used in case of classical pan/tilt cameras for example (Fig. IV.7).

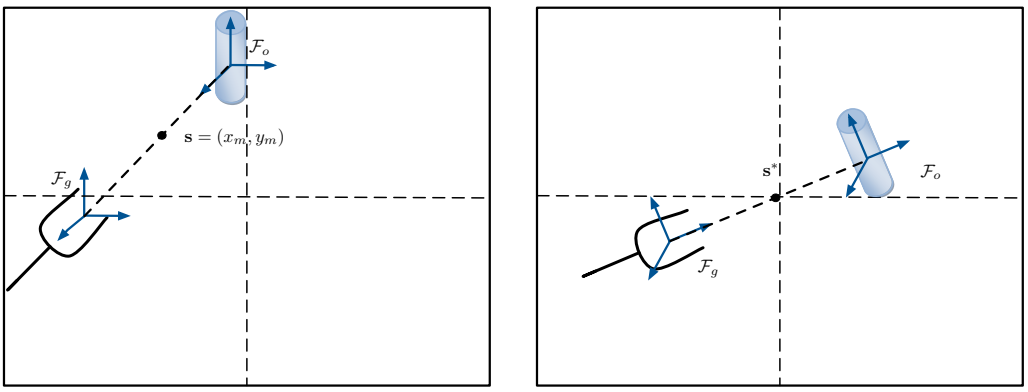


Figure IV.7 – 2 DOF visibility task

We consider the 2D pose of the tracked point $(x_m, y_m)^\top$ (projection of 3D point in the normal image plane) and its depth Z_m to build the feature vector $\mathbf{s} = (x_m, y_m)^\top$ and $\mathbf{L_s}$ is the same as the interaction matrix calculated in (IV.14):

$$\mathbf{L_s} = \begin{pmatrix} -1/Z_m & 0 & x_m/Z_m & x_m\, y_m & -\left(1+y_m^2\right) & y_m \\ 0 & -1/Z_m & y_m/Z_m & 1+x_m^2 & -x_m\, y_m & -x_m \end{pmatrix} \tag{IV.23}$$

(b) 3 DOF task using 2D point and depth visual feature

One more DOF can be added to have a better view of the items in the image, in other words, this task controls the depth of the object with respect to the camera and can be used, for example, in case of pan/tilt/zoom cameras.

In this case, we consider in addition to the 2D pose of the tracked point (x_m, y_m) the depth z_m to define the visibility task. Thus, $\mathbf{s} = (x_m, y_m, z_m)^\top$ and $\mathbf{s}^* = (0, 0, z^*)^\top$ where z^* is the desired distance of the object with respect to the camera image plane. Thus, the interaction matrix will be given by:

$$\mathbf{L_s} = \begin{pmatrix} -1/z_m & 0 & x_m/z_m & x_m\, y_m & -\left(1+x_m^2\right) & y_m \\ 0 & -1/z_m & y_m/z_m & 1+y_m^2 & -x_m\, y_m & -x_m \\ 0 & 0 & -1 & -y_m\, z_m & x_m\, z_m & 0 \end{pmatrix} \tag{IV.24}$$

(c) 4 DOF task using 2D segment visual feature

In this case, we consider a more complicated task that controls the orientation of the objects in the camera's image in addition to the control of their position in the center of the image, and their distance from the camera. To execute this task, the used visual feature is the segment formed by two points (not one point position as before): for example the segment between the robot's gripper and the object to grasp (Fig. IV.8), or between the left and right grippers.

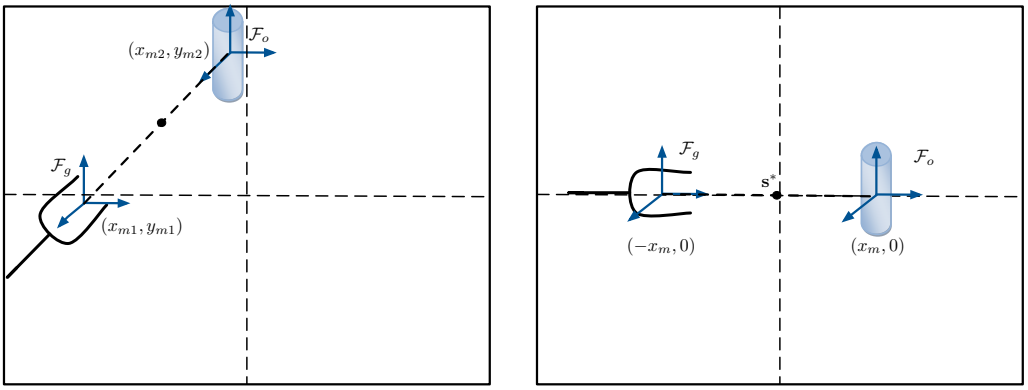


Figure IV.8 – 4 DOF visibility task using 2D segment visual feature

The goal is thus to put the segment in the center of field of view of the robot camera (2 DOF), to control the depth of the point with respect to the head (or in other words the projection of the distance between them in the camera image plane) (1 DOF) and to control the orientation of the segment (1 DOF).

Using the pose of the segment extremities (m_1 and m_2), we can calculate the feature vector $\mathbf{s} = (x_{m_1}, y_{m_1}, x_{m_2}, y_{m_2})^\top$. An appropriate feature value $\mathbf{s}^*$ should be considered to control the head orientation with respect to the segment, and the distance between the head and the segment. For example for a desired horizontal orientation, we can take $s^* = (-x_m^*, 0, x_m^*, 0)^\top$.

The corresponding interaction matrix is thus given by:

$$\mathbf{L_s} = \begin{pmatrix} -1/z_{m_1} & 0 & x_{m_1}/z_{m_1} & x_{m_1}\, y_{m_1} & -\left(1+x_{m_1}^2\right) & y_{m_1} \\ 0 & -1/z_{m_1} & y_{m_1}/z_{m_1} & 1+y_{m_1}^2 & -x_{m_1}\, y_{m_1} & -x_{m_1} \\ -1/z_{m_2} & 0 & x_{m_2}/z_{m_2} & x_{m_2}\, y_{m_2} & -\left(1+x_{m_2}^2\right) & y_{m_2} \\ 0 & -1/z_{m_2} & y_{m_2}/z_{m_2} & 1+y_{m_2}^2 & -x_{m_2}\, y_{m_2} & -x_{m_2} \end{pmatrix} \tag{IV.25}$$

(d) Another 4 DOF task using 2D segment visual feature

In the latter case, the orientation and the depth DOF are dependent, so cannot be controlled separately. Thus, we propose another 4 DOF visibility task which consists of controlling the center m of the segment, its length l and its orientation θ to a desired value (Fig. IV.9).

Considering the segment connecting the two operational points m_1 and m_2, the feature $\mathbf{s}$ will be given by:

$$\mathbf{s} = \begin{pmatrix} x_m \\ y_m \\ l \\ \theta \end{pmatrix} = \begin{pmatrix} (y_{m_1} + x_{m_2})/2 \\ (y_{m_1} + y_{m_2})/2 \\ \sqrt{(y_{m_1} - x_{m_2})^2 + (y_{m_1} - y_{m_2})^2} \\ \operatorname{atan}\left((y_{m_1} - y_{m_2})/(y_{m_1} - x_{m_2})\right) \end{pmatrix} \tag{IV.26}$$

For a centering task, a horizontal orientation and a predefined head-segment distance (controlled by l^*) are used to define the desired feature $\mathbf{s}^*$ which is given in this case by: $\mathbf{s}^* = (0, 0, l^*, 0)^\top$.

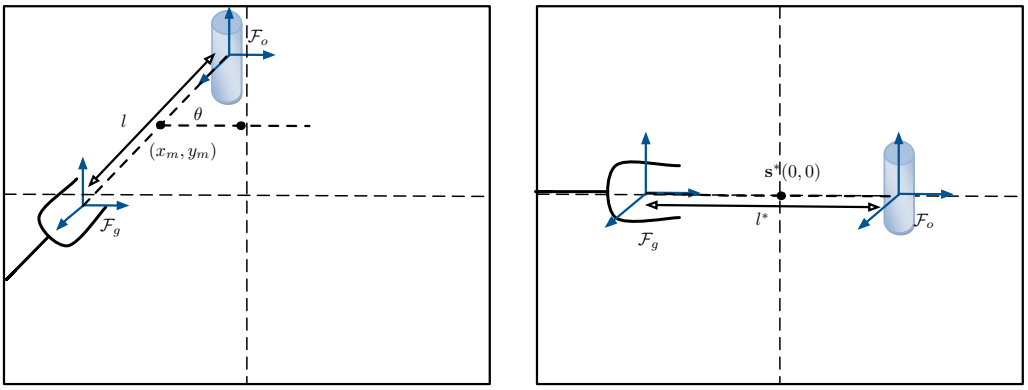


Figure IV.9 – Another 4 DOF visibility task using 2D segment visual feature

The interaction matrix is thus given by:

$$\mathbf{L_s} = \begin{bmatrix} a & b \end{bmatrix} \tag{IV.27}$$

where a and b are given by:

$$a = \begin{bmatrix} -\lambda_2 & 0 & \lambda_2 x_m - \lambda_1 \frac{l}{4} \cos\theta \\ 0 & -\lambda_2 & \lambda_2 y_m - \lambda_1 \frac{l}{4} \sin\theta \\ \lambda_1 \cos\theta & \lambda_1 \sin\theta & \lambda_2 l - \lambda_1 (x_m \cos\theta + y_m \sin\theta) \\ -\frac{\lambda_1}{l} \sin\theta & \frac{\lambda_1}{l} \cos\theta & \frac{\lambda_1}{l} (x_m \sin\theta - y_m \cos\theta) \end{bmatrix}$$

$$b = \begin{bmatrix} x_m y_m + \frac{l^2}{4} \cos\theta \sin\theta & -\left(1 + x_m^2 + \frac{l^2}{4} \cos^2\theta\right) & y_m \\ 1 + y_m^2 + \frac{l^2}{4} \sin^2\theta & -x_m y_m - \frac{l^2}{4} \cos\theta \sin\theta & -x_m \\ l\left(x_m \cos\theta \sin\theta + y_m \left(1 + \sin^2\theta\right)\right) & -l\left(y_m \cos\theta \sin\theta + x_m \left(1 + \cos^2\theta\right)\right) & 0 \\ y_m \cos\theta \sin\theta - x_m \sin^2\theta & x_m \cos\theta \sin\theta - y_m \cos^2\theta & -1 \end{bmatrix}$$

with $\lambda_1 = \frac{z_{m_1} - z_{m_2}}{z_{m_1} z_{m_2}}$ and $\lambda_2 = \frac{z_{m_1} + z_{m_2}}{2 z_{m_1} z_{m_2}}$

The defined visibility tasks differs in the number of constrained DOF, but the most relevant task definition is the last one which uses $\mathbf{s} = (x_m, y_m, l, \theta)^\top$ where 4 DOF are used to control the position and the orientation of the defined segment of the manipulation task in the camera frame.

After the presentation of several generic tasks that are widely used in several service tasks in case of kinematic control, i.e. when the robot is controlled with joint velocities, we extend in next sections the multi control points approach to the dynamic level.

IV.4 Dynamic Control of Multi-Arm Systems

The velocity-based approach has traditionally been preferred in many robotics applications due to its simplicity. However, for second-order systems, such as rigid body dynamics

systems, the acceleration-based approach is more appealing, especially when used in conjunction with an inverse dynamics control approach that explicitly needs knowledge of accelerations in joint space. Moreover, the obtained acceleration profiles can be directly used as reference signals (together with the corresponding positions and velocities) of a task-space dynamic controller. However, a second-order redundancy resolution scheme is invariably more demanding in terms of computational load.

Moreover, several studies show that, at the kinematic level, the difference between the high-level loop (measured by external sensors) and the low-level loop (joint measures) may make the control unstable at high speeds [VAC+02, ZLWS03]. In addition to that, the system's convergence time tends to infinity due to the exponential decay usually used in the kinematic control. Add to that the accumulation of the tracking errors, thus the kinematic control structure is not the most appropriate one to improve the performance of the sensor-based control. Therefore, a dynamic control should be considered where the joint accelerations/torques are directly calculated from sensors measurements.

In this section, the multi control points approach for multi-arm redundant systems is extended from the kinematic to the dynamic level. Thus, new generic tasks and control laws are presented to apply several prioritized tasks on a redundant system while taking into account the dynamic interaction between the system and the environment.

IV.4.1 Dynamic Model

The dynamic model of the robotic system can be expressed in joint space where joints torque is directly controlled to execute the desired motion. Another alternative is to control the system in the task space using the vector of forces and torques at the control point.

IV.4.1.1 Joint space control

The dynamic equation of a manipulator characterizes the relationship between its motion (position, velocity and acceleration) and the forces that cause these motions. The closed-form solution to this relationship is obtained through the Lagrangian or Newton-Euler equations and results in the following (without including additional torque components caused by friction, backlash and actuator dynamics):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \tau_{\mathbf{e}} = \tau \tag{IV.28}$$

where $\mathbf{q}$ is the $n \times 1$ vector of generalized coordinates consisting of the n joint angles,
$\mathbf{M}(\mathbf{q})$ is the $n \times n$ symmetric and positive definite inertia matrix,
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the $n \times 1$ Centrifugal and Coriolis force vector,
$\mathbf{G}(\mathbf{q})$ is the $n \times 1$ gravity loading vector,
τ is the $n \times 1$ torque vector, and
τ_e is the external torque applied on the robot such as $\tau_e = \mathbf{J}_s^\top \mathbf{h}_\mathbf{e}$ where $\mathbf{h}_\mathbf{e}$ is the vector of contact forces exerted on the manipulator by the environment including the forces applied on the body of the robot.

IV.4.1.2 Task space control

To achieve task space control, the joint space manipulator equation in (IV.28) can be rewritten as a function of the $m \times 1$ vectors of task space position $\mathbf{X}$, velocity $\dot{\mathbf{X}}$ and acceleration $\ddot{\mathbf{X}}$ as follows [Kha87]:

$$\mathbf{\Lambda}(\mathbf{X})\,\ddot{\mathbf{X}} + \mu(\mathbf{X}, \dot{\mathbf{X}}) + \mathbf{p}(\mathbf{X}) + \mathbf{h_e} = \mathbf{\Gamma} \tag{IV.29}$$

with

$$\begin{aligned}\mathbf{\Lambda}(\mathbf{X}) &= \mathbf{J}^{-\top}\,\mathbf{M}\,\mathbf{J}^{-1}\\ \mu(\mathbf{X}, \dot{\mathbf{X}}) &= \mathbf{J}^{-\top}\,\mathbf{C} \;-\mathbf{\Lambda}(\mathbf{X})\,\dot{\mathbf{J}}\,\dot{\mathbf{q}}\\ \mathbf{p}(\mathbf{X}) &= \mathbf{J}^{-\top}\,\mathbf{G}\end{aligned}$$

where $\mathbf{\Lambda}$ is the $n \times n$ task space inertia matrix,
μ is the $n \times 1$ task space Coriolis/Centrifugal vector,
$\mathbf{p}$ is the $n \times 1$ task space gravity vector,
$\mathbf{\Gamma}$ is the $n \times 1$ vector of forces and torques in the task space ($\mathbf{\Gamma} = \mathbf{J}^{-\top}\tau$) and
$\mathbf{J}(\mathbf{q})$ is the $6 \times n$ Jacobian matrix.

IV.4.2 Second-Order Redundancy Resolution

Regarding the space where the desired motion is specified, two classical possibilities arise: the joint space and the operational space (task space). However, due to several reasons, which are previously presented in the kinematic case, the task space control is more suitable when controlling the interaction between the manipulator and the environment is of concern. Since the desired task is often specified in the operational space and requires precise control of the end-effector motion, joint space control schemes are not suitable in these situations.

Furthermore, within this framework, redundancy can be resolved at the acceleration [HS87, Sen99, O'N02, ZCYZ12], and force levels [Kha87, FK97, OCB05, LZ11], where the desired joint accelerations, and torques are computed, respectively, for a desired end-effector acceleration, and force. In this part, several second-order solutions are discussed by considering different definitions of the Jacobian generalized inverse.

IV.4.2.1 Redundancy resolution at the acceleration level

Recalling the sensor-based technique and the task definition presented in section IV.1.2 at the kinematic level (equations IV.7 and IV.8):

$$\dot{\mathbf{e}} = \mathbf{J_s}\,\dot{\mathbf{q}} \tag{IV.30}$$

where $\mathbf{J_s}$ is the sensor Jacobian, expressed as $\mathbf{J_s} = \mathbf{L_s}\,{}^{\mathbf{s}}\mathbf{W_{cp}}\,{}^{\mathbf{cp}}\mathbf{J_q}$.

By differentiation of (IV.30), the second order kinematics equation gives the variation of the task error acceleration as:

$$\ddot{\mathbf{e}} = \mathbf{J_s}\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{s}\,\dot{\mathbf{q}} \tag{IV.31}$$

where $\dot{\mathbf{J}}_\mathbf{s}$ is the sensor Jacobian variation which can be obtained using several numerical differentiation methods or by direct analytic formulation for simple systems [Ahm92].

As seen in equation (II.12), the classical hierarchical redundancy resolution at the velocity level is achieved by rearranging (IV.30) to solve for $\dot{\mathbf{q}}$ in the following:

$$\dot{\mathbf{q}} = \mathbf{J}_s^{\#}\,\dot{\mathbf{e}} + \mathbf{P}\,\mathbf{z_1} \tag{IV.32}$$

where $\mathbf{z_1}$ is an arbitrary vector which controls the desired velocity behavior of a secondary task, and $\mathbf{P}$ represents a generalized projection matrix.

In the same manner, for redundancy resolution at the acceleration level, i.e. methods that directly compute joint accelerations from desired task space accelerations, (IV.31) is rearranged to solve for $\ddot{\mathbf{e}}$ in the following:

$$\ddot{\mathbf{q}} = \mathbf{J}_s^{\#}\left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s\,\dot{\mathbf{q}}\right) + \mathbf{P}\,\mathbf{z_2} \tag{IV.33}$$

where $\mathbf{z_2}$, similar to $\mathbf{z_1}$, is an arbitrary acceleration vector.

a - Case of Moore-Penrose pseudo-inverse

In case of considering the classical orthogonal projection ($\mathbf{P} = \mathbf{P_e}$) with the Moore-Penrose pseudo-inverse ($\mathbf{J}_s^{\#} = \mathbf{J}_s^{+}$), the vector $\mathbf{z_2}$ controls the desired acceleration behavior in the null space. Choosing $\mathbf{z_2} = \mathbf{0}$ in (IV.33) gives the simplest scheme operating at the acceleration level: the minimum norm acceleration solution, represented as:

$$\ddot{\mathbf{q}} = \mathbf{J}_s^{+}\left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s\,\dot{\mathbf{q}}\right) \tag{IV.34}$$

Similarly to the velocity-level pseudo-inverse solution, the joint movement generated by this locally optimal solution does not provide global acceleration minimization along the whole manipulator motion. More flexibility in the choice of (both local and global) performance criteria is obviously obtained by considering the full second-order solution:

$$\ddot{\mathbf{q}} = \mathbf{J}_s^{+}\left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s\,\dot{\mathbf{q}}\right) + (\mathbf{I_n} - \mathbf{J}_s^{+}\mathbf{J}_s)\,\mathbf{z_2} \tag{IV.35}$$

b - Case of inertia weighted pseudo-inverse

The inertia weighted pseudo inverse gives the solution that minimizes the instantaneous kinetic energy of the null space motion. It has been later proved by [FK97] that this is the only pseudo-inverse which is consistent with the manipulator dynamics, i.e. the null space motion does not produce the operational space acceleration. In this case, the jacobian inverse is given by:

$$\mathbf{J}_s^{\#} = \mathbf{J}_s^{\#\mathbf{M}} = \mathbf{M}^{-1}\mathbf{J}_s^{\top}(\mathbf{J}_s\mathbf{M}^{-1}\mathbf{J}_s^{\top})^{-1}$$

A weighted null space projection matrix is used in this case, and the control law will thus be given by:

$$\ddot{\mathbf{q}} = \mathbf{J}_s^{\#\mathbf{M}}\left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s\,\dot{\mathbf{q}}\right) + (\mathbf{I_n} - \mathbf{J}_s^{\#\mathbf{M}}\mathbf{J}_s)\,\mathbf{z_2} \tag{IV.36}$$

Such an inertia-matrix weighted pseudo-inverse has geometric significance since it endows the final task space with a kinetic energy metric defined on the tangent space of a selected manipulator. As noted in [NZ02], the use of such dynamically consistent pseudoinverse simplifies the process of decoupling the task and null-space dynamics.

IV.4.2.2 Redundancy resolution at the torque level

a - Case of inertia weighted pseudo-inverse

If the system is controlled at the torque level, the dynamic equation of the system as given by (IV.28) is considered to calculate the desired torque vector. In addition, using the second order differential kinematics equation (IV.31), applying the dynamic equation (IV.28) and multiplying by $\mathbf{J_s M^{-1}}$ gives:

$$\mathbf{J_s \ddot{q} + J_s M^{-1}(C + G + \tau_e) = J_s M^{-1} \tau} \tag{IV.37}$$

Substituting (IV.37) in (IV.31) gives:

$$\mathbf{J_s M^{-1} \tau = \ddot{e} - \dot{J}_s \dot{q} + J_s M^{-1}(C + G + \tau_e)}$$

To fit with the dynamics of the system, [Kha87] proposed to use the weighted pseudo-inverse with $\mathbf{M}$ as weights of the right hand side [DMB93], to minimize the acceleration pseudo energy ($\ddot{\mathbf{q}}^\top \mathbf{M} \ddot{\mathbf{q}}$) [PMU+07]:

$$\tau = \left(\mathbf{J_s M^{-1}}\right)^{\#\mathbf{M}} \left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s \dot{\mathbf{q}} + \mathbf{J_s M^{-1}(C + G + \tau_e)}\right) \tag{IV.38}$$

b - Case of transposed jacobian

Transposed jacobian (TJ) control is one of the simplest algorithms used to control motion of robotic manipulators. According to [Cra89], the transpose jacobian algorithm has been arrived at intuitively, and is similar to classic PD-action controllers. Apparently, the algorithm can be applied to redundant manipulators as shown by [ASY+93], and as discussed by [CCSS91] it does not fail when a singularity occurs.

$$\tau = \mathbf{J}^\intercal \left(\mathbf{\Lambda}\, \ddot{\mathbf{e}} + \mu + \mathbf{p} + \mathbf{h_e}\right)$$

[MP97] have compared the performance of this simple algorithm to those of various model-based algorithms. Both experimental and simulation results show the merits of the transposed jacobian algorithm in controlling of highly non-linear and complex systems with multiple DOF.

However, since the transposed jacobian is not dynamics-based, poor performance may result in fast trajectory tracking. Use of high gains can deteriorate performance seriously in the presence of feedback measurement noise.

Another drawback is that there is no formal method of selecting its control gains, and a heuristic selection of gains makes it difficult to apply. In [MP07], a Modified Transpose Jacobian algorithm is developed which employs stored data of the control command in the previous time step, as a learning tool to yield an improved performance.

IV.4.2.3 Discussion

The improved performance of the manipulator largely depends on the strategy used for the resolution of redundancy and the way they are implemented in the controller. Unfortunately, in

the majority of cases these schemes are computationally heavy and are difficult to implement in real-time. Additionally, they carry numerical instability, which makes their implementation more difficult.

In fact, the task space control is used either by defining the control law at the acceleration level than injecting it into the dynamic model of the robot, either by directly defining the dynamic control law at the torque level. In the second case, the dynamic model of the system is directly incorporated when defining and sequencing tasks which makes this control law more adequate and dynamically consistent.

In the next paragraph, the dynamic task sequencing will be presented. We will discuss the extension of the presented methods, at acceleration and torque levels, from the execution of one simple task to the execution of several task simultaneously.

IV.4.3 Dynamic Task Sequencing

The four task sequencing approaches, which are used to perform several prioritized tasks simultaneously in kinematic control (section II.2), can be extended and applied in the dynamic case. However, the extended and the commutative control laws will have the same limitations as previously presented.

Thus, the hierarchical and hybrid control are the most adequate to apply in the dynamic case. In the first approach, the classical projection of the low priority tasks in the null space of the higher priority ones is used. For the second one, the different tasks are stacked together into an extended task, and a dynamic weighting matrix is used to manage the priority between them.

Concerning the hierarchical control, the classical orthogonal projection is the mostly used one, it allows applying secondary tasks without disturbing the regulation of the main one. The other projection operators, which are previously presented in case of kinematic control, could not be used for dynamic control.

For example, the directional projection method (section II.3.2) consider projector's definition which is based on specific variation of the Lyapunov function that is not applicable in the dynamic case. Moreover, the norm projection method (section II.3.3) could not be applied in this case.

IV.4.3.1 Hierarchical control at the acceleration level

The goal of the kinematic task sequencing is to find a formalism to prioritize and execute simultaneous tasks which are executed on multiple control points of a robot to perform a desired behavior. However, when the system is velocity controlled, the dynamic change of the tasks and the relationships between control points are not taken into account; furthermore, the internal dynamics of the system is not considered. Therefore, a task sequencing formalism should be defined at the acceleration level to apply several generic tasks simultaneously.

Similarly to task sequencing at the kinematic level (section II.4.2), two possibilities arise in dynamic control: the first one, named classical method, consists into stacking the prioritized

tasks successively which results into inexact regulation of the secondary tasks, and the second one, named efficient method, guaranties their exact execution by taking into consideration the part of the secondary task which is executed by the main one.

Each generic task is defined by ($\ddot{\mathbf{e}}_\mathbf{i}$, $\dot{\mathbf{e}}_\mathbf{i}$, $\dot{\mathbf{J}}_\mathbf{i}$, $\mathbf{J}_\mathbf{i}$) where $\dot{\mathbf{e}}_\mathbf{i}$ and $\ddot{\mathbf{e}}_\mathbf{i}$ are respectively the desired error velocity and acceleration variation of the task $\mathbf{T}_i$ and $\dot{\mathbf{J}}_\mathbf{i}$ is the Jacobian variation. Thus, the second order differential kinematics equation of the task $\mathbf{T}_i$ is generally given by:

$$\ddot{\mathbf{e}}_\mathbf{i} = \mathbf{J}_\mathbf{i}\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{i}\,\dot{\mathbf{q}} \tag{IV.39}$$

(a) Classical task sequencing

Applying (IV.39) on task $\mathbf{T}_1$ gives: $\ddot{\mathbf{e}}_\mathbf{1} = \mathbf{J}_\mathbf{1}\,\ddot{\mathbf{q}}_\mathbf{1} + \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}_\mathbf{1}$, thus:

$$\ddot{\mathbf{q}}_\mathbf{1} = \mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}_\mathbf{1}\right) \tag{IV.40}$$

If we consider the two tasks $\mathbf{T}_1$ and $\mathbf{T}_2$, the system's acceleration is given by:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_\mathbf{1} + \mathbf{P}_\mathbf{1}\,\mathbf{z} \tag{IV.41}$$

such as $\mathbf{z}$ is an arbitrary control vector used to apply a secondary task without disturbing the realization of the main objective.

The classical task sequencing method consider the secondary task as: $\mathbf{z} = \mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right)$, and the final control law will be:

$$\ddot{\mathbf{q}}_\mathbf{classical} = \mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}\right) + \mathbf{P}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) \tag{IV.42}$$

where $\dot{\mathbf{q}}$ is the joint velocity corresponding to the execution of two tasks (calculated using the kinematic task sequencing).

In case of using the classical orthogonal projection ($\mathbf{P}_\mathbf{1} = \mathbf{I}_\mathbf{n} - \mathbf{J}_\mathbf{1}^{+}\mathbf{J}_\mathbf{1}$), it can be easily verified that the main task is fully regulated to the desired value on the contrary of the secondary one:

$$\begin{aligned}
\ddot{\mathbf{e}}_{\mathbf{1}/\ddot{\mathbf{q}}_\mathbf{classical}} &= \mathbf{J}_\mathbf{1}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}} \\
&= \mathbf{J}_\mathbf{1}\mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}\right) + \mathbf{J}_\mathbf{1}\mathbf{P}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) + \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}} \\
&= \ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}} \\
&= \ddot{\mathbf{e}}_\mathbf{1}
\end{aligned}$$

$$\begin{aligned}
\ddot{\mathbf{e}}_{\mathbf{2}/\ddot{\mathbf{q}}_\mathbf{classical}} &= \mathbf{J}_\mathbf{2}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}} \\
&= \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}\right) + \mathbf{J}_\mathbf{2}\mathbf{P}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) + \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}} \\
&= \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}\right) + \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) + \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\mathbf{J}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) + \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}} \\
&= \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}}\right) + \ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}} - \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\mathbf{J}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}\left(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}}\right) + \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}} \\
&= \ddot{\mathbf{e}}_\mathbf{2} + \mathbf{J}_\mathbf{2}\mathbf{J}_\mathbf{1}^{+}\left(\ddot{\mathbf{e}}_\mathbf{1} - \dot{\mathbf{J}}_\mathbf{1}\,\dot{\mathbf{q}} - \mathbf{J}_\mathbf{1}\mathbf{J}_\mathbf{2}^{+}(\ddot{\mathbf{e}}_\mathbf{2} - \dot{\mathbf{J}}_\mathbf{2}\,\dot{\mathbf{q}})\right)
\end{aligned}$$

(b) Efficient task sequencing

In this case, the secondary task vector $\mathbf{z}$ is calculated by applying (IV.39) on $\mathbf{T}_2$ and using equation (IV.41) to obtain:

$$\ddot{\mathbf{e}}_2 = \mathbf{J}_2\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} = \mathbf{J}_2\,\ddot{\mathbf{q}}_1 + \mathbf{J}_2\,\mathbf{P}_1\,\mathbf{z} + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}}$$

and using (IV.41), the value of $\mathbf{z}$ vector is found as:

$$\mathbf{z} = (\mathbf{J}_2\,\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \mathbf{J}_2\,\ddot{\mathbf{q}}_1 - \dot{\mathbf{J}}_2\,\dot{\mathbf{q}}\right)$$

Substituting $\mathbf{z}$ into (IV.41), we find:

$$\ddot{\mathbf{q}}_{\text{efficient}} = \mathbf{J}_1^+\left(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}}\right) + \mathbf{P}_1(\mathbf{J}_2\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \mathbf{J}_2\ddot{\mathbf{q}}_1 - \dot{\mathbf{J}}_2\dot{\mathbf{q}}\right)$$

In the case of using the classical orthogonal projection ($\mathbf{P}_1 = \mathbf{I}_n - \mathbf{J}_1^+\mathbf{J}_1$), the projector is idempotent and Hermetian, thus $\mathbf{P}_1(\mathbf{J}_2\mathbf{P}_1)^+ = \mathbf{J}_2\mathbf{P}_1$.

Finally, the control law, in case of two tasks, will be given by:

$$\ddot{\mathbf{q}}_{\text{efficient}} = \ddot{\mathbf{q}}_1 + (\mathbf{J}_2\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\ddot{\mathbf{q}}_1\right) \tag{IV.43}$$

with $\ddot{\mathbf{q}}_1 = \mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}})$.

In case of using the classical orthogonal projector, it can be easily verified that the main and secondary tasks are regulated to the desired values when using this method:

$$\begin{aligned}
\ddot{\mathbf{e}}_{1/\ddot{\mathbf{q}}_{\text{efficient}}} &= \mathbf{J}_1\ddot{\mathbf{q}} + \dot{\mathbf{J}}_1\,\dot{\mathbf{q}} \\
&= \mathbf{J}_1\ddot{\mathbf{q}}_1 + \mathbf{J}_1(\mathbf{J}_2\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\ddot{\mathbf{q}}_1\right) + \dot{\mathbf{J}}_1\,\dot{\mathbf{q}} \\
&= \mathbf{J}_1\mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}}) + \dot{\mathbf{J}}_1\,\dot{\mathbf{q}} \\
&= \ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}} + \dot{\mathbf{J}}_1\,\dot{\mathbf{q}} \qquad \text{(if } \mathbf{J}_1 \text{ is full rank)} \\
&= \ddot{\mathbf{e}}_1
\end{aligned}$$

$$\begin{aligned}
\ddot{\mathbf{e}}_{2/\ddot{\mathbf{q}}_{\text{efficient}}} &= \mathbf{J}_2\ddot{\mathbf{q}} + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} \\
&= \mathbf{J}_2\ddot{\mathbf{q}}_1 + \mathbf{J}_2(\mathbf{J}_2\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\ddot{\mathbf{q}}_1\right) + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} \\
&= \mathbf{J}_2\mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}}) + \mathbf{J}_2\mathbf{P}_1(\mathbf{J}_2\mathbf{P}_1)^+\left(\ddot{\mathbf{e}}_2 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}})\right) + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} \\
&= \mathbf{J}_2\mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}}) + \ddot{\mathbf{e}}_2 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\mathbf{J}_1^+(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\,\dot{\mathbf{q}}) + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} \\
&= \ddot{\mathbf{e}}_2 \qquad \text{(if } \mathbf{J}_2\mathbf{P}_1 \text{ is full rank)}
\end{aligned}$$

(c) Case of several tasks

In [SS91], the recursive form is expressed in terms of joint acceleration. Instead of deriving joint acceleration solutions from the second-order differential kinematics and then substituting in the robot dynamic model for designing a computed torque control, which may lead to internal unstable behavior for a redundant manipulator [NHY87], a dynamic

controller scheme can be derived from the joint velocity solution by direct differentiation with respect to time [KW88, SS91] by taking the derivative of (II.42):

$$\ddot{\mathbf{q}}_\mathbf{i} = \begin{cases} \ddot{\mathbf{q}}_{\mathbf{i-1}} + \bar{\mathbf{J}}_\mathbf{i}^{+}\left(\ddot{\mathbf{e}}_\mathbf{i} - \dot{\mathbf{J}}_\mathbf{i}\,\dot{\mathbf{q}}_{\mathbf{i-1}} - \mathbf{J}_\mathbf{i}\,\ddot{\mathbf{q}}_{\mathbf{i-1}}\right) + \dot{\bar{\mathbf{J}}}_\mathbf{i}^{+}\left(\dot{\mathbf{e}}_\mathbf{i} - \mathbf{J}_\mathbf{i}\,\dot{\mathbf{q}}_{\mathbf{i-1}}\right) & if \quad i = 2, \cdots k. \\ \mathbf{J}_\mathbf{1}^{+}\,\ddot{\mathbf{e}}_\mathbf{1} + \dot{\mathbf{J}}_\mathbf{1}^{+}\,\dot{\mathbf{e}}_\mathbf{1} & if \quad i = 1 \end{cases} \tag{IV.44}$$

where $\bar{\mathbf{J}}_\mathbf{i}$ and $\dot{\bar{\mathbf{J}}}_\mathbf{i}^{+}$ are given by:

$$\begin{aligned} \bar{\mathbf{J}}_\mathbf{i} &= \mathbf{J}_\mathbf{i}\mathbf{P}_{\mathbf{i-1}}^{\mathbf{A}} \\ \dot{\bar{\mathbf{J}}}_\mathbf{i}^{+} &= \bar{\mathbf{J}}_\mathbf{i}^{+}\dot{\mathbf{J}}_\mathbf{i}\bar{\mathbf{J}}_\mathbf{i}^{+} + \left(\mathbf{I} - \bar{\mathbf{J}}_\mathbf{i}^{+}\bar{\mathbf{J}}_\mathbf{i}\right)\dot{\mathbf{J}}_\mathbf{i}^{\top}\left(\bar{\mathbf{J}}_\mathbf{i}\bar{\mathbf{J}}_\mathbf{i}^{\top}\right)^{-1} \\ \mathbf{P}_\mathbf{i}^{\mathbf{A}} &= \begin{cases} \mathbf{P}_{\mathbf{i-1}}^{\mathbf{A}} - \left(\mathbf{J}_\mathbf{i}\mathbf{P}_{\mathbf{i-1}}^{\mathbf{A}}\right)^{+}\left(\mathbf{J}_\mathbf{i}\mathbf{P}_{\mathbf{i-1}}^{\mathbf{A}}\right) & if \quad i = 2, \cdots k. \\ \mathbf{I} - \mathbf{J}_\mathbf{1}^{+}\mathbf{J}_\mathbf{1} & if \quad i = 1 \end{cases} \end{aligned} \tag{IV.45}$$

IV.4.3.2 Hierarchical control at the torque level

In this part, we consider that the system is controlled at the torque level, and consider the dynamic equation of the system as given by (IV.28).

Applying the developed control law in equation (IV.38) into task $\mathbf{T}_1$, we get:

$$\tau_1 = \left(\mathbf{J}_1\mathbf{M}^{-1}\right)^{\#\mathbf{M}}\left(\ddot{\mathbf{e}}_1 - \dot{\mathbf{J}}_1\dot{\mathbf{q}}_1 + \mathbf{J}_1\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e})\right) \tag{IV.46}$$

For two tasks $\mathbf{T}_1$ and $\mathbf{T}_2$, the applied actuator torque is given by:

$$\tau = \tau_1 + \mathbf{P}_1\mathbf{z} \tag{IV.47}$$

Applying the second order differential kinematics equation (IV.39) gives:

$$\ddot{\mathbf{e}}_2 = \mathbf{J}_2\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} \tag{IV.48}$$

The dynamic equation (IV.28) applied on the system, then multiplied by $\mathbf{J}_2\mathbf{M}^{-1}$ gives:

$$\mathbf{J}_2\,\ddot{\mathbf{q}} + \mathbf{J}_2\mathbf{M}^{-1}\mathbf{C} = \mathbf{J}_2\mathbf{M}^{-1}\tau \tag{IV.49}$$

Substituting (IV.49) in (IV.48), and using (IV.47) gives:

$$\begin{aligned} \ddot{\mathbf{e}}_2 &= \mathbf{J}_2\mathbf{M}^{-1}\tau + \dot{\mathbf{J}}_2\,\dot{\mathbf{q}} - \mathbf{J}_2\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e}) \\ &= \mathbf{J}_2\mathbf{M}^{-1}\tau_1 + \mathbf{J}_2\mathbf{M}^{-1}\mathbf{P}_1\mathbf{z} + \dot{\mathbf{J}}_2\dot{\mathbf{q}} - \mathbf{J}_2\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e}) \end{aligned}$$

Thus:

$$\left(\mathbf{J}_2\mathbf{M}^{-1}\right)\mathbf{P}_1\mathbf{z} = \ddot{\mathbf{e}}_2 - \mathbf{J}_2\mathbf{M}^{-1}\tau_1 - \dot{\mathbf{J}}_2\dot{\mathbf{q}} + \mathbf{J}_2\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e})$$

Using the weighted pseudo-inverse with $\mathbf{M}$ as weight of the right hand side, to minimize the overall acceleration pseudo energy ($\ddot{\mathbf{q}}^\top \mathbf{M} \ddot{\mathbf{q}}$), we get:

$$\mathbf{P_1 z} = \left(\mathbf{J_2 M^{-1}}\right)^{\#\mathbf{M}} \left(\ddot{\mathbf{e}}_2 - \mathbf{J_2 M^{-1}} \tau_1 - \dot{\mathbf{J}}_2 \dot{\mathbf{q}} + \mathbf{J_2 M^{-1}} (\mathbf{C} + \mathbf{G} + \tau_e)\right)$$

And the final control law will be:

$$\tau = \tau_1 + \mathbf{P_1 z} = \tau_1 + \left(\mathbf{J_2 M^{-1}}\right)^{\#\mathbf{M}} \left(\ddot{\mathbf{e}}_2 - \mathbf{J_2 M^{-1}} \tau_1 - \dot{\mathbf{J}}_2 \dot{\mathbf{q}} + \mathbf{J_2 M^{-1}} (\mathbf{C} + \mathbf{G} + \tau_e)\right) \quad \text{(IV.50)}$$

with τ_1 given by (IV.46) and $\dot{\mathbf{q}}$ the joint velocity corresponding to the execution of two tasks (found using the kinematic task sequencing).

Finally, the hierarchical control using joints torque gives dynamically consistent results. Using the decomposition in (IV.50), the end-effector can be controlled by the task space command torque, whereas internal motions can be independently controlled by the joint torques that are guaranteed not to change the dynamic behavior of the end-effector.

IV.4.3.3 Hybrid control

Using the same definitions of the extended task and the extended jacobian as in the kinematic control (section II.2.3), this formalism can be also applied in the dynamic case to assemble several tasks together and to introduce additional constraints or tasks into the initial system. Therefore, the task-space augmentation identify a single solution among the infinite compatible with the main task.

The solution is given at the differential level by:

$$\dot{\mathbf{e}}_{\mathbf{ext}} = \mathbf{J}_{\mathbf{ext}}\, \dot{\mathbf{q}}$$

By differentiating with respect to time, and using a weighting matrix $\mathbf{H}$ in the task space, the extended task acceleration is given by:

$$\mathbf{H}\, \ddot{\mathbf{e}}_{\mathbf{ext}} = \mathbf{H}\, \mathbf{J}_{\mathbf{ext}}\, \ddot{\mathbf{q}} + \mathbf{H}\, \dot{\mathbf{J}}_{\mathbf{ext}}\, \dot{\mathbf{q}}$$

Using a generalized inverse, the joint acceleration is given by:

$$\ddot{\mathbf{q}} = (\mathbf{H}\, \mathbf{J}_{\mathbf{ext}})^{\#}\, \mathbf{H}\, (\ddot{\mathbf{e}}_{\mathbf{ext}} - \dot{\mathbf{J}}_{\mathbf{ext}}\, \dot{\mathbf{q}}) \quad \text{(IV.51)}$$

Using the same approach as in the acceleration level, the hybrid control could be used also to define control law at the torque level by:

$$\tau = \left(\mathbf{H J_{ext} M^{-1}}\right)^{\#\mathbf{M}}\, \mathbf{H}\, \left(\ddot{\mathbf{e}}_{\mathbf{ext}} - \dot{\mathbf{J}}_{\mathbf{ext}} \dot{\mathbf{q}} + \mathbf{J_{ext} M^{-1}} (\mathbf{C} + \mathbf{G} + \tau_e)\right) \quad \text{(IV.52)}$$

Note that several values of the weighting matrix $\mathbf{H}$ could be considered as those previously presented in section II.2.3.

IV.5 Dynamic Multi Control Points Approach

In section IV.1 the multi control points formalism is presented to control multi arm redundant systems at the kinematic level. Similarly to that, the dynamic multi control points approach is based on the definition of multiple control points, distributed on appropriate parts of the robotic system.

For each control point, we study and control its dynamic interaction with the environment/robot using different proprioceptive and/or exteroceptive sensors data. From the study of these interactions, we define the dynamic task functions which control the robot behavior during the execution of the desired tasks. These task functions are then pushed into a control structure which manages the priority and the compatibility between them.

IV.5.1 Control Point Definition

As previously presented in the kinematic case, the choice of the control points depends on the robot's architecture and the desired tasks to be executed. The most common ones are the Center of Mass, the end-effectors points, and the contact points with the environment. The latter is used to define interaction tasks between the robot and the environment objects as grasping and manipulation tasks.

IV.5.2 Useful Sensors and Features

The same sensors as those presented in section IV.1.3 could be used for the dynamic control. Furthermore, the defined visual features which are discussed in the kinematic part (section IV.2.3) could also be used in this case.

Additional techniques could be also defined to detect the visual features variation. For example, in [CC01], these features are specified with dynamic criteria which is homogeneous to speed in the image, the relation between the variations of velocity field features and the camera velocity is also modeled. The camera motions are controlled so that the current velocity field in the image becomes equal to motion field in the desired configuration. This approach is used for positioning a camera parallel to a plane and following trajectory. In [KMC04], the application of the velocity field control is applied to visual servoing of robot manipulator under fixed camera configuration.

IV.5.3 Generic Task Definition

Regardless the used control law at the acceleration or torque level, and the desired trajectory of the control point; to define a task at the dynamic level the following data should be determined.

IV.5.3.1 Feature vectors s and $\dot{s}$

The value of the used feature vector $\mathbf{s}$ is given by the robot's proprioceptive or exteroceptive sensors (section IV.1.3). Several types of features could be used as those presented in the kinematic visual servoing case (section IV.2.3).

To determine the variation of the feature vector $\dot{\mathbf{s}}$, we can benefit from the several studies which use motion features in sensor-based control instead of their positions. For example, in [MBG96] the signal, which is used in the closed loop control, is composed of the visual feature positions and their first derivatives. Furthermore, in [CC01] the relationship between image motion derivative and camera velocity and acceleration is exploited to achieve a positioning task.

Recently, in [DAAAM08, DAMM09], a high speed visual tracking method is defined to find the 3D pose and velocity. It is based on the sequential acquisition of regions of interest which has a double benefit on visual servoing: increase the visual control sampling frequency by reducing the amount of acquired/transmitted data, and use the observed object pose and velocity in the associated image projection model.

Finally, more generally, the variation of the used feature vector $\dot{\mathbf{s}}$ could be determined using robot's model or other sensor-based techniques:

- Using the relation in (IV.8), the feature vector variation can be model-based estimated from joints velocity and sensor's jacobian as: $\dot{\mathbf{s}} = \mathbf{J_s}\,\dot{\mathbf{q}}$.
- The feature variation can be calculated by applying equation (IV.3): $\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v_s}$, using the velocity tensor $\mathbf{v_s}$ at the sensor's point which is determined by applying several specialized techniques on the robot's exteroceptive data as in [DAAAM08].
- In case of slow motion, a simplification of this method consists to consider the instantaneous variation of the feature as $\dot{\mathbf{s}} = \frac{d\mathbf{s}}{dt} = \frac{\mathbf{s_i}-\mathbf{s_{i-1}}}{dt}$ when dt is sufficiently small.

IV.5.3.2 Desired feature values $\mathbf{s}^*$ and $\dot{\mathbf{s}}^*$

Depending on the mobility of the target object, the desired control point pose and velocity, $\mathbf{s}^*$ and $\dot{\mathbf{s}}^*$ respectively, could be constant for fixed object or variable for mobile objects. In the latter case, the same methods as above are used to determine the desired values.

IV.5.3.3 Sensor's jacobian $\mathbf{J_s}$

The sensor jacobian, calculated as $\mathbf{J_s} = \mathbf{L_s}\,{}^{\mathbf{s}}\mathbf{W_{cp}}\,{}^{\mathbf{cp}}\mathbf{J_q}$ from the articular Jacobian ${}^{\mathbf{cp}}\mathbf{J_q}$ at the control point given by the robot's model, the interaction matrix $\mathbf{L_s}$ which depends on the chosen feature, and the twist transformation matrix ${}^{\mathbf{s}}\mathbf{W_{cp}}$ given by (IV.5).

IV.5.3.4 Variation of sensor's jacobian $\dot{\mathbf{J}}_\mathbf{s}$

Considering constant twist transformation ${}^{\mathbf{s}}\mathbf{W_{cp}}$ and interaction $\mathbf{L_s}$ matrices, the sensor jacobian variation $\dot{\mathbf{J}}_\mathbf{s}$ is directly related to the variation of the articular jacobian $\dot{\mathbf{J}}_\mathbf{q}$ (Hessian) which can be derived either from numerical differentiation or from analytical formulation [Hou05].

The Hessian matrices are generally computed by hand calculations, finite-differentiation methods, or by automatic differentiation technique as in [RASR11, FA12]. Several tools, as the symbolic software SYMORO+, are used to calculate the value of the product of the Hessian by

the joint velocity ($\dot{\mathbf{J}}_\mathbf{q}\dot{\mathbf{q}}$) in a symbolic form [KC+97].

In the past several decades, a number of applications required the use of the Hessian, although these are not as common. In addition to its use in the acceleration control (IV.31), it is used in the Quadratic Rate Control which is useful in the vicinity of manipulator singularities [PL91], in case of limited computational resources, and furthermore to classify singularities [Bed90, SOC95].

The Hessian is also widely used in the graph theory where there exist a variety of Hessian matrices where each one corresponds to a specific network function [KK68, WW78]. For parallel manipulators, [DBDS97] obtained a closed form analytic expression for the derivative of the inverse Jacobian matrix with respect to time and with respect to the active joint variables. Moreover, [MG91] formulated the time derivative of the Jacobian of a fully parallel manipulator for use in acceleration analysis.

IV.5.3.5 Error regulation methods

Several methods can be used for dynamically regulating the task error to the desired values. We present in this paragraph various examples of linear dynamic regulation. The first one is classically used in the majority of the applications.

(a) Linear dynamic behavior

For tasks where the desired motion of the control point is specified, a linear dynamic behavior can be obtaining by considering:

$$\ddot{\mathbf{e}} = -\mathbf{K_d}\,\dot{\mathbf{e}} - \mathbf{K_p}\,\mathbf{e} = -\mathbf{K_d}\;(\dot{\mathbf{s}} - \dot{\mathbf{s}}^*) - \mathbf{K_p}\;(\mathbf{s} - \mathbf{s}^*)$$

where $\mathbf{K_p}$ and $\mathbf{K_d}$ are the constant position and velocity gain matrices.

A proper choice of $\mathbf{K_p}$ and $\mathbf{K_d}$, i.e. $\mathbf{K_p} = k_p\,\mathbf{I_m}$ and $\mathbf{K_d} = k_d\,\mathbf{I_m}$ with $x^2 + k_d x + k_p$ a Hurwitz polynomial, implies that $\mathbf{e}$ converges exponentially to zero.

(b) Another linear dynamic behavior

Another choice of the error regulation is to only consider:

$$\ddot{\mathbf{e}} = -\mathbf{K_d}\,\dot{\mathbf{e}} = -\mathbf{K_d}\;(\dot{\mathbf{s}} - \dot{\mathbf{s}}^*)$$

The resultant control law will thus minimize $\|\ddot{\mathbf{e}} + \mathbf{K_d}\,\dot{\mathbf{e}}\|$. In [HHS07] the Lyapunov function $\mathcal{L} = \frac{1}{2}\,\|\dot{\mathbf{e}}\|^2$ is used to prove the stability of this system.

(c) Constant velocity behavior

An interesting approach for tasks that involve large motion to a goal position, where a particular path is not required, is based on the selection of the decoupled end-effector command vector as:

$$\ddot{\mathbf{e}} = -\mathbf{K_d}\;(\dot{\mathbf{s}} - \nu\,\dot{\mathbf{s}}^*)$$

where $\dot{\mathbf{s}}^* = \frac{k_p}{k_d}(\mathbf{s} - \mathbf{s}^*)$ and $\nu = min(1, \frac{\mathbf{V}_{max}}{\|\dot{\mathbf{s}}^*\|})$.

If $\|\dot{\mathbf{s}}^*\| < \mathbf{V}_{max}$ this control law will be equivalent to a PD controller with the following behavior:

$$\ddot{\mathbf{s}} + \mathbf{k_d}\ \dot{\mathbf{s}} + \mathbf{k_p}(\mathbf{s} - \mathbf{s}^*) = \mathbf{0} \tag{IV.53}$$

If the desired velocity reaches values beyond the maximum allowable velocity, the above control law will yield the following behavior which will maintain a constant velocity in the direction of the goal:

$$\ddot{\mathbf{s}} + \mathbf{k_d}\ \left(\dot{\mathbf{s}} - \mathbf{V_{max}}\frac{\dot{\mathbf{s}}^*}{\|\dot{\mathbf{s}}^*\|}\right) = \mathbf{0}$$

This allows a straight line motion of the control point at a given speed $\mathbf{V}_{max}$. The velocity vector $\dot{\mathbf{s}}$ is, in fact, controlled to be pointed toward the goal position while its magnitude is limited to $\mathbf{V}_{max}$. The control point will then travel at $\mathbf{V}_{max}$ in a straight line, except during acceleration and deceleration segments. This command vector is particularly useful when used in conjunction with the gradient of an artificial potential field for collision avoidance [Kha87].

IV.5.4 Examples of Generic Tasks

As presented in the previous section, controlling the system at the torque level, by directly including the dynamic model of the system, gives dynamically consistent result and provides better performance than controlling at the acceleration level. Thus, we suggest defining the desired tasks using the following control law:

$$\tau = \left(\mathbf{J_s}\ \mathbf{M^{-1}}\right)^{\#\mathbf{M}}\left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_\mathbf{s}\ \dot{\mathbf{q}} + \mathbf{J_s}\ \mathbf{M^{-1}}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e})\right) \tag{IV.54}$$

Depending on the desired task to be executed, several modifications on this control law can be adopted. On the next paragraphs, several examples of generic tasks are presented.

IV.5.4.1 Positioning task

For the positioning task, the desired configuration $\mathbf{s}^*$ is reached at a desired speed $\dot{\mathbf{s}}^*$. Furthermore, in this case, there is no contact between the control point and the environment, thus $\tau_\mathbf{e} = \mathbf{0}$. This generic task could be applied, for example, for the COM positioning or for free end-effectors motion tasks. The corresponding control law will be:

$$\tau_{\mathbf{pos}} = \left(\mathbf{J_s}\ \mathbf{M^{-1}}\right)^{\#\mathbf{M}}\left(-\mathbf{K_d}\ (\dot{\mathbf{s}} - \dot{\mathbf{s}}^*) - \mathbf{K_p}\ (\mathbf{s} - \mathbf{s}^*) - \dot{\mathbf{J}}_\mathbf{s}\ \dot{\mathbf{q}} + \mathbf{J_s}\ \mathbf{M^{-1}}(\mathbf{C} + \mathbf{G})\right) \tag{IV.55}$$

IV.5.4.2 Contact task

This generic task can be used, for example, to apply precise forces on the system environment, for object lifting tasks or simple end-effector support to ground. For the tasks with rigid contact on the controlled point, there is no variation of feature vectors thus $\ddot{\mathbf{e}} = \mathbf{0}$. However, in

this case, the desired contact forces are modeled in $\tau_{\mathbf{e}}$ vector. Thus, the corresponding control law will be:

$$\tau_{\mathbf{contact}} = \left(\mathbf{J_s}\ \mathbf{M^{-1}}\right)^{\#\mathbf{M}} \left(-\dot{\mathbf{J}}_\mathbf{s}\ \dot{\mathbf{q}} + \mathbf{J_s}\ \mathbf{M^{-1}}(\mathbf{C} + \mathbf{G} + \tau_\mathbf{e})\right) \tag{IV.56}$$

Recall that τ_e is the external torque applied on the robot such as $\tau_e = \mathbf{J_s^\top h_e}$ where $\mathbf{h_e}$ is the vector of contact forces exerted on the manipulator by the environment including the forces applied on the body of the robot.

The presented tasks are generic ones, thus could be used to perform several tasks on the robotic system. For example, in case of humanoid robot control, the positing task could be used to maintain the robot's balance by keeping the COM in a desired plane, or to move the robot's arm to a desired position and it can be used also to retain a desired posture of the robot's hip or chest for example. Furthermore, it can be even used when robot locomotion is involved, as it is previously represented in the kinematic multi control point. In the other hand, the contact task is used in case of hand manipulation with contact with the object, and for foot placement.

IV.6 Conclusion

In this chapter, we presented an approach for controlling a general redundant multi-arm system at the task level, using generalized generic tasks to ensure the method's adaptability to various robotic platforms. This method is first elaborated at the kinematic level, and then extending to the dynamic case.

The multi-control points approach consists on controlling the robotic system by studying and monitoring the system/environment interactions using different types of embedded and external sensors. Therefore, various types of sensors are first presented, then the elements of the general control law are discussed, especially in case of using vision measurements in the control law, and finally several generic robotic tasks are defined, at the kinematic level, for positioning, cooperating and ensuring visibility.

In the dynamic approach, the generic tasks are defined using control laws for redundancy resolution at acceleration and torque levels, with a comparison between two different methodologies for the dynamic task sequencing.

In the next chapter, the defined approach will be applied into two platforms: HRP2 and NAO humanoid robots. Thus, the presented generic tasks and their corresponding control laws will be applied into localization, navigation and manipulation tasks using several embedded and external sensors.

V

Application of Service and Assistive Tasks

Contents

After the development of the general framework of multi-control points approach at the kinematic and dynamic levels in the previous chapter, next sections will use this approach to perform multiple service tasks into several robotic platforms using the appropriate redundancy resolution technique between those presented and compared in the previous chapters.

Therefore, a presentation of the used platforms is given in the first section where the mechanical and software architecture of the HRP-2 and Nao humanoid robots are approached in addition to the used embedded and external sensors.

In the second section, we present the general structure of the applied scenarios in the experimental part. Three applications on HRP-2 and Nao robots are presented and decomposed into several simple tasks.

One special and important task when using visual servoing is the tracking one; it is an essential task when performing robot's localization or object's manipulation; therefore, different tracking methods and tools are discussed in the third section including the presentation of the 3D model-based and cloud tracking techniques.

After that, these methods and techniques are used to apply several applications into the two platforms. Firstly, two different approaches are developed for Nao robot's self localization and navigation tasks. Secondly, manipulation task is applied by simulation, on the HRP-2 humanoid robot using OpenHRP simulator, and in real-time on the Nao humanoid robot. Furthermore, a discussion and proposition of new error regulation strategies is presented in this section to improve the manipulation capability of the robot.

V.1 Experimental Platforms

We present in this section the different robotic platforms which are used to perform the various service and assistive tasks. Therefore, a description of the architecture of the multi-arm platform, HRP-2 and Nao humanoid robots is presented, in addition to the main embedded and external sensors which are used to perform the desired scenarios. Furthermore, the software environments which are exploited during simulations and experiments are also described in this section.

V.1.1 Robotic Systems

The desired scenarios are applied into three robotic systems: A multi-arm system, HRP2 and Nao humanoids. This part presents the mechanical architecture, the number of DOF and the embedded sensors of these systems.

V.1.1.1 Multi-arm platform ARMS

The first robotic platform consists of a multi-arm system which is designed to be used in the ARMS project.

Project's objectives:
ARMS is an ANR ARPEGE project accepted in 2010 for 4 years [ARM] which is funded by the french National Agency of Research. This project is a collaboration between three research laboratories (LASMEA, LAMI and IRCCyN) and two industrial partners (ADIV and Clemssey). The main goal is to study the robotization of bovine muscle separation in meat cutting and transformation processes by using a multi-arm system.

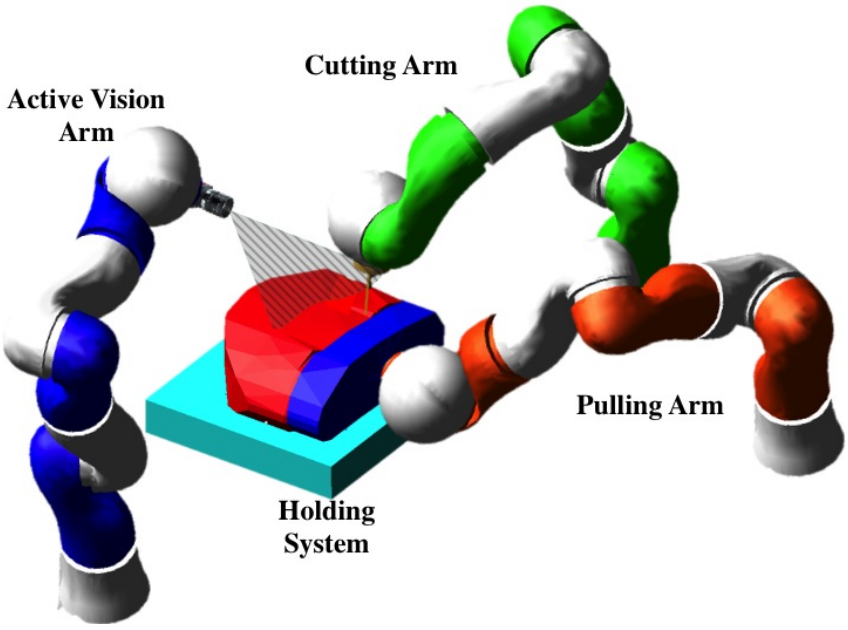


Figure V.1 – Schema of ARMS platform

Platform's architecture:
Four main mechanical actions (handling, pulling, pushing and cutting) are envisaged to be applied on this platform. Thus we consider, as shown in figure V.1, a holding system (table) for maintaining the meat piece (composed of two muscles) in a fixed position and three fixed Kuka lightweight robots (LWR4+). Two robots are used to apply meat handling and cutting tasks and the third is used to integrate external sensors that extract relevant information in real time from the muscle and platform's environment.

The MD-H parameters of 7 DOF kuka LWR robot and the general geometric scheme were given in Table III.8 and Figure III.23 respectively. Furthermore, in the designed platform, a cutting tool (knife) and a vision system are mounted on the end-effector of the cutting arm and active vision arm respectively.

V.1.1.2 HRP-2 robot architecture

HRP-2 is a robotic platform for the Humanoid Robotics Project headed by the Manufacturing Science and Technology Center (MSTC) in 2002. The robotic system was designed and integrated by Kawada Industries, Inc. together with the Humanoid Research Group of National Institute of Advanced Industrial Science and Technology (AIST) [HRP].

Dimensions	
Height	1540 mm
Width	620 mm
Depth	355 mm
Weight	58 Kg (inc. batteries)
Degrees of Freedom	
Head	2 DOF
Arm	6 DOF × 2
Hand	1 DOF × 2
Waist	2 DOF
Leg	6 DOF × 2
Total	30 DOF
Sensors	
Torso	3 axes vibration gyro
	3 axes velocity sensor
Arms	6 axes force sensors
Legs	6 axes force sensors
Walking speed	
up to 2 km/h	

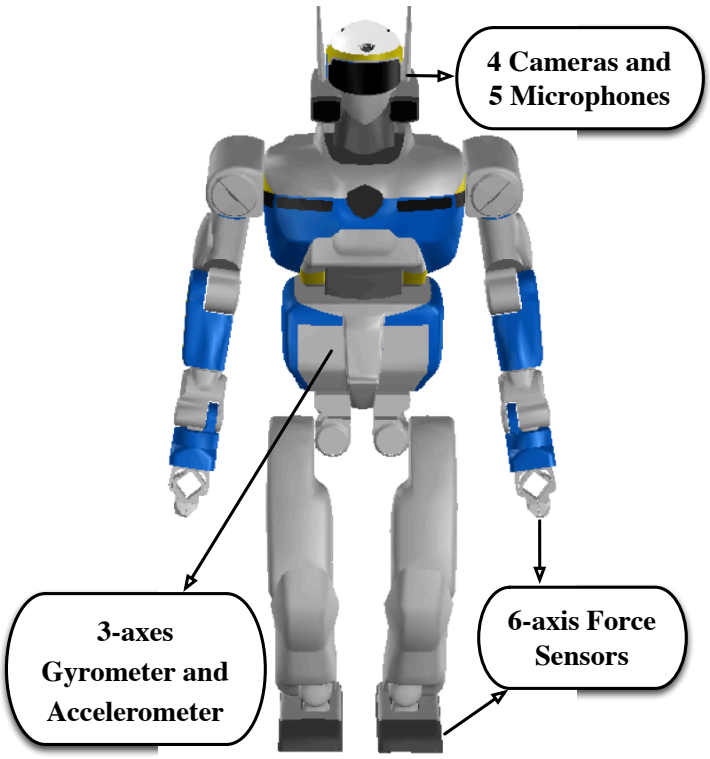


Table V.1 – HRP-2 robot specifications and useful sensors presentation

HRP-2 has 30 DOF including 2 DOF for its hip, 6 DOF for each arm and leg, 2 DOF for the head, and 1 DOF for each gripper (can hold 2 kgf/hand). The cantilevered crotch joint allows for walking in a confined area with a speed up to 2 km/h. Its highly compact electrical system packaging allows it to forgo the commonly used "backpack" used on other humanoid robots (more specifications given in Table V.1).

This humanoid has 4 cameras (a pair of narrow-angle cameras and a pair of wide-angle cameras) allowing it to obtain visual information from near and far scenes. The chest of the robot hides two important sensors in mobile robotics: a gyroscope informing the robot orientation in 3D space and an accelerometer giving the robot accelerations. HRP-2 is also equipped with 4 force sensors, one in each foot and each hand, these sensors are used for the measurement of the efforts in the ankles and wrists joints. It has also a series of 5 microphones for obtaining audio data (Table V.1).

V.1.1.3 Nao robot architecture

Nao H25 Robot [GHB+09], developed by Aldebaran robotics, is a biped robot with 25 DOF (5 in each leg, 1 in the pelvis, 2 in the head, 5 in each arm and 2 actuated hands). It has 3-fingered robotic hands used for grasping and holding small objects (300 g can be carried up using both hands). Nao's motion is based on DC Motors (direct and stepper types) and the robot has a limited autonomy of 1 hour approximately (more specifications in Table V.2).

Nao Robot has 2 ultrasound devices, situated in his chest, that provide space information in 1 meter range distance if the object is situated at 30 degrees from the robot's chest. 2 bumpers are situated in front of each foot, these contact sensors help to determine if the robot is touching something. Nao has 8 Force Sensing Resistors (FSR) situated at sole of feet (4 FSR in each foot) which are useful especially when generating movements sequences, to know

Dimensions	
Height	573.2 mm
Width	273.3 mm
Depth	290 mm
Weight	5.2 Kg
Degrees of Freedom	
Head	2 DOF
Arm	5 DOF × 2
Hand	1 DOF × 2
Waist	1 DOF
Leg	5 DOF × 2
Total	25 DOF
Walking speed	
up to 0.5 km/h	

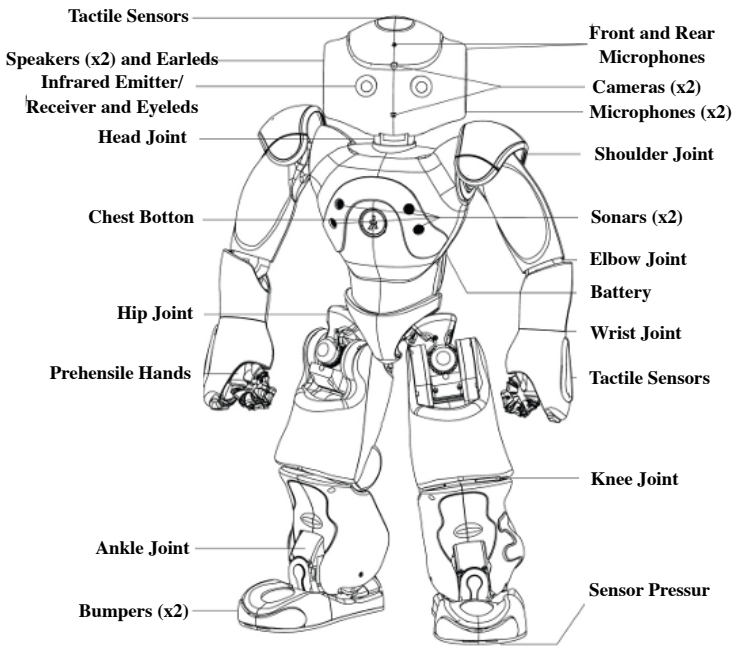


Table V.2 – Nao robot specifications and useful sensors presentation

if one position is a zero moment pose (ZMP). Nao has 2 inertial sensors: a gyrometer and an accelerometer, which are two important devices when studying motion concisely kinematics and dynamics. These sensors help us to know if the robot is in a stable position or in unstable one when the robot is walking (Table V.2).

Nao's walking uses a simple dynamic model (linear inverse pendulum) and quadratic programming. It is stabilized using feedback from joint sensors. This makes walking robust and resistant to small disturbances, and torso oscillations in the frontal and lateral planes are absorbed. Nao can walk on a variety of floor surfaces, such as carpeted, tiled, and wooden floors. The best walk was demonstrated in the 2009 RoboCup edition with the university of Bremen [GHRL09]. They created a very stable walk with a speed of 12 cm/s (equivalent to 0,43 km/h).

V.1.2 Used Sensors

Several embedded and external sensors are used in the experiment part when applying the desired service and assistive tasks into the various platforms. Therefore, we present in this part the main sensorial elements which are used during tasks execution.

V.1.2.1 External Marlin camera

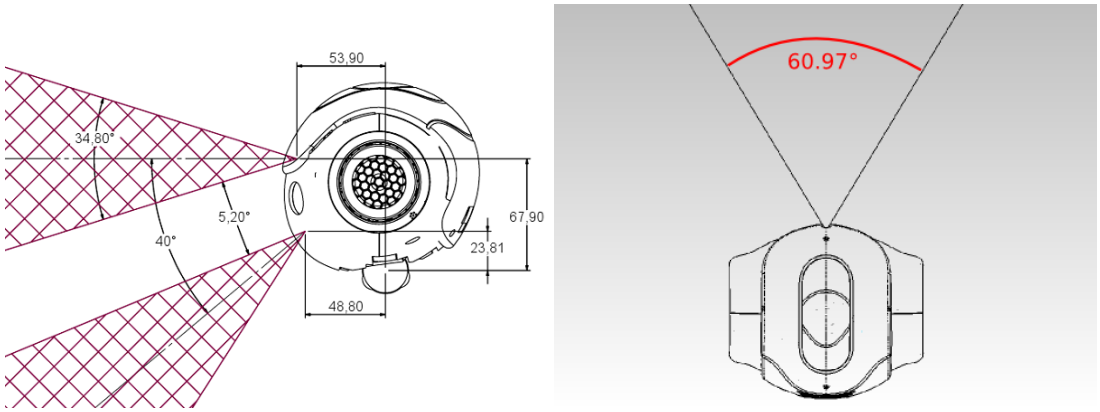
In the simulation part on HRP-2, the robot's cameras are replaced by an external one which is connected to the simulator's laptop using a IEEE 1394a fireWire. The used camera is the black and white AVT MARLIN (F-131B) with highly sensitive SXGA 1280 (H) × 1024 (V) global shutter CMOS. At full image resolution, it offers up to 25 fps and an efficient data transmission and image processing in terms of time and data sizes, which makes it suitable for our robotic applications.

Sensor	2/3" progressive scan CMOS
Resolution	1280 x 1024
Pixel Size	6.7 μm x 6.7 μm
Dimensions	58mm x 44mm x 29mm
Weight	120 g
Gain	0-16 dB

Table V.3 – Specifications of AVT Marlin camera (F-131B)

V.1.2.2 Nao embedded cameras

The Nao robot's head (v3.2) contains two identical CMOS cameras having VGA, i.e. 640 × 480 resolution, which can capture up to 30 images per second (in YUV422 color space) with a fixed focus on a range of 30 cm to infinity. Cameras configuration do not allow stereo vision as their view-fields do not overlap (Fig. V.2). The first camera is heading forward and the other downward in order to see the floor in front of Nao. Moreover, the robot's software implementation did not allow to use them simultaneously, but Nao can move his head 239° horizontally and 68° vertically.



a - Vertical field of view b - Horizontal field of view

Figure V.2 – Field of view of Nao robot's head (v3.2)

V.1.2.3 External 3D vision system (Kinect)

Since humanoid robot is targeted to human-friendly indoor environments, a smart home endowed with networked sensors would provide additional useful information to monitor both humans and robots. In smart environments, components are working together by exchanging information via the local network [SHS08].

Various sensors could be used to control and monitor the entire living space: several are embedded in the robotic system (force sensors, inertial center, (omni) cameras, odometers ...) and others are attached to the robot's environment (laser sensors, external cameras ...).

In the presented applications, in addition to the embedded sensors (especially cameras), an external 3D vision system is used to perform the desired tasks. In addition, one of the relatively new and powerful low-cost sensors is the Kinect which provides color and range images, suitable for human motion detection and tracking. It was used in many applications: Dingli

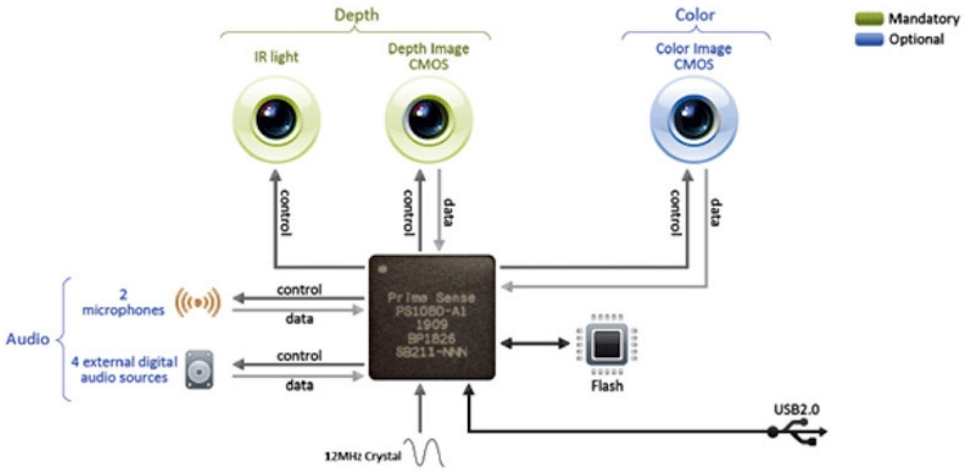


Figure V.3 – Schema of Kinect components

et al. [DAM12] have created an Ambient-Assisted Living application which monitors a person's position, labels objects around a room and raises alerts in case of falls. Stone and Skubic [SS11] have investigated this sensor for in-home fall risk assessment. Ni et al. [NWM11] have used color-depth fusion schemes for feature representation in human action recognition.

This vision system, i.e.the Kinect camera, consists of two optical sensors whose interaction allows a three-dimensional scene analysis. One of the sensors is an RGB camera which has a video resolution of 30 fps. The image resolution given by this camera is 640x480 pixels. The second sensor has the aim of obtaining depth information corresponding to the objects found at the scene.

The working principle of this sensor is based on the emission of an infrared signal which is reflected by the objects and captured by a monochrome CMOS sensor. A matrix is then obtained which provides a depth image of the objects in the scene. An investigation of the geometric quality of depth data obtained by the Kinect sensor was done by [KE12], revealing that the point cloud does not contain large systematic errors when compared with a laser scanning data.

V.1.3 Software Platforms

In this part, we present the different external software platforms that will be used to apply the desired tasks either by simulation for ARMS platform and HRP-2 robot or in real-time for Nao robot. In addition to that, the embedded software architectures are also described for these robots.

V.1.3.1 Simulation of multi-arm platform

Before applying the desired scenario on the real multi-arm platform which is mounted for the ARMS project, a co-simulation between Matlab/Simulink and Adams was performed to validate the effectiveness of the designed control laws.

In fact, the software ADAMS, was used to generate automatically the dynamic model of the platform. This model was then fed under MATLAB/Simulink environment to be analyzed. The model is used in the control scheme to predict the behavior of the system using the designed controller.

ADAMS and MATLAB/Simulink were successfully interfaced to yield a powerful tool to model and analyze the behavior of this multi-arm system. This tool allowed us to easily modify the control scheme and investigate its effect on the dynamic behavior of the system.

V.1.3.2 HRP-2 softwares and OpenHRP simulator

Regarding the IT architecture, HRP-2 contains on his chest two computers running the Linux distribution Fedora. The first one which manages the robot's joints is called HRPC, while the second one is related mainly to vision processing is called HRPV. These two computers communicate with each other using a protocol based on the CORBA3 architecture and are connected by ethernet. HRP-2 holds also a WiFi card which permit the connection to the robot

computers from another remote station.

Furthermore, CORBA service allows remote procedure calls; it encapsulates abstract function call and class definitions using a language called IDL (Interface Definition Language) which can then be used to generate C++ code. These calls can be used to transfer data through a network and transparently execute remote code. Thus, a CORBA server provides the possibility to interact with the controller by creating, reading and sending signals as well as sending script commands.

Due to the limits in the availability of the robot in LAAS laboratory in France, the scenario on this robot is applied using the Open Architecture Human-centered Robotics Platform (OpenHRP) [Ope] which is an integrated software platform for robot simulations and software developments. It can simulate the dynamics of structure varying kinematic chains between open chains and closed ones like humanoid robots. It can detect rapidly and precisely the collision between robots and their working environment including other robots.

It can also simulate the fields of vision of the robots, force/torque sensors and gradient sensors according to the simulated motions. In addition to that, it provides various software components and calculation libraries that can be used for robotics related software developments.

V.1.3.3 Nao embedded softwares

The Nao robot is fully programmable at high level (ex: cognition) or low level (ex: motion primitives) with Aldebaran Robotics Software Development Kit (NAOqi). The embedded Nao software, includes a fast, secure reliable and cross-platform, distributed robotics framework that gives access to all the features of the robot (sending commands to actuators, reading sensors, managing Wi-Fi connections...), and provides a solid foundation to improve Nao's functionality. NaoQi's functions can be called in C++, in Python and even in Urbi.

Furthermore, Choregraphe running on a PC gives access, thanks to a graphical user interface, to all the functions provided by NAOqi. In addition, this user-friendly software and its flow diagram allow an easy event-based, sequential or parallel programming using a preprogrammed set of behavior boxes which are easily configurable.

V.1.3.4 ROS environment

In addition to the embedded software, we used the ROS (Robot Operating System) environment to manage simultaneously external and embedded robot's modules when using external sensors. ROS is a software framework which provides libraries and tools for creating robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more [QCG+09]. With its modular design, ROS makes it easy to develop network robot systems.

One of the external modules which is used in the experimental part is the Point Cloud Library (PCL) which is a standalone, large scale, open project for 3D point cloud processing [RC11]. The PCL framework contains numerous state-of-the art algorithms including filtering,

feature estimation, surface reconstruction, registration, model fitting and segmentation, as well as higher level tools for performing mapping and object recognition.

The tracking module of PCL [Ued] provides a comprehensive algorithmic base for the estimation of 3D object poses using Monte Carlo sampling techniques and for calculating the likelihood using combined weighted metrics for hyper-dimensional spaces including Cartesian data, colors, and surface normals. Seamless interfacing with the Kinect sensor, Nao robot, and PCL is provided in ROS.

V.2 Application Presentation

In this section, we present the general framework of the applied scenarios in the experimental part. In the first part, the three applications on HRP-2 and Nao robots are presented, then in the second one the various frames are defined in the robots' body and environment. Finally, presented applications are decomposed into several simple tasks which are defined in the last part.

V.2.1 Scenario Presentation

The first application consists in controlling the HRP-2 robot to perform object manipulation while maintaining the robot's equilibrium. Initially in standing positing, the robot uses its visual system to detect and track an object in the environment and performs a manipulation task using one and two hands. This scenario is applied using the OpenHRP software platform which allows a perfect simulation of the dynamic/kinematic structure and behavior of the robot; furthermore embedded cameras are replaced by an external camera which is connected to the simulator.

In the second application, only embedded sensors in Nao robot will be used with the multi control points approach to perform the desired service tasks. Thus, 3D visual feedback data and a model-based tracking techniques are used to execute, in real-time and closed loop, many tasks in a semi structured environment. The robot's cameras are calibrated, and a rough geometric model of the objects is available (doors, tables, items to grasp ...).

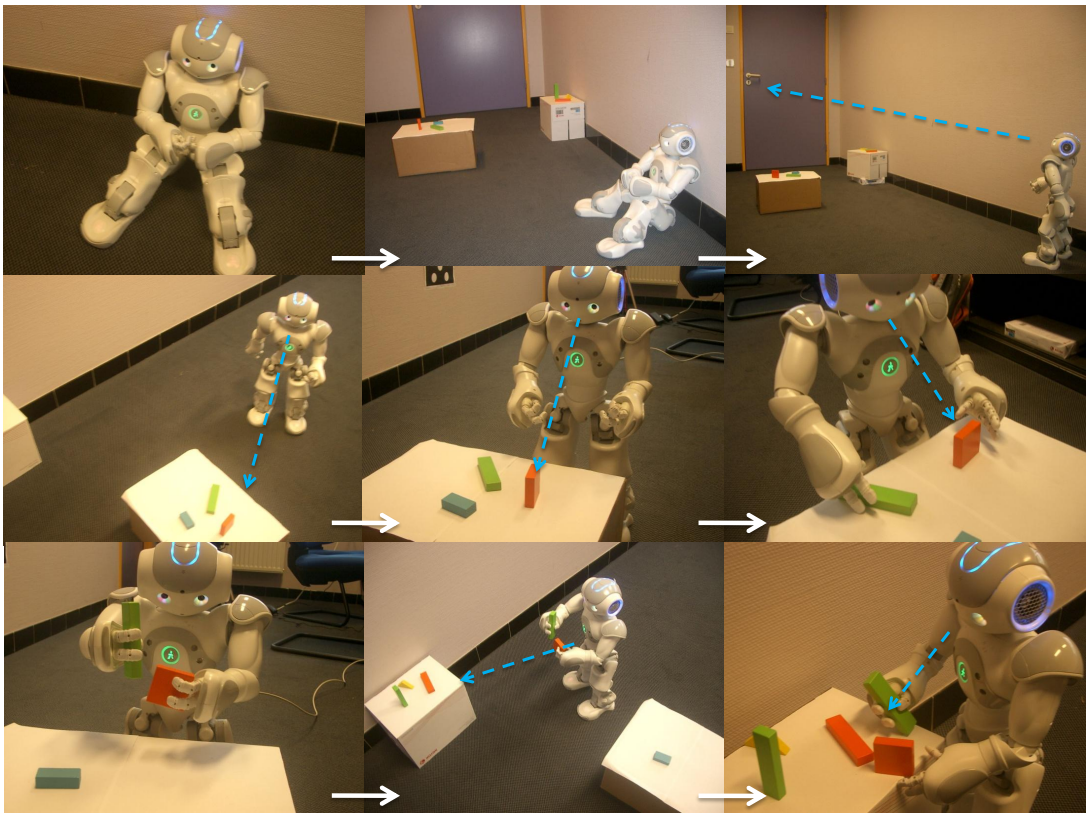


Figure V.4 – Nao picks up the orange and green pieces and deposits them on the table near the door

The envisioned scenario consists of a Nao robot which can carry out service tasks, moving around the room and manipulating objects. One of the missions that can be requested from such robot would be to "*Pick up the orange and green pieces and deposit them on the table nearby the door*" (see Fig. V.4).

In the third application, external feedback is added to the system which is composed of a humanoid robot, navigating in an indoor smart environment, where a 3D vision system, consisting of one or more low cost Kinect cameras, monitors and tracks both the user and robot's activities, as depicted in Fig. V.5.

Therefore, in this case, the robot assists the user to pick up an object in an indoor environment. In fact, the Kinect sensor is used to monitor and track the 6D pose of the Nao robot to perform a localization and a precise navigation to a detected object to pick it up from the floor.

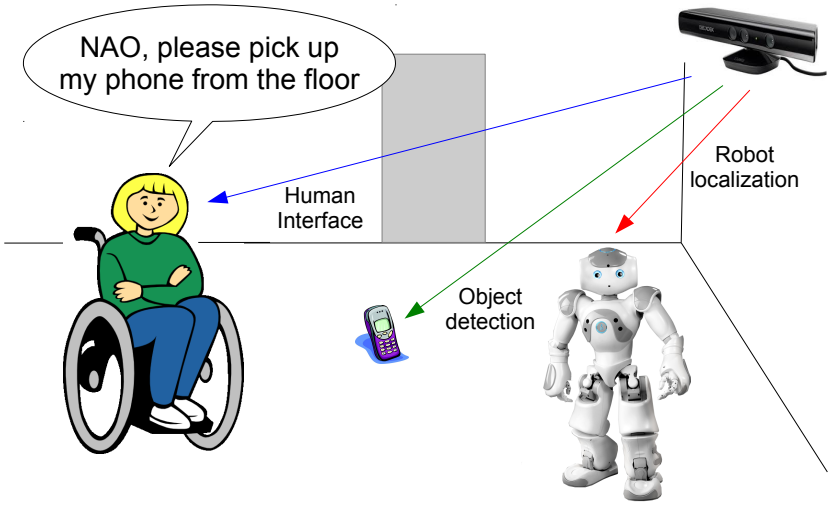


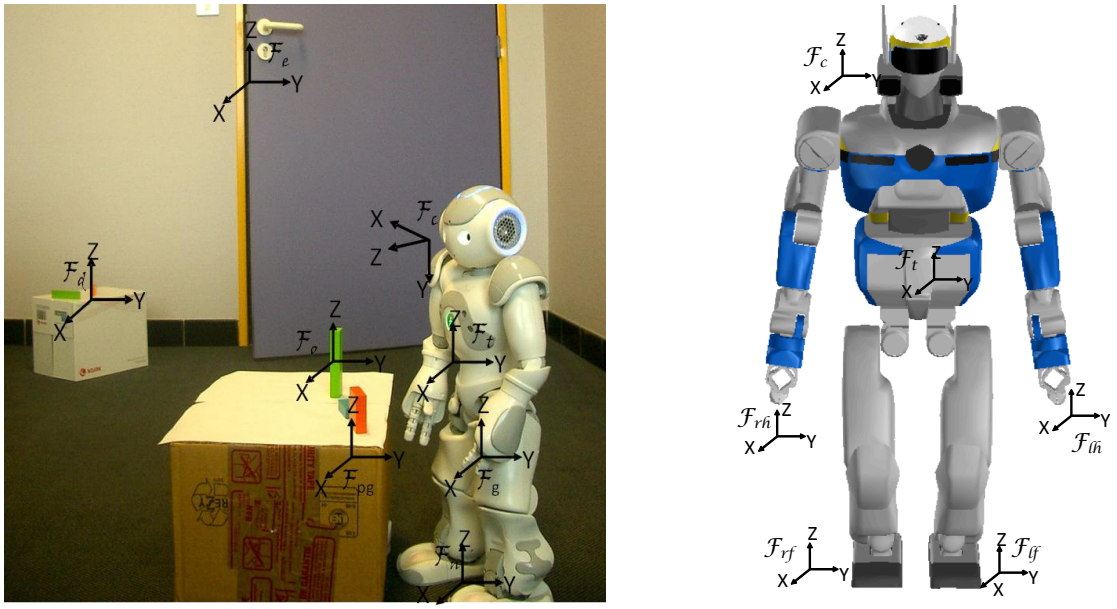
Figure V.5 – Overview of the system: the Kinect sensor in the smart environment can monitor the user activity, detect objects on the floor, and localize precisely the robot.

V.2.2 Frame Definition

To execute the desired tasks, the multi-control points approach is applied on the humanoid robots HRP-2 and Nao; thus, we should first define several coordinate frames on the robot's body and environment's items as represented in Fig. V.6.

For the manipulation tasks on HRP-2 robot, the entire structure is controlled to maintain the robot's equilibrium in a vertical position, and to apply the rigid object grasping tasks using the right and left grippers. Therefore, we consider 6 control points on the robot (see Fig. V.6b): $\mathcal{F}_{lf}$ the left foot frame, $\mathcal{F}_{rf}$ the right foot frame, $\mathcal{F}_{lh}$ the wrist frame in the left hand , $\mathcal{F}_{rh}$ the wrist frame in the right hand, $\mathcal{F}_c$ the camera frame in the robot's head and finally $\mathcal{F}_t$ the CoM frame in the torso of the robot.

In Nao's body, we consider the following frames: Nao's space frame $\mathcal{F}_n$ (between the robot's feet), $\mathcal{F}_t$ on the robot's torso, $\mathcal{F}_c$ a camera attached frame, robot's hand frame $\mathcal{F}_h$, robot's gripper frame $\mathcal{F}_g$ and a pre-grasping frame $\mathcal{F}_{pg}$. Second, the following frames are defined in



a - Useful frames in Nao's environment

b - Useful frames in HRP-2 robot

Figure V.6 – Frames definition in robots body and environment

the robot's environment: object's frame $\mathcal{F}_o$, desired object's pose frame $\mathcal{F}_d$ and environment's frame $\mathcal{F}_e$ for the localization task.

V.2.3 Task Definition

Regardless the robotic platform, to execute such missions, many problems should be addressed and resolved first:

— How can the robot localize itself with respect to its environment?
— How to decompose the robot's mission into elementary tasks?
— What information should be given to the robot to execute each task?
— What are the more useful techniques the robot can exploit to perform these tasks?

In the rest of this part, the envisaged scenarios are decomposed into simpler generic tasks to reduce the complexity of the problem to be solved. Thus, the desired scenarios will be executed by answering the following questions:

(a) **Where am I ? → (Self) Localization task**
Depending on the desired mission to be executed, the robot either searches for the corresponding set of models in its environment to be self-localized with respect to them (such as table, door, light switch, corner, ...), or it is localized by an external fixed or mobile sensor mounted in the environment (such as a camera in the room's ceiling, a camera embedded in another robot, ...).

In the second application, the robot detects the items in the field of view of its camera, chooses the appropriate one and calculates its position/orientation with respect to the item

using one of the tracking techniques. On the other hand, in the third scenario, the external sensor localize the robot by directly detecting and tracking its model. It calculates its 3D pose using a part of the robot's model or some markers in the robot's body for example.

(b) **What must I do and How to do it ? → Task scheduler**
Each scenario is interpreted to the robot's language and decomposed into elementary tasks by the task scheduler [GFMGS08].

For example, in case of the second application of service tasks on Nao robot (Fig. V.4), the following tasks are considered:
— Localize the desired objects to manipulate
— Move in the appropriated direction
— Detect and track the desired objects
— Keep these pieces in the robot's field of view
— Move the robot's arms and grasp these pieces
— Go to the desired table and deposit objects

Furthermore, this task scheduler manages the dynamic priority between tasks and checks the performance of their execution. An example of this scheduler is presented in section II.4.3 where a global architecture (Fig. II.13) is proposed to sequence a "Stack of Tasks" in order to reach the desired goal while taking into account several environment constraints.

(c) **Which data will I use ? → Definition of used data**
To execute the defined elementary tasks, the scheduler chooses the appropriate models for each one: models of the objects to manipulate, model of the environment's items (for the self-localization task), model of the robot's body/part (for the localization task), parameters of the robot's embedded cameras and external sensors...

(d) **Where to move ? → Robot locomotion and navigation tasks**
This task is used to move to the region where the task should be executed. Knowing the position of at least one item of the environment, the robot walks in the appropriate direction until entering the range of a defined distance from the object.

During this task, the robot motion is controlled in real-time and closed loop to avoid some possible modifications or unexpected events. Thus, an obstacle avoidance algorithm should be used to adapt the robot's trajectory [SVVY09].

Conversely to the desired stable and balanced locomotion in case of mobile robot, another constraint appears in case of applying manipulation on a fixed humanoid robot: the equilibrium task. It consists in controlling the robot's center of mass to maintain its balance and thus preventing the robot from falling down while performing the manipulation task.

(e) **How to perceive? → Detection and tracking tasks**
Using one of the various techniques of visual tracking that will be presented in the next

section, this task allows us to track in real-time the pose of the desired item such as the robot or the environment's objects to manipulate.

Furthermore, to increase its robustness, a module for automatic re-initialization of the tracker could be implemented, it uses the previous poses of the tracked object to estimate the actual pose. This module can be used in case of a failure due to an occlusion or the fast motions of the robot's camera (especially when walking). Moreover, the automatic transition between the tracking of different objects can also be implemented using the knowledge of the rough relative position between environment's objects.

(f) **How to keep the concerned points in the robot's field of view ? → Visibility task**
This task consists of controlling the position/orientation of the robot's head to focus a (fixed/mobile) point of the environment (item's center, gripper, virtual point...) in the center of the camera's image (presented in section IV.3.3).

This task can be used, for example, in hand-eye coordination for dynamic grasping of objects by focusing on the gripper-item midpoint, to keep the robot's hand and object in the robot's field of view. The used DOF in this task depends on the geometry of the robot's head and the desired complexity of the task. The head's Yaw/Pitch can be simply controlled to focus on the object's center, or a more complex task can also control the distance between the robot's head and the object.

(g) **How to perform the manipulation ? → Grasping task**
This task uses the hand's control point and allows the robot to move it to a desired static or mobile pose. This task can be used to perform pre-grasping, grasping, and displacing objects tasks.

In case of pre-grasping task, the goal position is determined using one of the grasping strategies [DC02]. Usually, they depend on the geometry of the object to manipulate and on the shape of the robot's gripper. Thus, the grasping strategy controls the relative position and/or the angle between the gripper and the item to grasp.

Likewise, for a grasping task the same technique is considered: the robot's arm moves to a desired pose by minimizing the relative distance/angle between the object and the robot's hand in the gripper's Frame.

After presenting robots architecture, defining the useful frames in the robot's body and environment, and decomposing the desired scenario in several generic tasks, we will apply these tasks in next sections. Firstly, we give an overview on the different methods for visual tracking, with the existing tools and we present the technique that will be used in the experimental part. Then, the localization and navigation tasks will be discussed and performed on the Nao robot using two different techniques. And finally, the manipulation and grasping tasks are presented and performed on the HRP-2 (by simulation) and Nao robot (in real-time).

V.3 Visual Tracking Techniques

For robust sensor-based control, in the multi control points approach, a suitable technique should be used to find the position of the robot parts or environment's objects using exteroceptive sensors instead of using the robot's geometric model via joints encoders. One of the most useful techniques to achieve that is the visual tracking, which is an important channel to obtain the external information, and is the prerequisite for the robot to complete tasks.

This technique refers to detection, extraction, identification and tracking of moving targets from the image sequence to obtain motion parameters, such as position, velocity, acceleration, and trajectory etc. In more detail, the process of robot visual tracking is listed as follows: Firstly, the model of the target is determined then a feature is selected to distinguish the target from the scene, but the level of the feature selection depends on the priori knowledge and familiarity to the target. Secondly, the feature is taken as the target state, then the tracking algorithm is determined according to the historical data of the target state. Thus, the process of tracking continually detects the selected features from the input image, to establish and confirm the characteristics between frames.

In the first part of this section, we discuss and classify the different methods for target tracking. Later on, different visual servoing tools are explored and two techniques are presented: the 3D model-based tracker and the Cloud tracking method. These trackers will be used later in the experimental part to track different objects of the robot's environment and the robot's body respectively.

V.3.1 Target Tracking Methods

Several target tracking methods are developed, they can be classified into 3 categories: the first one is based on the matching process, the second one uses motion information, and the last one is based on the estimation.

V.3.1.1 Visual tracking based on matching

The matching process can be applied on the visual features or on the target model. The first case consists on tracking the feature points (segments, straight lines, curves...) which are invariable with respect to the motion, and are not influenced by the size, location, direction and light. The advantage of this algorithm is that the whole target is seen as a relevant object, thus even if occlusion occurs, some features remain visible which can provide the continuity of the tracking.

The core of the second method, the model-based target tracking, is to make use of the priori knowledge of the 2D or 3D model of the target, to match the target model in the detected image. In fact, the known 3D structure model of the target is used to predict the state of the next frame through the track histories. Then, the predicted model will be projected onto the image to be matched with the actual image data; a special estimation function is used to measure the similarity between projection model and image data.

Because of the support of the priori knowledge, these methods have better robustness and

results in case of interference; it can make accurate analysis on three-dimensional motion trajectory, and can also track the gesture of a moving target. Nevertheless, how to propose a 3D model covering all the features, and how to select and to extract these features remains a big challenge.

Another method is the visual tracking based on the regional. The basic idea of this algorithm is to get the target template through the image segmentation. The template can be a rectangular or irregular shape which is slightly bigger than the moving target. A matching measure is used to completely search target image in the next frame image before using the relevant algorithms to track moving targets. The advantage of this algorithm is that the tracking precision is very high, but a large amount of calculation is required when matching in grayscale image or having lager search area.

A last method is the contour matching tracking algorithm, it uses closed boundary contours which are adapted to the real target in the image gradually. In fact, the boundary contours are dynamically updated, in subsequent frames, while tracking the target.

V.3.1.2 Visual tracking based on the information of motion

This technique studies the motion of the target in the image sequence using algorithms as the differential method or the optical flow method.

There exists two types of differential methods: inter-frame difference method and background difference method. The latter is suitable for the case of static background. Inter-frame difference method detects the change by using strong correlation between adjoining frames of the video sequence.

On the other hand, optical flow method [NYC+99] studies the instantaneous two-dimensional velocity distributions of the pixel movement on the observation surface. The two-dimensional velocity vector is the projection of the visible three-dimensional velocity vector in the imaging plane.

V.3.1.3 Visual tracking based on the estimation

Since the goal is to find out the exact location of the moving target in the next frame, the most rapid way is to locally search the probable position of the target in the next frame. There are several algorithms which can achieve that, including Kalman filter and particle filter algorithms.

Kalman filter is an efficient recursive filter: as long as the estimated value of the previous state and the observed value of the current state are known, the estimated value of the current state can be calculated without the need to record historical information of observed or estimated value. It uses linear mean square estimation for state sequence of a dynamic system. There are two stages of the operation of Kalman filter: prediction and update. In the forecast period, filter estimates the current state through using the estimated value of previous state. In the update stage, filter optimizes the observations of the current state by using the observations which obtained in the forecast period, to obtain a more accurate estimated value.

Particle filter is the optimal Bayesian estimation method, also known as sequential Monte Carlo method. The basic idea is to use all known information to construct the posterior probability density of the system state variables, in other words, to use the state transition model to predict a priori probability density of state, then use the recent observations to qualify, and finally get the posterior probability density. According to Monte Carlo theory, when the number of particles is sufficient, this group of particles will be able to completely describe the posterior probability distribution.

With Kalman filter, we need measurement equation when estimating the state, but the particle filter does not require the expression of measurement equation, it only needs to know the conditional probability density, which is easy to get in the images.

V.3.2 Visual Servoing Tools

Many tracking tools have been implemented in several visual servoing toolboxes. One of the first tools is the Matlab Robotics Toolbox by Peter Corke [Cor96] which provides many functions that are useful in robotics such as kinematics, dynamics, and trajectory generation. It is useful for simulation as well as analyzing results from experiments with real robots. The companion Machine Vision Toolbox [Cor05] provides many functions that are useful in machine vision and vision-based control.

The Matlab/Simulink Visual Servoing Toolbox [Cerb] aimed to provide a set of functions and blocks for simulation of vision-controlled systems. Although a working version is available, its development stopped due to the limitations of Simulink for object-oriented programming. Most of the functions needed to be coded in Matlab; thus, its performance is too low for real-time simulations.

Another Matlab extension, the Epipolar geometry toolbox [MP05] consists of a collection of functions for position and design of camera/robot and scene objects, computation of perspective/catadioptric projection and epipolar geometry (perspective and panoramic), and epipolar geometry estimation algorithms (from corresponding points) for pinhole and calibrated panoramic cameras.

While hundreds of software packages for vision exist, and many other for simulated robotics are available, very few of them have been designed for both domains, thus being suitable for visual servoing tasks. One of these visual servoing toolboxes is Javiss (Java-based Visual Servo Simulator) [Cera] which departs from other simulators in that it has been designed from scratch in a modular, distributed way. Not only the tasks are distributed among different modules (agents), but each module can run on a different computer across the network. This enables the development of distributed visual servoing tasks for cooperative robot teams. In addition, such a flexible approach increases the performance of the task with distributed computing power, and it may contribute to solve hardware limitations with the real equipment [Cer06].

Finally, a very important platform is ViSP (Visual Servoing Platform) [MSC05], a modular framework that allows fast development of visual servoing applications. It implements the control of robot motions, the modeling of the visual features, and the tracking of the visual measurements. ViSP features a wide class of control skills as well as a library of real-time tracking processes and a simulation toolkit. The platform is a library implemented in C++

which is developed by the INRIA team Lagadic at IRISA Rennes.

Note that several of the presented tools and libraries are found as independent packages which could be used under ROS environment such as the OpenCV library of programming functions for real time computer vision, and ViSP package which can be useful in robotics, computer vision, augmented reality and computer animation.

V.3.3 3D Model-Based Tracker

On ViSP, the tracking algorithms use the matching technique presented in paragraph V.3.1.1. They are decomposed in several classes: dot tracker, KLT tracker and moving edges tracker. The first one tracks white dots on a black background, or black dots on a white background. A dot is a part of image where the connected pixels have the same level. ViSP has also been interfaced with the OpenCV library to provide a KLT (for Kanade - Lucas -Tomasi) tracker.

The goal is to align a template to an input image. Finally, the moving edges tracker is based on matching through convolution masks. In fact, the edge is sampled, and at each iteration the samples are tracked along the normal to the edge. The equation of the edge is thus computed at each iteration which enables to find the normal to the edge at each sample and detect the outliers.

Furthermore, the latter is used to implement the 3D model-based tracker which consists in computing the pose of a 2D or 3D object in an image sequence. It tracks a 3D model thanks to the moving edges method, represented in Fig. V.7, using a virtual visual servoing technique. Thus, it requires a 3D model and needs to compute the initial pose which is used to project the model on the image. The tracking method assumes that the pose corresponding to the previous image is known, the new lines are tracked, and the goal is to move the pose to match the object in the new image with the projection of the model.

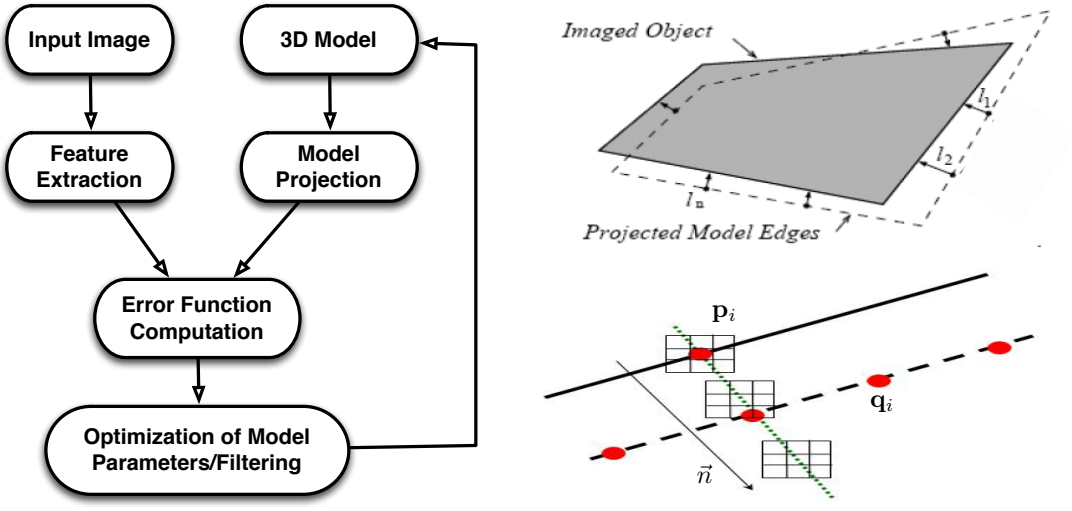
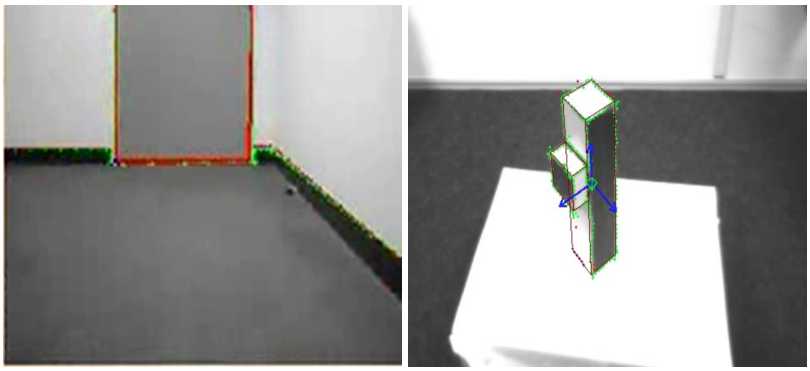


Figure V.7 – Model-based tracking system (left) using the moving edge detection technique (right)

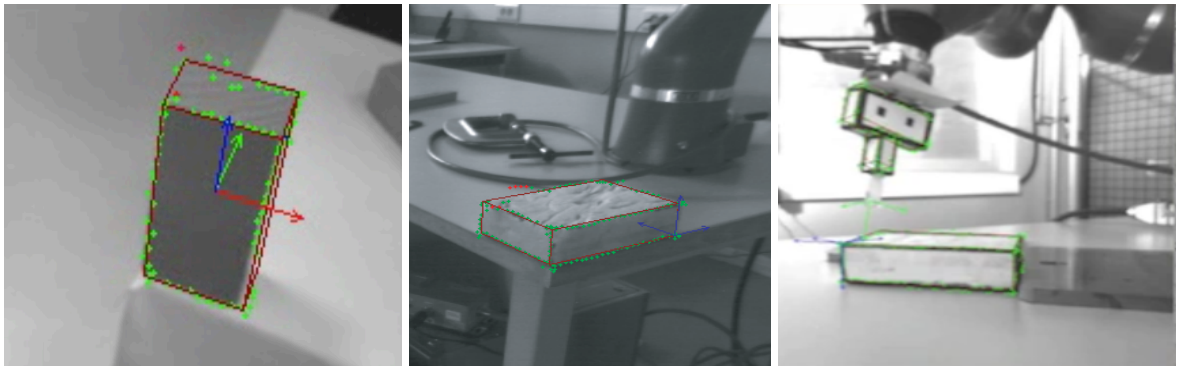
The following error function (err) between image features p_i and model projection q_i is thus minimized along the normal direction $\vec{n}$ (see Fig. V.7):

$$err = \sum_i \Delta(p_i, q_i) = \sum_i |(q_i - p_i) \cdot (n_i)| \tag{V.1}$$

Before presenting, in next sections, the 3D model-based tracker of ViSP when used to perform robot's localization and object's manipulation, we present in Fig. V.8 several examples of objects tracking.



a - Door tracking b - Object tracking



c - Another object tracking d - Box tracking e - Box and EF tracking

Figure V.8 – Tracking examples using 3D model-based technique

V.3.4 PCL Cloud Tracking

The Point Cloud Library (PCL) is an open project for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them.

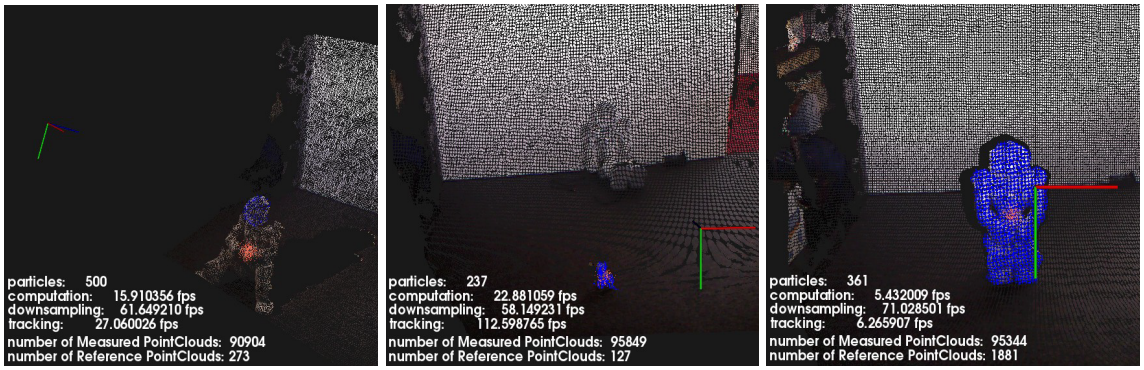
A point cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface. When

color information is present, the point cloud becomes 4D. Point clouds can be acquired from hardware sensors such as stereo cameras, 3D scanners, or time-of-flight cameras, or generated from a computer program synthetically [RC11].

Tracking 3D objects in continuous point cloud data sequences is an important research topic for mobile robots: it allows robots to monitor the environment and make decisions and adapt their motions according to the the world's changes, and furthermore to estimate the three dimensional pose of an object in real-time.

In [Ued], a 3D tracking library for the Point Cloud Library (PCL) project is developed to provide a comprehensive algorithmic base for the estimation of 3D object poses using Monte Carlo sampling techniques and for calculating the likelihood using combined weighted metrics for hyper-dimensional spaces including Cartesian data, colors, and surface normals. This library is optimized to perform computations in real-time, by employing multi CPU cores optimization, adaptive particle filtering (KLD sampling) and other modern techniques.

This library is used in our experiments to track and localize several environments objects, complete robot's body and only a part of the robot (head), using their rigid model (Fig. V.9).



a - Sitting robot tracking b - Toy object tracking c - Robot's full body tracking

Figure V.9 – 3D tracking of the Nao robot and the ball on the floor with the Kinect.

V.4 Control of a Multi-Arm System for Meat Cutting

This part presents the first application on the multi-arm platform which was presented in paragraph V.1.1.1 with the objectives of the ARMS project.

In fact, to control the 21 DOF multi-arm system, we suggest to use the previously presented methodology to control directly the platform of three robotic arms instead of controlling each robotic arm independently, as classically done.

Therefore, we use the kinematic multi-control points approach which was developed in section IV.1. Thus we begin by choosing the relevant control points which are used to define in the second part the applied generic tasks and constraints with their corresponding control laws. Finally, the appropriate redundancy resolution and task sequencing method are used to apply the desired scenario in simulation using Matlab/Simulink/ADAMS environment.

Note that the presented application focuses only on the control part of this project without approaching the issues of deformable meat modeling and detection of muscles separation area. Thus, we consider that this line is well-defined and given as an input to the control scheme.

V.4.1 Control Point Definition

To execute the desired tasks that were presented in the project's objective, the following control points (Figure V.10) will be used:

— The contact points (1) and (2) between the meat object and the cutting and pulling arms respectively.

— The control point (3) at the end-effector of the active vision robot

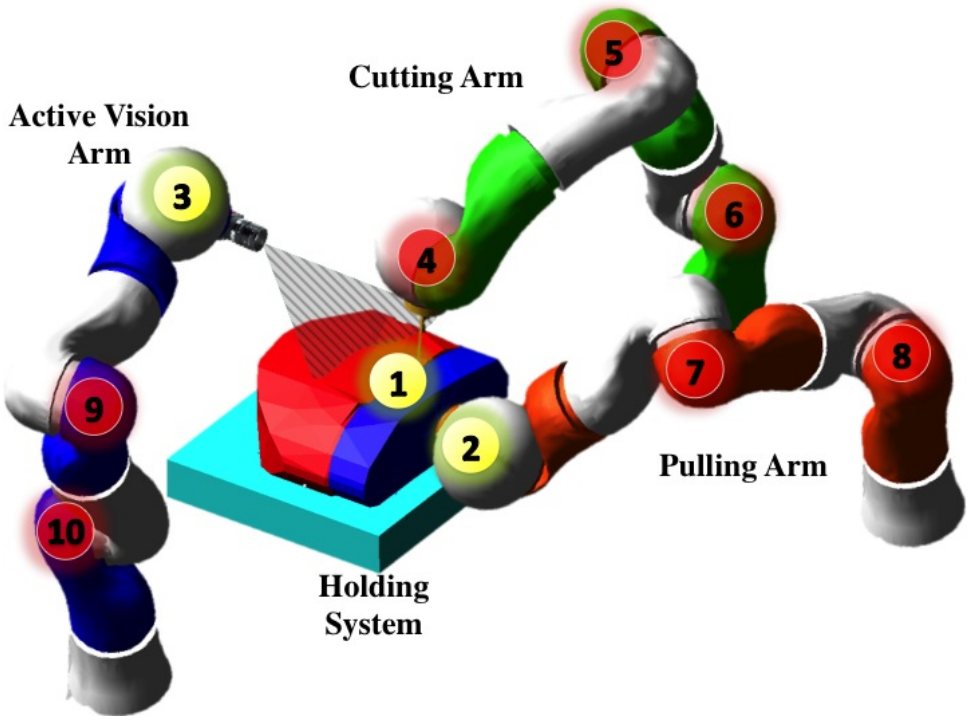


Figure V.10 – Control point definition in the multi-arm system

— The control points (4) to (10) to manage the interaction between the robots and the environment .

V.4.2 Task Definition

To execute the desired scenario of meat cutting and muscle separation, several tasks should be applied simultaneously on the multi-arm platform. In fact, the object is fixed on the table (holding system) and the first robotic arm performs a cutting task while the second arm apply pulling force to ensure the separation of the two muscles. The third arm holds a vision system at its end-effector to ensure the tracking and the visibility of the cutting line.

V.4.2.1 Cutting task

This task uses the control point (1) at the extremity of the cutting tool (knife) mounted on the first robot. It consists on moving and applying a force along the guide line between the two muscles (red and blue part in Figure V.10). Therefore, it uses the target following generic task given in (IV.19) to control the position and orientation of the cutting control point and the applied force. The target pose and force values are modified in real-time with respect to the deformation of the meat muscles and subsequently the cutting line (Figure V.11).

$$\dot{\mathbf{q}}_{\text{cutting}} = -\lambda \left(\mathbf{L}_s \, {}^{s}\mathbf{W}_{cp_1} \, {}^{cp_1}\mathbf{J}_q\right)^{+} \left(\mathbf{s}_1 - \mathbf{s}_1^*\right) \tag{V.2}$$

where the feature vector $\mathbf{s_1} = [\mathbf{r_1}(t)\,;\mathbf{F_1}(t)]$ is composed of the control point's position and applied force respectively, $\mathbf{s_1}^* = \left[\mathbf{r}_1^*(t)\,;\mathbf{F}_1^*(t)\right]$ is the corresponding desired value. The number of used DOF by this task depends on the constrained motion and force directions.

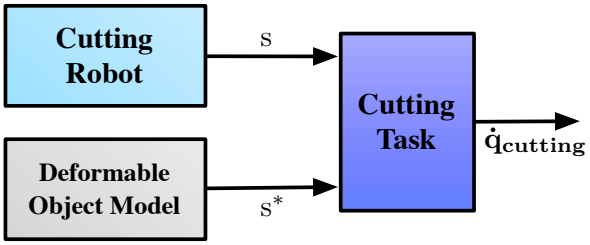


Figure V.11 – Scheme of the cutting task

V.4.2.2 Pulling task

The control point (2) at the end-effector of the second robotic arm is used to define the pulling task which is executed simultaneously with the cutting one. The object is considered to be rigidly attached to the pulling arm. Thus, this task consists on moving this control point to a desired position and orientation to ensure the separation of the two muscles and a better visibility of the cutting area (Figure V.12).

$$\dot{\mathbf{q}}_{\text{pulling}} = -\lambda \left(\mathbf{L}_s \, {}^{s}\mathbf{W}_{cp_2} \, {}^{cp_2}\mathbf{J}_q\right)^{+} \left(\mathbf{s}_2 - \mathbf{s}_2^*\right) \tag{V.3}$$

where the feature vectors $\mathbf{s_2}$ and $\mathbf{s}_2^*$ are the actual and desired control point's pose respectively. Similarly to the previous task, the number of used DOF depends on the constrained directions of motion in position and/or orientation.

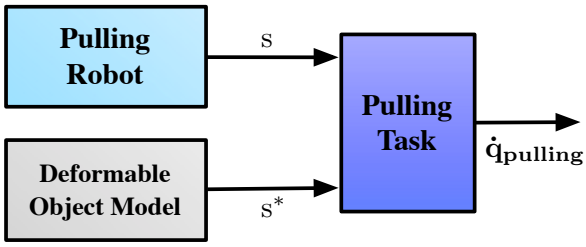


Figure V.12 – Scheme of the pulling task

V.4.2.3 Visibility task

The third main task controls the pose of the vision system mounted at the end-effector of the third robotic arm in such a way to maintain the muscles separation area in the camera's field of view. In fact, the cutting line is detected and tracked in real-time by the vision system, but to ensure that this line will not leave the field of view of the camera a visibility task is considered.

The goal of this task is to set the cutting line in the center of the camera image in a predefined orientation and at a predefined distance. Thus this task (Figure V.13) uses the 4 DOF generic visibility task with 2D segment visual feature $\mathbf{s} = (u, v, L, \theta)$ that was given by (IV.26) and (IV.27).

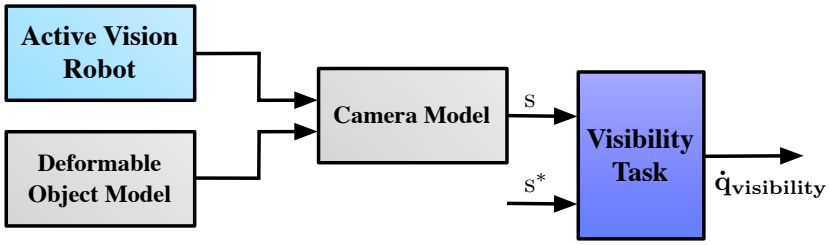


Figure V.13 – Scheme of the visibility task

V.4.3 Constraint Definition

In addition to the presented tasks, several constraints should be considered in the final control law that will be implemented on the platform. These constraints are defined to control the interaction between the different robotic arms and their environment. We note $\dot{\mathbf{q}}_{\mathbf{i-j}}$ the joint velocity corresponding to the constraint that control the interaction between the i and j control point.

V.4.3.1 Self-Collision avoidance

For each robotic arm, a self-collision avoidance constraint is defined to keep the distance, between the robot's end-effector and the position of the second robot's joint, greater than a predefined threshold. Each task uses two control points to control only the relative position between them, thus only 3 DOF are used by the generic positioning function given by (IV.19).

These self-collision avoidance constraints are noted $\dot{\mathbf{q}}_{\mathbf{4-6}}$, $\dot{\mathbf{q}}_{\mathbf{2-8}}$ and $\dot{\mathbf{q}}_{\mathbf{3-10}}$ for the cutting, pulling and vision arms respectively (Figure V.14).

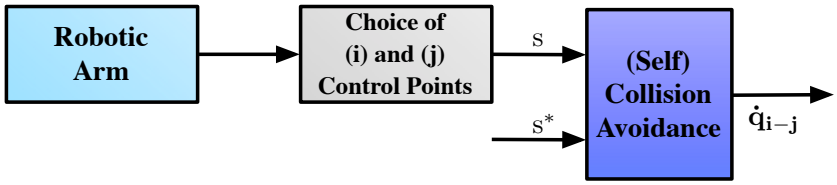


Figure V.14 – Scheme of (self) collision avoidance task

V.4.3.2 Collision avoidance

In addition to the previous constraints, the collision between the different robots should be taken into account to avoid possible contact between the robotics arms during task execution.

In fact, two constraints ($\dot{\mathbf{q}}_{4-3}$ and $\dot{\mathbf{q}}_{2-3}$) are defined to avoid the collision between the robots' end-effectors. In addition to that, three other constraints ($\dot{\mathbf{q}}_{4-9}, \dot{\mathbf{q}}_{2-9}$ and $\dot{\mathbf{q}}_{5-7}$) are used to avoid the collision between intermediate joints of the robots. As in the previous case, each constraint uses 3 DOF to control the relative position between two control points to be greater than a pre-defined threshold (Figure V.14).

V.4.3.3 Occlusion avoidance

The third type of constraints which has been considered is the occlusion avoidance one. In fact, in addition to maintaining the cutting line in the camera's field of view (which is ensured by the visibility task), another constraint should be added to avoid the occlusion of this cutting line by other robots in the platform which may lead to tracking lose.

This constraint is defined in the camera image space, it uses two control points of the platform: (3) and (4). In fact, it moves the control point (3) at the active vision arm end-effector to keep the control point of the cutting robot away from the plane formed by the camera and the cutting line. Thus two constraints $\dot{\mathbf{q}}_{\mathbf{occ}_1}$ and $\dot{\mathbf{q}}_{\mathbf{occ}_2}$ are defined to control the distance of the cutting arm to the edges of the camera's field of view (Figure V.15).

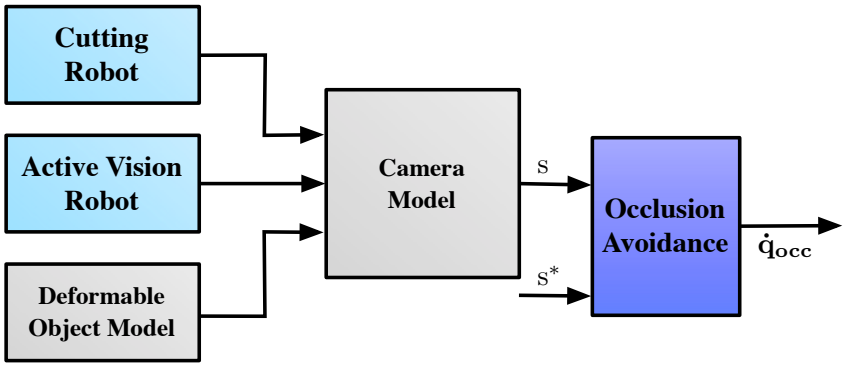


Figure V.15 – Scheme of occlusion avoidance task

Fig. V.16 shows the system from the point of view of the vision robot. A field of view is supposed as a cone with its origin at the origin of the camera. The task of the robot is to keep the cutting surface within the field of view at all times. An occlusion occurs when a foreign object prevents the vision robot from obtaining the surface parameters. The position of the node points are then exported to the simulator environment where they are used to reconstruct

a surface and thus generate a trajectory.

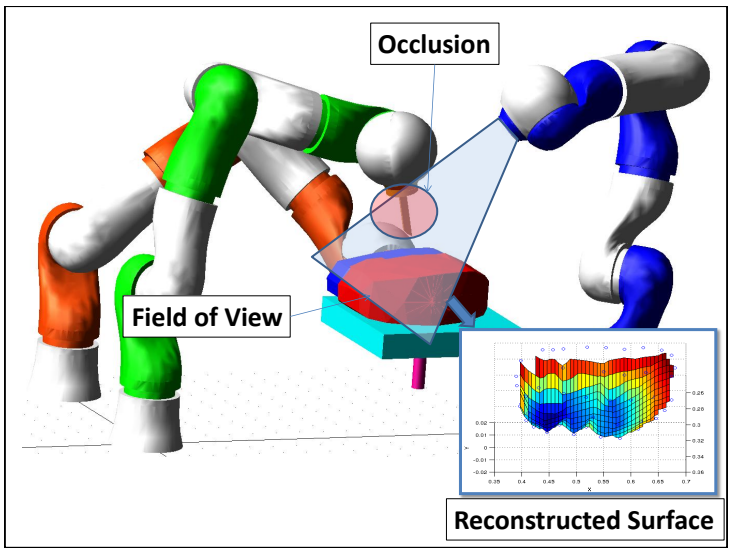


Figure V.16 – Vision System

V.4.3.4 Joint limit avoidance

In addition to the previous constraints, the classical joint limit avoidance is considered for each robotic arm to keep the joint angles in the acceptable domain. It is defined by the gradient of a function representing the performance criterion. For each of the three robotic arms, the joint velocity $\dot{\mathbf{q}}_{\mathbf{JLA}}$ is thus given by:

$$\dot{\mathbf{q}}_{\mathbf{JLA}} = \frac{\partial \mathbf{h_s}}{\partial \mathbf{q}} \tag{V.4}$$

where $\dot{\mathbf{q}}_{\mathbf{JLA}}$ is the articular velocity to perform the joints limits avoidance and $\mathbf{h_s}$ is a cost function designed to be minimal at safe configuration and maximal in the vicinity of the joint limits.

The configuration $\mathbf{q}$ of the robot is considered safe with respect to its joint limits if for all joints $q_i \in \left[q_{l_0 i}^{\min}, q_{l_0 i}^{\max}\right]$, where $\rho \in [0, 1/2]$ is a tuning parameter and $q_{l_0 i}^{\min}, q_{l_0 i}^{\max}$ are given by:

$$\begin{cases} q_{l_0 i}^{\min} = q_i^{\min} + \rho \Delta q_i \\ q_{l_0 i}^{\max} = q_i^{\max} - \rho \Delta q_i \\ \Delta q_i = q_i^{\max} - q_i^{\min} \end{cases}$$

The used cost function has a quadratic form and given by:

$$\mathbf{h_s} = \frac{\beta}{2} \sum_{i=1}^{n} \frac{\Delta_i^2}{\Delta q_i}$$

where

$$\Delta_i = \begin{cases} q_i - q_{l_0 i}^{\min}, & if \quad q_i < q_{l_0 i}^{\min} \\ q_i - q_{l_0 i}^{\max}, & if \quad q_i > q_{l_0 i}^{\max} \\ 0, & else \end{cases}$$

The classical avoidance task is thus given by:

$$\dot{\mathbf{q}}_{\mathbf{JLA}} = \frac{\partial \mathbf{h_s}}{\partial \mathbf{q}} = \beta \boldsymbol{\Delta} \quad \text{where} \quad \boldsymbol{\Delta} = \left({}^{\Delta_1}\!/_{\Delta q_1}, \quad \ldots \quad {}^{\Delta_n}\!/_{\Delta q_n} \right) \tag{V.5}$$

β is a scalar which sets the amplitude of the control law due to the secondary task. If β is too small, it may be insufficient to avoid a joint limit. If β is too large, it will result in some overshoot in the effector velocity. Therefore, it is usually set based on trial and errors.

V.4.4 Task Sequencing and Redundancy Resolution

Once all the tasks and constraints are defined, the appropriate method for redundancy resolution and task sequencing should be chosen to be use in the general control scheme (Figure V.17). In fact, for this application the platform has only 21 DOF and we define several main tasks (16 DOF) and constraints that use a large number of DOF which is greater than the available one in the platform.

Thus we suggest to use the new generalized projection operator with the hierarchical control defined in (III.9). We consider that the first three tasks are the high priority ones that should be always executed, and the other constraints as secondary tasks with less priority.

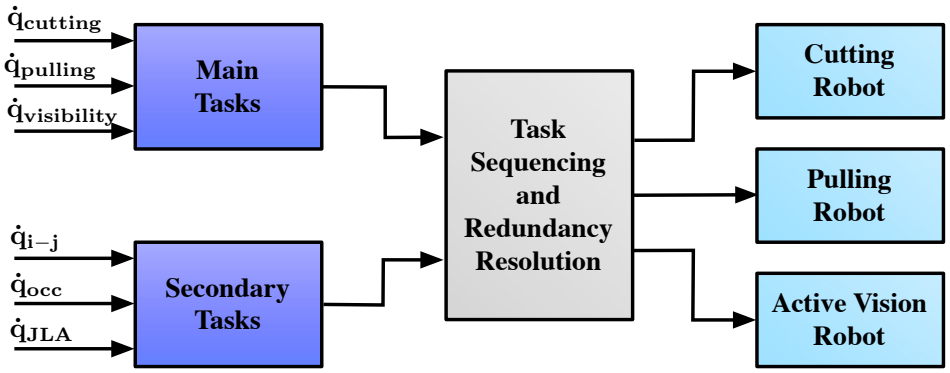


Figure V.17 – Task sequencing and redundancy resolution scheme

V.4.5 Simulation Results

Before applying the desired scenario on the real platform, the defined tasks are tested on a simulated platform. In fact, the Matlab/Simulink environment was linked to the ADAMS software to have a closed control schema of the platform (Figure V.18).

A snapshot of the system after cutting passages is given in Fig. V.19. The figure shows the gradual successful separation of the meat muscles, in particular the state of the system is shown at the end of a cutting passage. It can be seen that after each cutting passage the meat

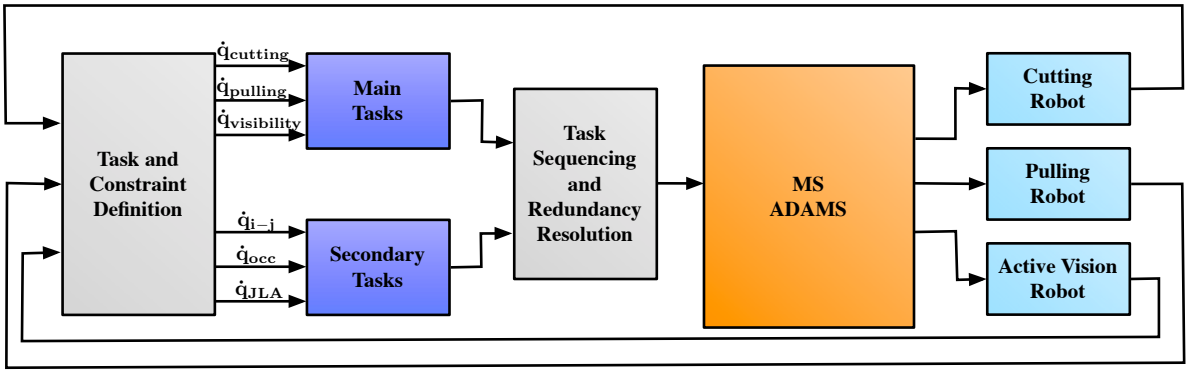


Figure V.18 – General control scheme with Matlab/Simulink and ADAMS

muscles are pulled further apart.

It should be noted that after each passage the state of the surface as changed dramatically, therefore the system must be updated by the visual primitive.

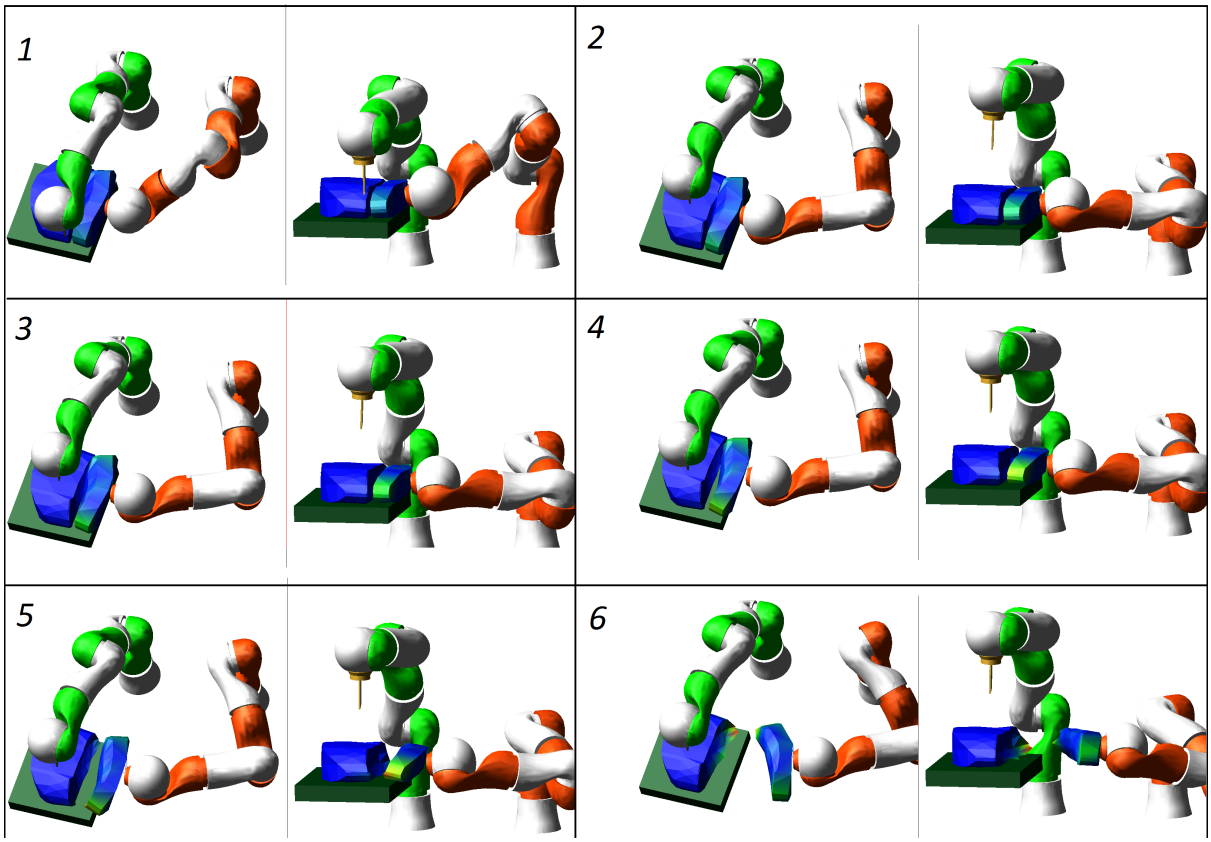
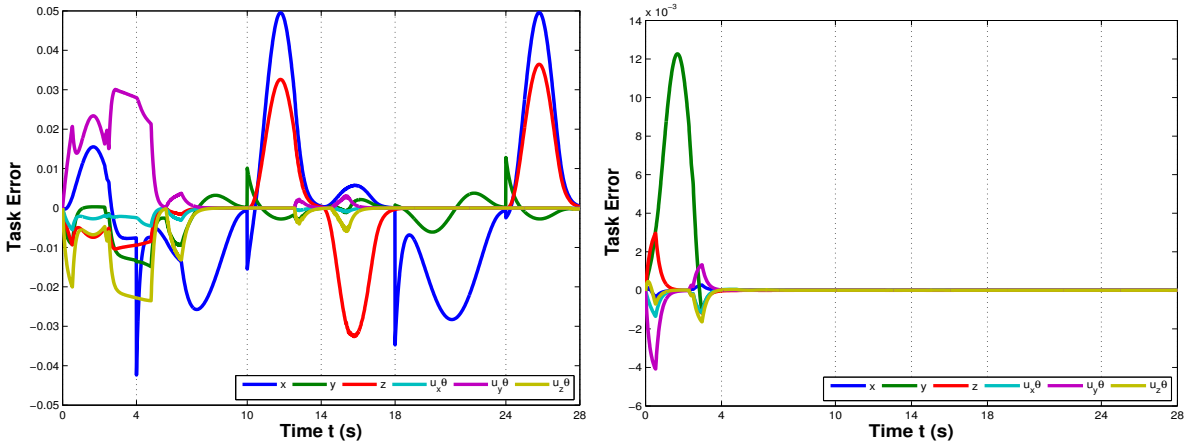


Figure V.19 – Snapshot of separation process

V.4.5.1 Main tasks

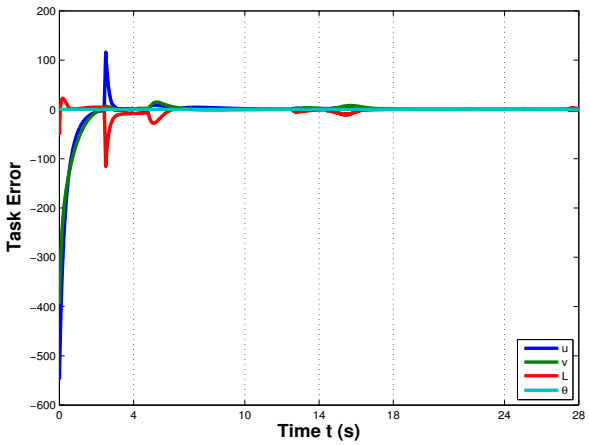
The scenario consists in applying the cutting process two times, each one takes 14 sec. In fact, the first 4 sec corresponds to the pre-cutting phase where the knife is positioned at the first extremity of the meat piece, then 6 sec for the cutting phase where the knife passes along the cutting line to the other extremity, and finally 4 sec to the post-cutting phase where the knife returns to its initial position. These three phases can be easily identified in the simulation results (Figure V.20a), where the cutting robot successfully follows the desired trajectory.

For the pulling task, the simulation results are shown in (Figure V.20b) where the task error is sufficiently small during all the task execution. For the 4 DOF visibility task, the simulation results are shown in (Figure V.20c) where the task error is exponentially regulated to zero during the different phase of the meat cutting and muscles separation process. These results ensure the successful execution of this task which consists in maintain the cutting line in the center of the image in a predefined orientation and distance by using the segment type visual feature.



a - Cutting task's error

b - Pulling task's error



c - Visibility task's error

Figure V.20 – Simulation results of the three main tasks: cutting, pulling and visibility

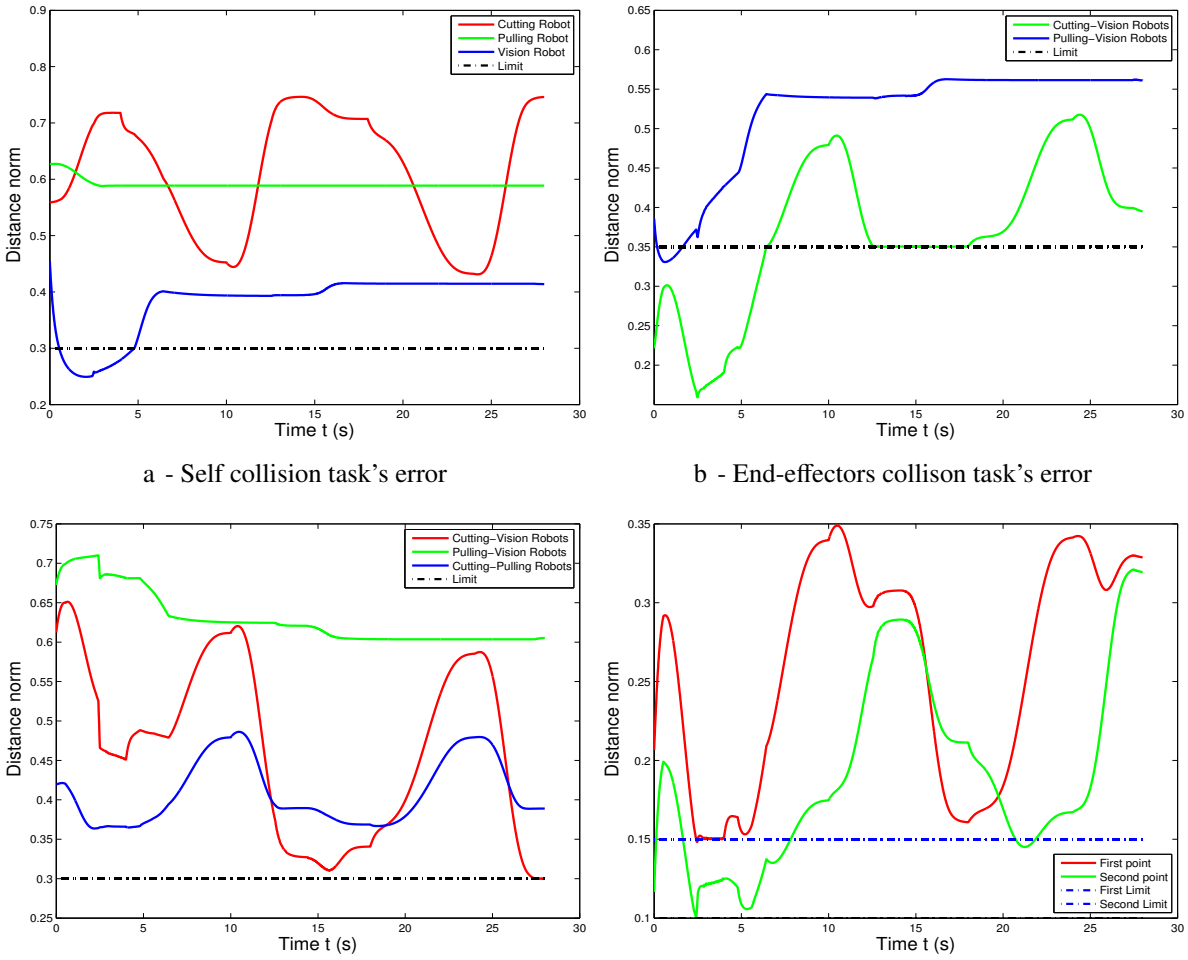
V.4.5.2 Secondary tasks

The other constraints are also considered as secondary tasks in the control law. The variation of these 9 constraints (collision and occlusion avoidance) are presented in Figure V.21.

Even in some cases the controlled constraint is initially under the desired limit, after the initialization and during the task execution this value increases and stays grater than the required threshold. The simulation results show the efficiency of using the proposed redundancy resolution technique even when insufficient number of DOF is available to execute all tasks.

Note that applying all the presented tasks and constraints simultaneously on this complex system is not possible with the classical approaches of redundancy resolution. In fact, this scenario was tested with several approaches and in particular with the orthogonal projection control method which led to unsatisfactory results and inability to meet the specified constraints.

This scenario will be validated on the real multi-arm platform of ARMS project.



a - Self collision task's error

b - End-effectors collison task's error

c - Intermediate joints collision task's error

d - Occlusion task's error

Figure V.21 – Simulation results of the secondary tasks: (self) collision and occlusion avoidance tasks

V.5 Localization and Navigation Tasks

A growing number of research studies deal with object manipulation in everyday environments, to perform assistive tasks as helping people with physical disabilities to retrieve dropped objects [JK10, KCF+12], or other service tasks such as retrieving and delivering objects at home. However, to apply such a task, we need a precise localization of the humanoid robot in its environment which is a challenging issue due to rough odometry estimation, noisy onboard sensing, and the swaying motion caused by walking [HWB10].

In this section, we present experiment results on localization and navigation tasks of the defined scenarios in section V.2. After analyzing the related state of the art, we first discuss the case of self-localization on Nao robot using the MBT technique, then the case of the robot's localization and navigation using PCL tracking technique.

V.5.1 State of the Art on Localization Methods

Accurate localization, which is considered to be mainly solved for wheeled robots, is still a challenging problem when dealing with biped robots. Many problems arise such as foot slippage, stability problems during walking, and limited payload capabilities, preventing precise localization in their environment. In addition, humanoids usually cannot be assumed to move on a plane to which their sensors are parallel because of their walking motion.

Several robot localization approaches have been proposed, with different accuracy depending on the type of sensors used for measuring the environment features: sonar, laser, robot vision, ceiling cameras, etc. Errors in sonar and laser-based localization are far from the precision needed for most manipulation tasks. Vision is the most suitable sensor for this purpose. It allows to locate quite precisely the target object in the environment and to track its motion in the case of movable system.

Furthermore, most of the indoor localization algorithms use particle based filters or Kalman type filters to solve the problem of noisy sensors and controls. Particle filters inherently help solve the problem of ambiguous landmarks, whereas Kalman filters must track multiple hypotheses to work in ambiguous environments [HK09, QM10].

Recently, Monte Carlo methods have been used to perform localization on mobile robots [TFBD01], as well as other methods including grid-based Markov localization and Kalman filtering [Gut02]. Furthermore, many studies have been made to solve the humanoid localization problem by tracking their pose in the two-dimensional space. For example, Ido et al. [ISMO09] applied a vision-based approach and compare the current image to previously recorded reference images in order to estimate the location of the robot. Owald et al. [OHB10] and Bennewitz et al. [BSBB06] compared visual features to a previously learned 2D feature map during pose tracking. Pretto et al. [PMB+09] tracked visual features over time for estimating the robot's odometry. Cupec et al. [CSL05] detected objects with given shapes and colors in the local environment of the humanoid and determine its pose relative to these objects.

In addition to that, many techniques using laser range data have also been developed: Stachniss et al. [SBG+08] presented an approach to learn accurate 2D grid maps of large environments with a humanoid equipped with a Hokuyo laser scanner. Such a map was subse-

quently used by Faber et al. [FBE+09] for humanoid localization in 2D. Similarly, Tellez et al. [TFM+08] developed a navigation system for such a 2D environment representation using two laser scanners located in the feet of the robot.

Since a 2D map is often not sufficient for humanoid motion planning, several methods use 2.5D grid maps which additionally store a height value for each cell. In [TKN06] Thompson et al. track the 6D pose of a humanoid equipped with a laser scanner in such hybrid representation. Hornung et al. [HWB10] track a humanoid's 6D pose in a 3D world model, which may contain multiple levels connected by staircases.

V.5.2 Self-Localization Task During Locomotion

In this part, we apply the scenario detailed in the section V.2.1 on the humanoid robot Nao for self-localization of the robot when walking. During the execution of this task, only a rough model of the door and a part of the room is used. In fact, the robot tracks the door and thus it is localized with respect to the environment while walking. We use the model of the door and the lines of the room around it to initialize the MBT which subsequently gives the pose of the door in the camera's frame (${}^{c}\mathbf{M}_{e}$) in real-time. From the given pose value, the robot's pose can be calculated: ${}^{e}\mathbf{M}_{n} = ({}^{c}\mathbf{M}_{e})^{-1}\,({}^{n}\mathbf{M}_{c})^{-1}$.

Note that the tracker is automatically reinitialized each time the tracking failed due to large camera displacements during walking; it uses the last found pose of the object to reinitialize the tracking. In our experiments, the robot walks in open loop in the direction of the door for a distance of 1 meter.

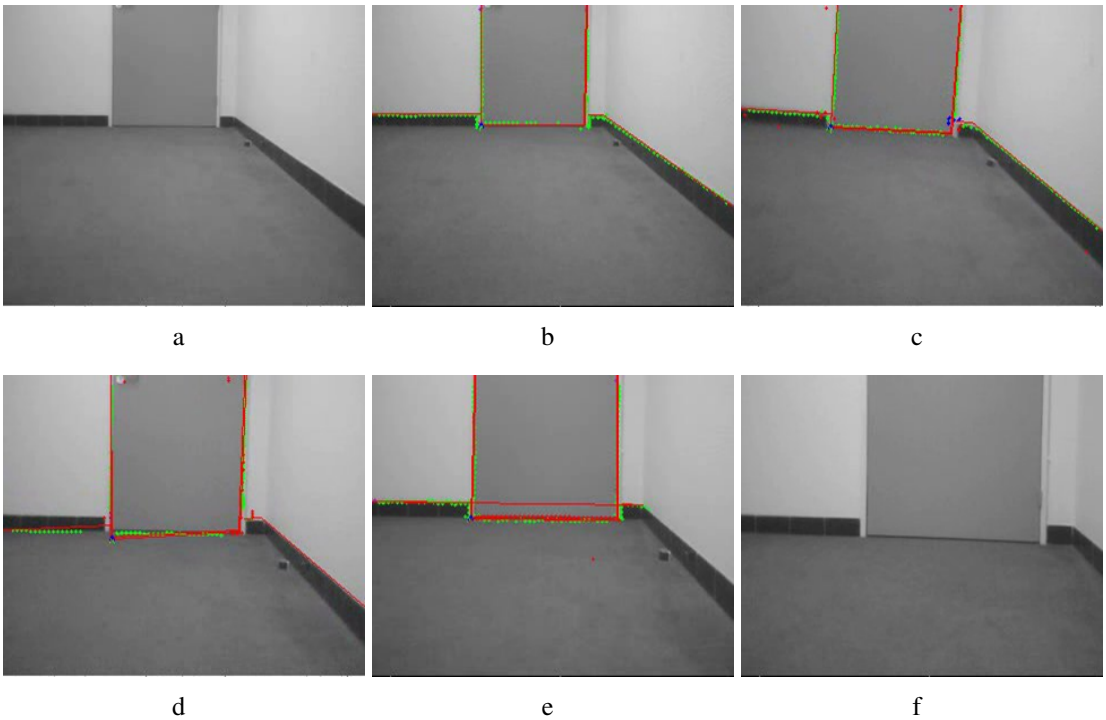


Figure V.22 – Experiment photos of self-localization task during robot's locomotion

During this experiment, we calibrated and used the top camera embedded on the robot's head with a control rate equal to that of the camera (20 Hz). Note that the presented tasks have been implemented and tested several times on the Humanoid Nao robot to ensure the efficiency of this method [1].

Fig. V.22a shows the robot's environment before launching the tracking and locomotion tasks. Then Fig. V.22b to Fig. V.22e show the tracking of the door during the task's execution, and when the task is completed (Fig. V.22f). In Fig. V.23, the distance between the robot's frame $\mathcal{F}_n$ and the origin of the door's frame $\mathcal{F}_e$ is plotted, in addition to the X, Y and Z components of this pose in $\mathcal{F}_n$. We can notice that the distance decreases from 3.94 m to 2.88 m.

Therefore, these results show that the robot successfully tracks the door while walking the desired distance (1 m) with a final error of 6 cm, which allow us to localize the robot successfully.

For the self-localization task, the error may be relatively large with respect to other localization methods, but it is acceptable and sufficient when dealing with indoor locomotion for manipulation tasks, because of the high robustness of the MBT technique and the implemented automatic re-initialization of the tracking process.

In Fig. V.23, between the 10^{th} and 32^{nd} sec, we can identify some disturbances which are

1. See a video of the applied tasks on Nao robot on www.youtu.be/WflA_jRBMkM

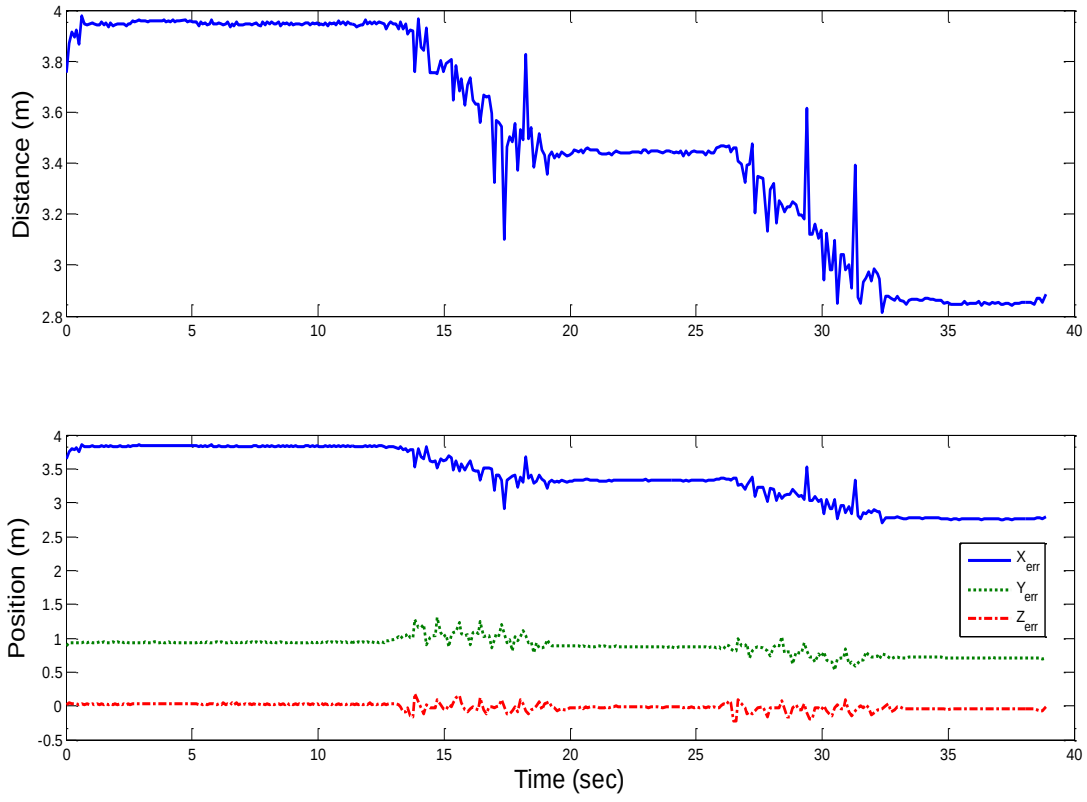


Figure V.23 – Experimental results of the self-localization task in Nao's frame

caused by the displacement of the camera during the robot's locomotion. During this period, the automatic re-initialization module of the tracking is used to prevent localization task from failure. To improve this method, an additional module could be used to dynamically compensate the camera's motions during robot's walking (using data from the robot's accelerometer/gyrometer), or by fusion with other sensors data; for example using an external camera on the ceiling of the room to provide a wider view and more visual data, or the Nao's odometry module.

V.5.3 Localization and Navigation of the Robot for Assistive Task

In addition to the technique applied in the previous paragraph, most of the methods cited in the state of the art part used either embedded cameras and sensors on the robot or exteroceptive ones which are mounted on the robot's head or body. All these sensors are usually used to track the environment and subsequently localize the robot.

In this part, external sensors are fixed in the robot's environment and are used to localize the walking robot and the desired objects. Thus, the contribution of this method relies on the development of a robust closed-loop navigation system for humanoid robots in indoor environments using localization with Kinect cameras. Thus, the main goal is to develop a robust system which can track and localize the robot when walking, and to control its motion precisely enough, to retrieve an object from the floor.

In contrast to the first application where it is considered as stand-alone self contained robot, Nao robot benefits from the information coming from other external sensing devices in this application. In order to achieve a robust localization while walking, and to retrieve an object from the floor, an external Kinect sensor is used to integrate RGBD camera information in the control law. In fact, Monte Carlo localization estimates the 6D torso pose of the robot, which is then used for closed-loop navigation control. Experimental results with Nao robot show that, by cooperating with the environmental sensors, the overall precision of humanoid robot navigation is dramatically improved.

V.5.3.1 Robot's tracking and localization

The 3D tracking technique of PCL tracking library (presented in paragraph V.3.4) consists of tracking 3D objects (position and orientation) in continuous point cloud data sequences. It was originally designed for robots to monitor the environment, make decisions and adapt their motions according to changes in the world, but it is equally suitable for tracking the robot while it walks around the environment (as seen in Fig. V.9).

In our application, we use a rigid model of the torso and head parts of the robot, to track the system on real-time and find the 3D pose of the robot model even when walking. In future works, an articulated model could be used to track the whole body of the robot.

Using the PCL Cloud tracking technique we can find the actual pose of the robot and the desired one (the object pose) with respect to the Kinect. Thus, the relative 3D pose can be calculated, and we can control the direction of walking in the plane: the position in X and Y directions and the orientation of the robot.

V.5.3.2 Robot's navigation

The Nao robot is able to walk in velocity control mode. This enables the walk to be controlled reactively, as the most recent command overrides all previous commands. However, the walk uses a preview controller to guarantee stability. This uses a preview of time of 0.8 sec, so the walk will take this time to react to new commands. At maximum frequency this equates to about two steps [Rob].

The control law is derived using the classical sensor-based technique [SLBE91]. Regardless the sensor's configuration, a set of visual features $\mathbf{s}$ has to be designed from the visual measurements obtained from the system configuration. A control law is thus designed so that these features $\mathbf{s}$ reach a desired value $\mathbf{s}^*$, defining a correct realization of the navigation task.

The generic positioning task previously defined in paragraph IV.3.1 could be used to move the robot to the desired pose. Thus, if the velocity $\mathbf{V}$ is considered as input of the robot controller, the control law which performs the desired exponential decoupled decrease of the error $\mathbf{e} = (\mathbf{s} - \mathbf{s}^*)$ is given by:

$$\mathbf{V} = -\lambda \mathbf{L}_s^+ (\mathbf{s} - \mathbf{s}^*) \tag{V.6}$$

where λ is the classical proportional gain that has to be tuned to minimize the time to convergence of the task, and $\mathbf{L}_s^+$ is the Moore-Penrose pseudo-inverse of the interaction matrix $\mathbf{L_s}$.

In our case, the considered feature is the relative position/orientation between the robot and the ball on the plane: $\mathbf{s} = [x, y, \theta_z]^\top$. For the desired feature value $\mathbf{s}^*$, the following strategy is considered: first, while the error is greater than a given threshold e_{min}, the robot will head towards the line joining its current location and final destination; second, when the error is lower than such threshold, the robot will head to its final orientation:

$$s^* = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ atan(y/x) \end{bmatrix} & if \quad \|\mathbf{e}\| > e_{min} \\ \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & if \quad \|\mathbf{e}\| < e_{min} \end{cases} \tag{V.7}$$

The calculation of the interaction matrix $\mathbf{L_s}$ used in the control law (V.6) gives:

$$\mathbf{L_s} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{y}{x^2+y^2} & \frac{-x}{x^2+y^2} & 1 \end{bmatrix} & if \quad \|\mathbf{e}\| > e_{min} \\ \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & if \quad \|\mathbf{e}\| < e_{min} \end{cases} \tag{V.8}$$

V.5.3.3 Robot's veering correction

Closed-loop control is able to converge in the presence of uncertainties in sensors and actuators. However, faster convergence is achieved if the system is properly calibrated. In a similar way to human beings [KSL07], biped robots suffer from inability to maintain a straight path when walking without vision: slight differences between each leg stride caused by back-slash, friction, or motor power will produce a veering behavior which can be estimated and corrected. Therefore, a simple veering correction procedure is now introduced: the robot is commanded in open-loop to walk straight away with a constant linear velocity. Its trajectory is recorded with the Kinect sensor, and two fitting steps are performed:

(a) **Circle fitting**:
The robot's trajectory while commanded in open-loop to walk straight with constant linear velocity is shown in Fig. V.24, with the best fitting circle. In fact, without correction, for a 1 meter walked distance, the lateral error grows to 35 cm, and the orientation error is 30°.

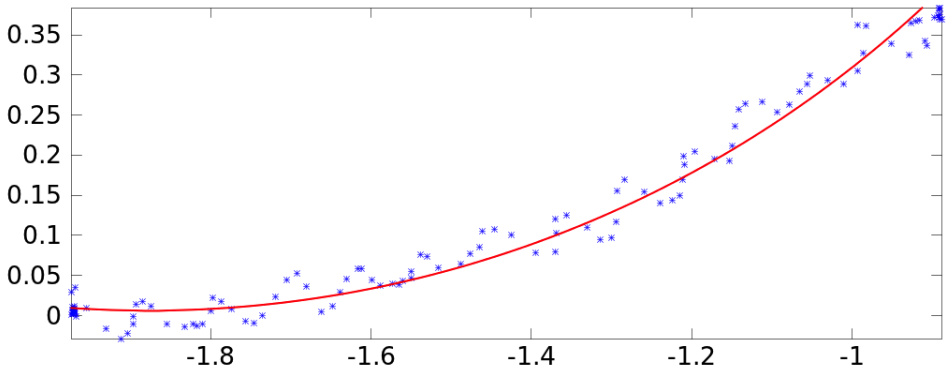


Figure V.24 – First step of veering correction: the robot's trajectory and the best fitting circle

(b) **Line fitting**:
The robot is now commanded to walk with a constant linear velocity, and a constant angular velocity (correction), as computed from the previous fitting step. Fig. V.25 depicts the recorded trajectory, and the LMS (Least Median-of-Squares) line fitting. As shown in the figure, with angular correction, the orientation error is not noticeable, but a lateral error persists, 25 cm for a 1.5 m walked distance.

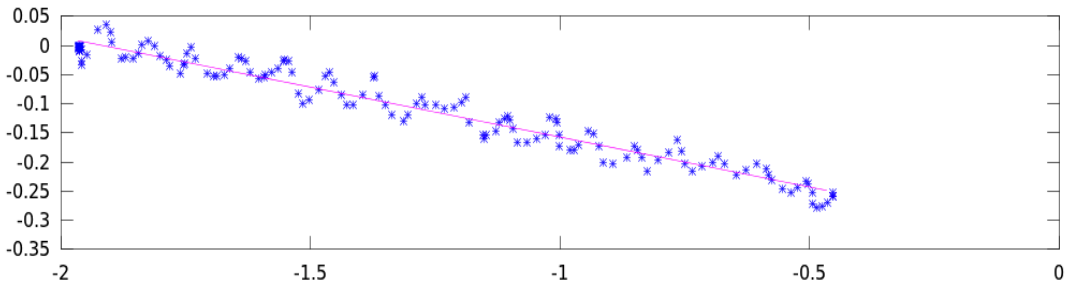


Figure V.25 – Second step of veering correction: Linear fitting of the new trajectory

(c) **Veering correction result**:

This magnitude of later error is compensated with a lateral velocity term, resulting in only 3 cm for the same walked distance (Fig. V.26). Thus, the error is significantly reduced by using this veering correction technique :

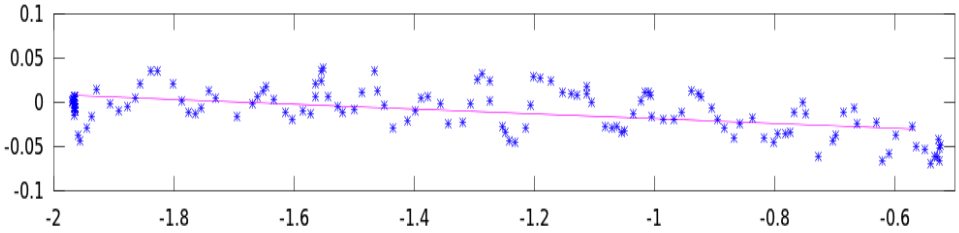


Figure V.26 – Veering correction result: the final trajectory with angular and lateral compensation

As a result, for a given command motion, the actual velocity sent to the robot $\mathbf{V_r}$ consists of the original commanded values $\mathbf{V}$ with compensation terms for the lateral and angular velocities. The radius of the fitting circle R and the slope of the fitting line m are used to compute the angular and lateral velocity compensation respectively.

Thus, the control law given in (V.6) is modified to include the veering compensation matrix $\mathbf{L_{veering}}$ as below:

$$\mathbf{V_r} = -\lambda\, \mathbf{L_{veering}}\, \mathbf{L}_s^{+}(\mathbf{s} - \mathbf{s}^*) \tag{V.9}$$

$$\text{with } \mathbf{L_{veering}} = \begin{bmatrix} 1 & 0 & 0 \\ -m & 1 & 0 \\ -R & 0 & 1 \end{bmatrix}$$

Finally, the velocity is translated to Nao's walk arguments (see [Rob] for details).

V.5.3.4 Experimental results

In the experimental part [2], the Nao robot is commanded to approach an object lying on the floor as seen in Fig. V.27. The initial distance to the object is 1.5 m, and the orientation of the initial pose with respect to the final pose is 25°. Figures. V.28, V.29 and V.30 depict respectively the commanded velocity, the pose error, and the planar trajectory of the robot.

As can be seen in Fig. V.28, the robot initially walks towards the destination at full speed, then it progressively decreases its velocity as the error is diminished. The profile of the angular velocity reflects the two stages defined in (A.16): from 0 sec to 20 sec, the robot turns towards the destination; after 20 sec the robot is nearer to the destination than the threshold (fixed to 30 cm) and it turns to its final orientation.

2. A video of the Nao robot while grasping a dropped object from the floor is available on www.youtu.be/SITPE9rgXsM

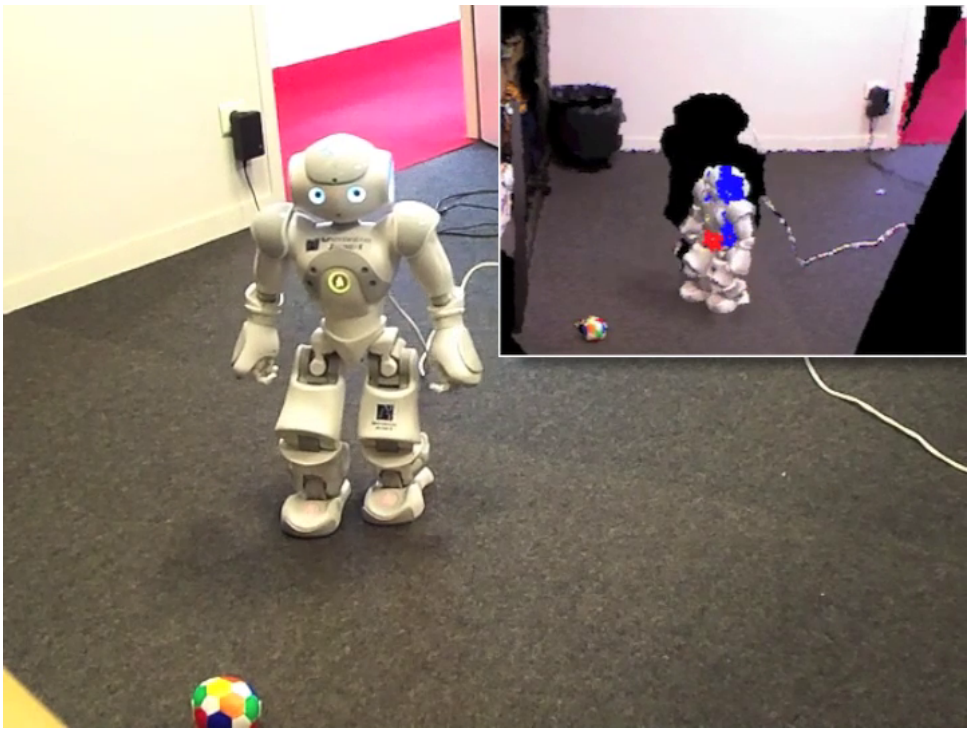


Figure V.27 – Experimental setup (external and Kinect views): as Nao robot enters the room, it is commanded to move towards an object lying on the floor (the ball on the left-bottom corner of the image).

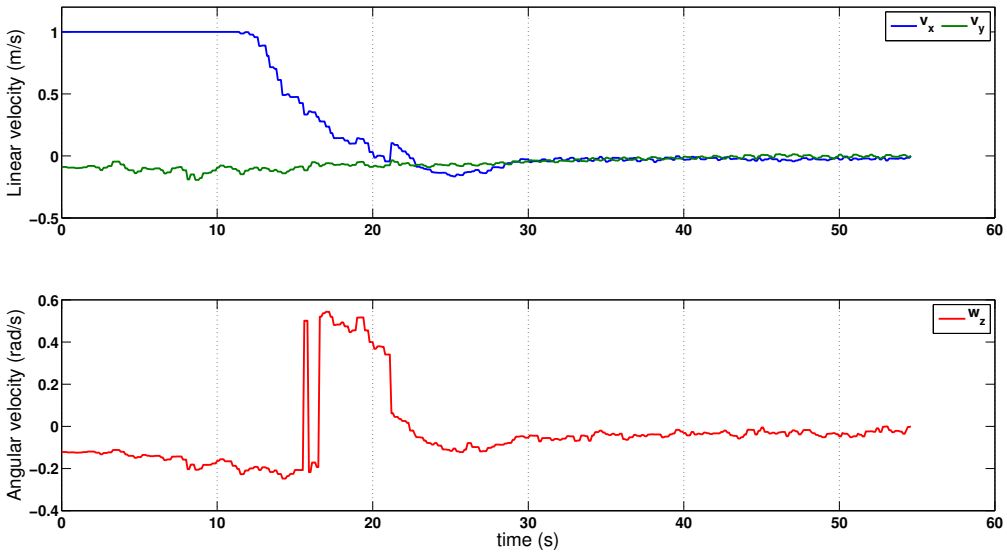


Figure V.28 – Robot velocity in Nao's local frame: only planar and angular velocity are sent to the robot controller.

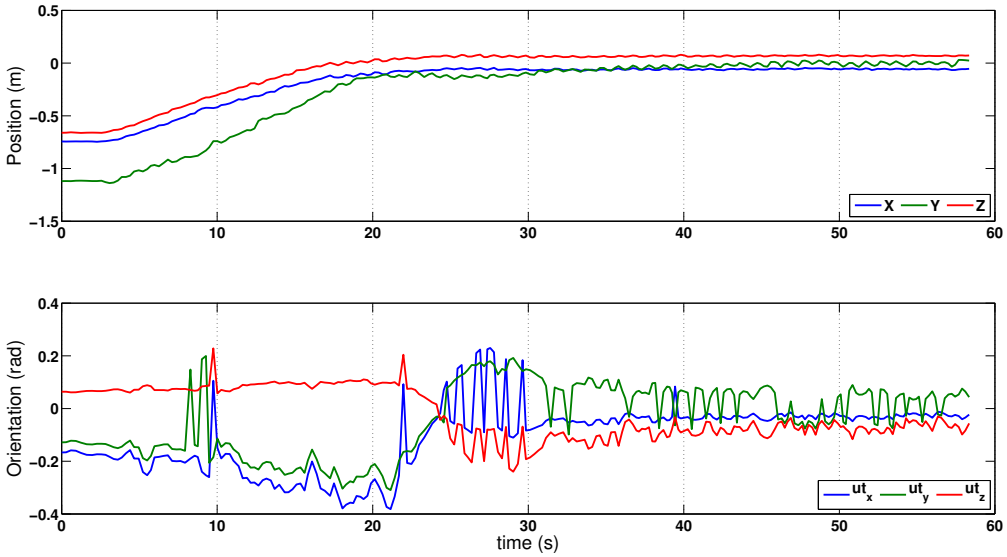


Figure V.29 – Pose error of the robot torso (position and orientation), as measured by the Kinect sensor.

This control strategy explains why the angular error (ut_z) in Fig. V.29 does not decrease initially since this error is measured with respect to the final orientation of the robot.

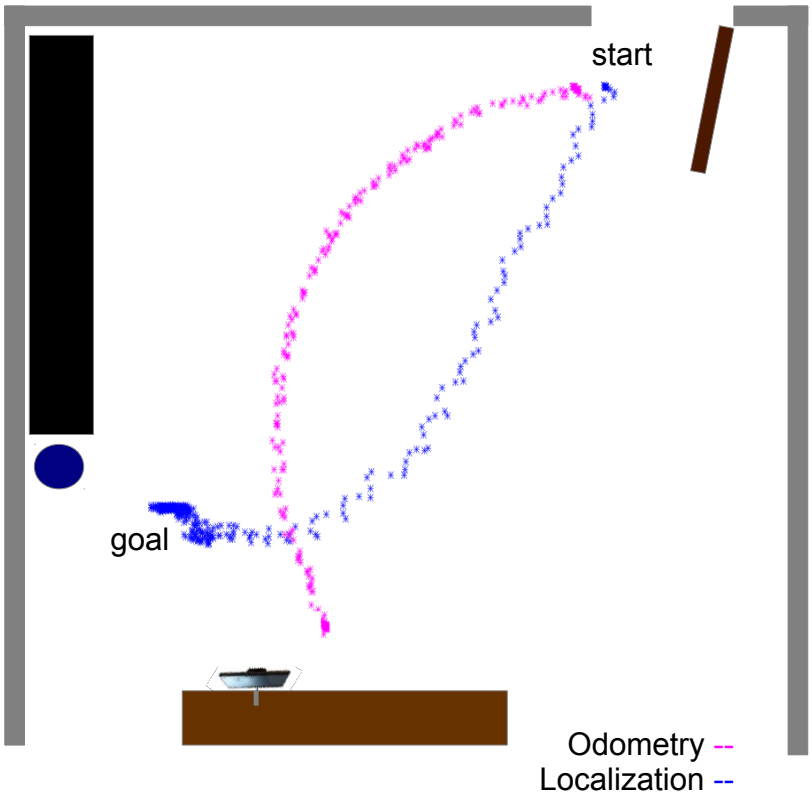


Figure V.30 – Trajectories carried out in the experiments: while the odometry diverges, localization is able to attain the goal.

The trajectory of the robot in the room is depicted in Fig. V.30. When using only odometry, the robot is not able to attain the destination goal, the final pose is 40 cm away from the desired one. However, localization and closed-loop navigation allows the robot to attain the goal within a few centimeters precision.

The final mean position error is $(x, y, \theta_z) = (0.012, 0.018, 0.07)$ and the standard deviation is $(\sigma_x, \sigma_y, \sigma_\theta) = (0.002, 0.004, 0.05)$.

Once the robot arrived to the desired position/orientation, the navigation task is finished and the robot successfully grasps an object from the floor (as shown in the video). Repeated experiments show that the localization error given by the proposed method above is sufficient to fetch the dropped object on the floor.

In the current implementation, the system is able to detect small objects lying on the floor plane, as well as to localize the robot. Later, we can incorporate the skeletal tracking of the Kinect sensor, to be able to interface directly with human gestures.

V.5.3.5 Conclusion

In this part, we validated the presented localization and navigation method for Nao humanoid robot in a smart environment, where an external Kinect sensor is used for monitoring and tracking the 3D pose of the robot. This method is suitable for indoor environments, allowing not only to detect the robot and the object but also to track people who interact with the robot.

Experiment results show the significant improvement of this navigation method over other odemetry-based techniques. In fact, the accomplished localization precision allows to retrieve objects from the floor for assistance and service robotics.

V.6 Manipulation Tasks with Humanoid Robots

V.6.1 State of the Art

Many approaches have been used to apply service tasks on humanoid robots [BKX08]: several works have been carried out in the area of teleoperation, by controlling a robotic system to perform distance tasks using a multi-modal human-system interface which provides sensory feedback to the operator and allows him to interact with the remote environment by mapping his actions (Fig. V.31). It was applied to bi-manual manipulation and walking [EMS+09], haptic interface for mobile teleoperators [NS04] and recently on a teleoperation system with a haptic device for micro-manipulation [HSE+11].

Furthermore, many practical learning control systems are used to control complex robots involving multiple feedback sensors and multiple command variables during both repetitive and nonrepetitive operations [MI87]. The issue of teaching a robot to manipulate everyday objects through human demonstration has been studied by [DA10] who proposed a method that enables a robot to decompose a demonstrated task into sequential manipulation primitives, series of sequential rotations and translations [JK09].

Other earlier works used also the Model-Based Control strategies, executed by automatically generating a control sequence that moves the robot to the states specified by the program to develop executives that emphasize model-based approaches and deep integration of automated planning [MSCY07].

Figure V.31 – Humanoid robots control by teleoperation and imitation

V.6.2 Manipulation with HRP-2 Robot

In this section, we apply a manipulation scenario with the humanoid robot HRP-2. The architecture of the robot was presented in paragraph V.1.1 with the used simulator (OpenHRP) and the external camera.

The kinematic multi-control points approach is applied on the humanoid HRP-2 for several manipulation tasks. In fact, the entire structure is controlled to maintain the robot's equilibrium in a vertical position, and to apply the rigid object grasping tasks using the right and left grippers.

Later on, we consider 6 control points on the robot's body (as represented in Fig. V.6b): the left and right feet, the left and right wrists, the robot's head and the CoM of the robot. On the next, after defining the desired scenario and the relevant control points, we use the already presented generic tasks to define the appropriate tasks to be applied and their corresponding control laws. Later on, the stack of tasks will be used to stack and manage the priority between these tasks.

V.6.2.1 Equilibrium task

An essential purpose in the robot's control is to maintain the equilibrium and stability of the system. From the available motion patterns of the humanoid robots, the 3D translational trajectory is suggested most frequently for the waist in order to imitate human motion and reduce energy consumption. In order to test a static equilibrium condition of the robot there is a need to establish its support region. Thus, the CoM of the robot must satisfy constraints represented below to keep its stability:

$$\begin{cases} \sum_{i=1}^{n} \mathbf{F}_i + m\mathbf{g} = 0 \\ \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{F}_i + \mathbf{c} \times m\mathbf{g} = 0 \end{cases}$$

where $\mathbf{F}_i$ is the vector of a reaction force at each leg, $\mathbf{r}_i$ is the position vector of the i^{th} leg contact point, $\mathbf{c}$ is the position vector of the COM, m is the mass of the robot, $\mathbf{g}$ is the gravity force vector.

Since we are studying the static robot's behavior at the kinematic level with hands manipulation, the equilibrium constraint can be simplified to regulate the robot's CoM to a desired pose only. In this case, the equilibrium task is a particular case of the generic positioning task defined in section IV.3.1.

The used control point for the execution of this task is the CoM of the robot (Torso frame $\mathcal{F}_t$ in Fig. V.6b), data is provided from the proprioceptive sensors (gyroscope) of the robot. The goal of this task is to control the 3D pose of this control point. Two possible cases can be considered: if the goal is to maintain the equilibrium of the robot (i.e. the projection of the CoM should be in the stability region) thus the control of the position of the CoM is sufficient and a 3 DOF task can be defined. In the other case, if a fixed posture (with fixed orientation) of the robot is desired, 6 DOF are constrained by this task.

In our simulations, we consider the first case, thus only 3 DOF are constrained and the feature $\mathbf{s} = (x_t, y_t, z_t)$ is regulated to $\mathbf{s}^* = (x_t^*, y_t^*, z_t^*)$, using the control law in (IV.10). Note that the used sensor directly gives the position of the CoM in $\mathcal{F}_t$ frame; thus the matrices $\mathbf{L_s}$ and ${}^s\mathbf{W}_t$ are equal to the identity matrix, and the ${}^t\mathbf{J}_q$ is given by the robot's model.

Finally, the control law for this task is thus given by:

$$\dot{\mathbf{q}}_\mathbf{t} = -\lambda \, {}^\mathbf{t}\mathbf{J}_\mathbf{q}^+ \, (\mathbf{s} - \mathbf{s}^*) \tag{V.10}$$

V.6.2.2 Posture task

In our experiments, a static manipulation is executed; thus, the robot's legs evaluate with respect to the floor. Therefore, to keep a fixed posture of the lower part of the robot a task is defined to maintain a constant relative distance/orientation between the ankles of the robot. Thus, the generic cooperative task defined in section (IV.3.2) is used to keep a fixed relative pose between the two control points on the robot's feet ($\mathcal{F}_{lf}$ and $\mathcal{F}_{rf}$ frames in Fig. V.6b).

The considered features are thus the corresponding 3D points: $\mathbf{s}_1 = (x, y, z, u_x\theta, u_y\theta, u_z\theta)^\top_{lf}$ and $\mathbf{s}_2 = (x, y, z, u_x\theta, u_y\theta, u_z\theta)^\top_{rf}$.

Using the jacobian matrices (${}^{\mathbf{rf}}\mathbf{J_q}, {}^{\mathbf{lf}}\mathbf{J_q}$), interaction matrices ($\mathbf{L_{s_1}}, \mathbf{L_{s_2}}$) at the control points, and transformation matrices (${}^{\mathbf{s_1}}\mathbf{W_{lf}}, {}^{\mathbf{s_2}}\mathbf{W_{rf}}$) as defined in (IV.20) and (IV.21) respectively, we can define the corresponding control law as in (IV.22):

$$\dot{\mathbf{q}}_{\mathbf{feet}} = -\lambda \begin{bmatrix} -\mathbf{L_{s_1}}\, {}^{\mathbf{s_1}}\mathbf{W_{lf}}\, {}^{\mathbf{lf}}\mathbf{J_q} & \mathbf{L_{s_2}}\, {}^{\mathbf{s_2}}\mathbf{W_{rf}}\, {}^{\mathbf{rf}}\mathbf{J_q} \end{bmatrix}^+ (\mathbf{s} - \mathbf{s}^*) \tag{V.11}$$

Note that when applying this task with the previous one, the CoM and the relative pose between legs will be fixed, thus the lower part of the robot will be in a stable position and will not fall down during the application of manipulation tasks.

V.6.2.3 Visibility task

The control point for this task is located in the neck articulation ($\mathcal{F}_c$ frame in Fig. V.6b), it is used to manage the orientation of the robot's head, i.e. the visibility task (presented in paragraph IV.3.3).

Between the discussed definitions, we use, in this simulation, only a 2 DOF task to center 2D pose of the gripper's center in the camera image plane. The interaction matrix is given by (IV.23) and the generic control law (IV.10) is used:

$$\dot{\mathbf{q}}_{\mathbf{visib}} = -\lambda \left(\mathbf{L_s}\, {}^{\mathbf{cam}}\mathbf{W_c}\, {}^{\mathbf{c}}\mathbf{J_q}\right)^+ (\mathbf{s} - \mathbf{s}^*) \tag{V.12}$$

where

$$\mathbf{L_s} = \begin{pmatrix} -1/Z_c & 0 & x_c/Z_c & x_c y_c & -\left(1 + y_c^2\right) & y_c \\ 0 & -1/Z_c & y_c/Z_c & 1 + x_c^2 & -x_c y_c & -x_c \end{pmatrix}$$

V.6.2.4 Grasping task

The generic positioning task defined in section (IV.3.1) is used to grasp a fixed object by the right and left robot's gripper. The control law uses the classical one defined on (IV.10).

The control law of "GraspRight" task which moves the right hand gripper to a predefined pose (6 DOF task) is given by:

$$\dot{\mathbf{q}}_{\mathbf{GraspRight}} = -\lambda\, {}^{\mathbf{rh}}\mathbf{J_q}^+ (\mathbf{s} - \mathbf{s}^*) \tag{V.13}$$

where $\mathbf{s} = (x, y, z, u_x\theta, u_y\theta, u_z\theta)^\top_{rh}$

For the "GraspLeft" task, the left hand gripper is only controlled to move to a desired position (3 DOF). In addition, an external camera is integrated in the OpenHRP simulator to track and grasp an object (that define the target frame $\mathcal{F}_{target}$). Thus, the 3D model-based tracking technique is used to track and get the 3D pose of a rigid object which is used to apply the 3 DOF grasping task using the following control law:

$$\dot{\mathbf{q}}_{\mathbf{GraspLeft}} = -\lambda \left(\mathbf{L_s}\, {}^{\mathbf{lh}}\mathbf{J_q}\right)^+ (\mathbf{s} - \mathbf{s}^*) \tag{V.14}$$

where $s = (x, y, z)^\top_{lh}$ and $s^* = (x, y, z)^\top_{target}$

V.6.2.5 Stack of Tasks

The presented tasks should be executed simultaneously on the HRP-2 robot, thus a task sequencing formalism is required to manage the priority between them. We apply in this simulation the directional projection technique and its corresponding task sequencing method known as Stack of Tasks (SoT), which is detailed in section (II.4.3).

The core architecture of the SoT is its work flow-based representation that leads to a clear graphical representation of data dependency and avoids re-computation of an expensive piece of data by implementing automatic caching. However, to fill the gap between solving an opti-

Priority	Task name	Used DOF	Control points
1	Equilibrium Task	3 DOF	Center of Mass (6)
2	Posture Task	6 DOF	Left (1) and Right (2) Foot
3	Grasping Task (GraspLeft)	3 DOF	Left gripper (3)
4	Grasping Task (GraspRight)	6 DOF	Right gripper (4)
5	Visibility Task	2 DOF	Robot's head (5)
6	Gripper Task	2×1 DOF	Left (3) and Right gripper (4)

Table V.4 – Summary of the simulated tasks with their corresponding control points, the number of constrained DOF and their priority

mization problem using the SoT and creating a real robotics application, third party softwares should be integrated through CORBA. The latter allows remote procedure calls, and provides the possibility to interact with the StackOfTasks by creating, reading and sending signals as well as sending script commands.

In our simulations, we used CORBA service to implement clients which communicate and exchange data between the tracking tools of ViSP and the SoT.

The final step in the multi-control points approach is the definition of tasks priority. For the defined scenario, the equilibrium task is considered as the highest priority task, then the task to control the robot's posture, then the grasping task and finally the visibility task is the lowest priority one. The applied tasks are summarized in Table V.4, with the corresponding control points, number of used DOF and the task priority.

Note that two tasks, of 1 DOF each, are added to close the right/left grippers on the object once the manipulation task is accomplished (Gripper task).

V.6.2.6 Simulation results

The presented tasks are simulated on OpenHRP using an external camera and the simulation results are plotted in Fig. V.33. The first two graphs (Fig. V.33a, V.33b) represent the error variation for the equilibrium and posture tasks. The first one maintains the robot's center of mass on it's initial position, and the second one maintains the initial relative pose between the robot's legs. These high priority tasks are conserved and respected during all the manipulation process.

In Fig. V.33c, the "GraspLeft" task is represented, this task consists in grasping the object in Fig. V.8b by the robot's left hand and using an external camera to apply the object tracking (using MBT of ViSP) and thus to determine the desired position of the task. Small oscillations can be detected in the graph due to the object's tracking algorithm, however it has no influence on the task execution and robustness.

Simultaneously, the "GraspRight" task moves the robot's right hand to a predefined desired pose (6 DOF). The execution of this task is represented in Fig. V.33d where the error exponentially converges to zero. For the last task, the 2 DOF visibility task moves the robot's head to put the desired object in the center of the image. The error regulation of this task is represented in Fig. V.33e.

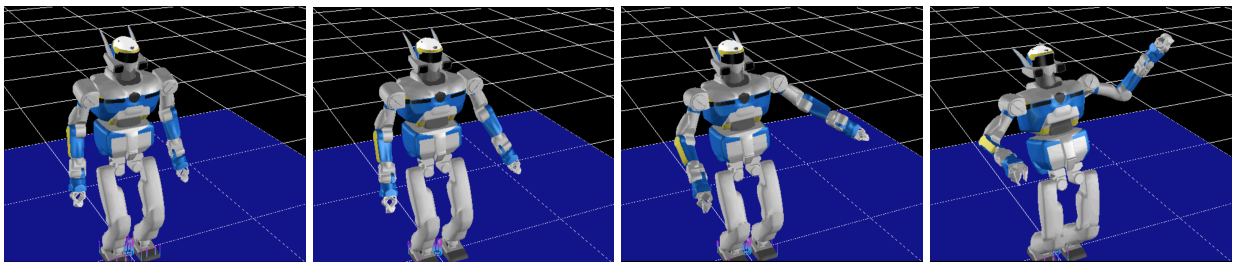
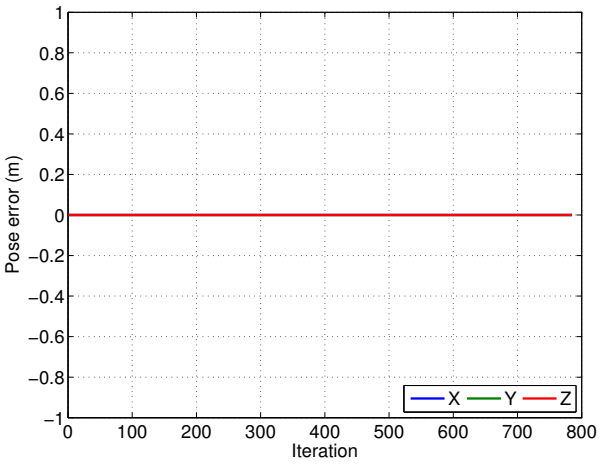
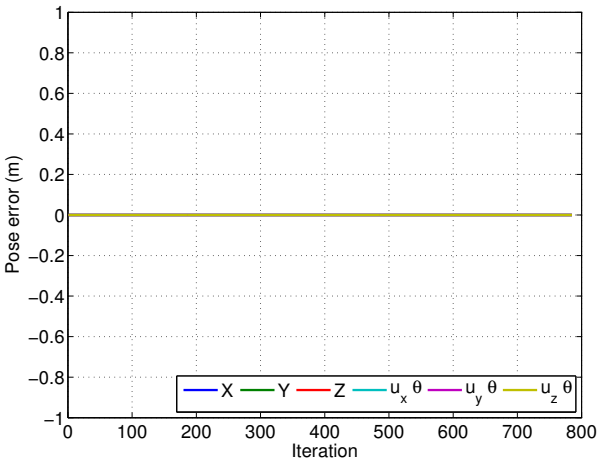


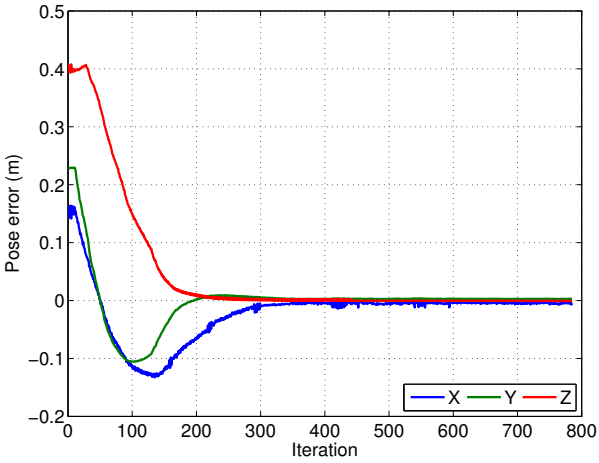
Figure V.32 – HRP-2 robot while applying equilibrium, visibility and manipulation tasks



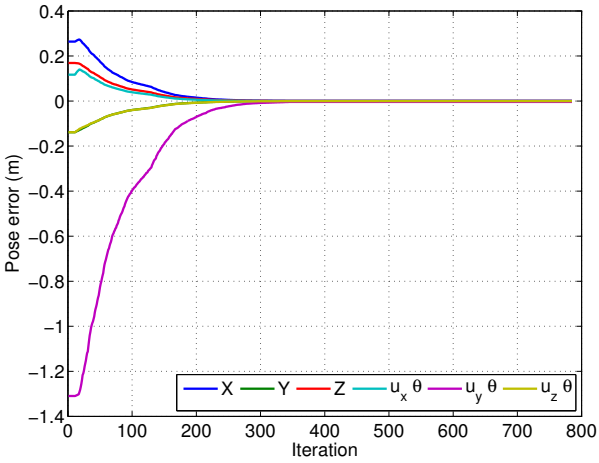
a - Equilibrium task error



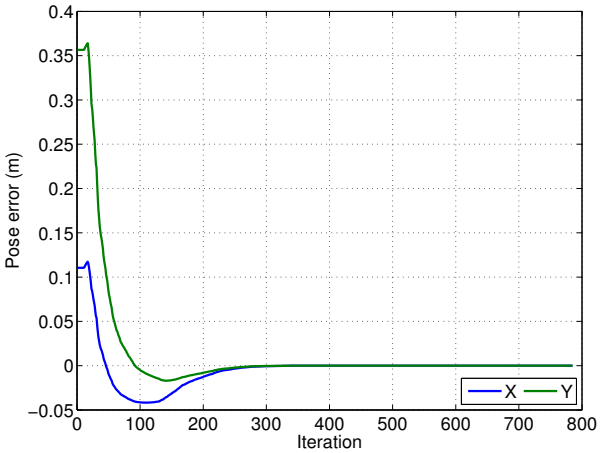
b - Posture task error



c - Left hand grasping task error



d - Right hand grasping task error



e - Visibility task error

Figure V.33 – OpenHRP simulation results of equilibrium, visibility and manipulation tasks

V.6.2.7 Conclusion and discussion

The simulation results show the successful and robust execution of the manipulation scenario with the HRP-2 robot using the multi-control points approach to control the redundant system and the directional projection method (SoT) to manage and apply the different simultaneous tasks. During the manipulation process, we find that we have fully exploited the redundancy of the robot having 30 DOF to execute several tasks (which constrained 22 DOF in total).

Note that in this case, we considered that the control points remain constrained during all the process to test the capacity of the multi control points approach and the stack of tasks technique to apply several simultaneous tasks. In other cases, we could remove these tasks/constraints after their execution, and benefit from the liberated DOF to apply additional tasks.

V.6.3 Manipulation with Nao Robot

In this part, we use the kinematic multi control points approach presented in section IV.1 to apply several service tasks with the Nao humanoid robot. System's architecture is detailed in paragraph V.1.1.3; and useful control points and frames are defined in paragraph V.2.2.

The envisaged service scenario is to integrate the real-time visual servoing techniques on a humanoid robot in closed loop, to perform different manipulation tasks. Real-time model-based tracking technique is used to apply 3D visual servoing tasks on the Nao robot. Several elementary tasks are used to perform this goal: head servoing for the visibility task, detection, tracking and manipulation of environment's objects.

V.6.3.1 Detection and tracking task

In this part, the MBT technique (section V.3.3) is used to track simple item models; it's initialized manually at the beginning of the application. Afterwards, the model is automatically detected and tracked; this tool allows us to instantly determine the pose of the desired item frame in the robot camera's frame (${}^{c}\mathbf{M}_{o}$).

V.6.3.2 Visibility task

Throughout this application, the visibility task is used for controlling the robot head's orientation to focus the item's center in the midpoint of the camera's image. Two DOF are used by this task to control the head's Yaw and Pitch. The task's goal is thus to regulate exponentially ($\dot{\mathbf{e}} = -\lambda\mathbf{e}$) the horizontal and vertical position of the center of the object projection $\mathbf{s}_{x,y} = {}^{c}\mathbf{T}_{o(x,y)}$ to zero ($\mathbf{s}^{*} = (0,0)$).

Using the object 3D pose ${}^{c}\mathbf{T}_{o} = (X, Y, Z)^{\top}$, and the 2D pose $(x, y)^{\top}$ of the tracked point (projection of 3D point in the normal image plane), we apply the control law defined in (IV.10)

using the visual primitive $\mathbf{s} = (x, y)$ and its corresponding interaction matrix $\mathbf{L_s}$ given by:

$$\mathbf{L_s} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+y^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+x^2 & -xy & -x \end{bmatrix} \tag{V.15}$$

Note that for this task the $\mathbf{J}$ matrix in (IV.10) uses the Jacobian of the robot's head control point calculated from the robot's geometric model (which is generated using the symbolic software SYMORO+ [KC+97]).

V.6.3.3 Pre-Grasping task

During this task, the robot's arm is supposed to move close to the item to grasp. According to Nao's gripper's geometry (of one DOF) and the item's shape (rectangular model), the number of constrained DOF and the pre-grasping position is predefined in the robot's manipulation parameters. This position is defined with respect to the object's position in Nao's frame ${}^{n}\left({}^{o}\mathbf{M}_{pg}\right)$.

In our case, regarding the gripper's and item's shapes, 4 DOF are enough to execute this task: 3 DOF constraints the gripper's pose and 1 DOF (Yaw angle) for the gripper's orientation. Furthermore, the pre-grasping distance is fixed to 5 cm, but we should not forget that the tracker gives the item's pose in real-time, thus the desired pre-grasping pose is calculated in closed loop.

The task's target is then to move the robot's arm to the pre-grasping pose. The task's error is extracted from the relative pose between the gripper and pre-grasping point (${}^{g}\mathbf{M}_{pg}$) which is regulated to zero. Note that ${}^{g}\mathbf{M}_{pg} = ({}^{n}\mathbf{M}_{g})^{-1}\,{}^{n}\mathbf{M}_{c}\,{}^{c}\mathbf{M}_{o}\,{}^{o}\mathbf{M}_{pg}$ where ${}^{n}\mathbf{M}_{g}$ and ${}^{n}\mathbf{M}_{c}$ are given by the robot's proprioceptive sensors.

Considering that the visual primitive is parameterized by $\mathbf{s} = (\mathbf{t}, \mathbf{u}\theta)$ where $\mathbf{t}$ is the position error between the current and desired frame, while $\mathbf{u}\theta$ is the orientation error, decomposed as the axis $\mathbf{u}$ and angle θ of the rotation between these two frames. The control law (IV.10) is then applied using the Jacobian at the robot's gripper, and the corresponding interaction matrix $\mathbf{L_s}$ given by:

$$\mathbf{L_s} = \begin{bmatrix} -\mathbf{I}_3 & [\mathbf{t}]_\times \\ \mathbf{0}_3 & -\mathbf{L}_\omega \end{bmatrix} \tag{V.16}$$

where $\mathbf{I}_3$ and $\mathbf{0}_3$ are the 3×3 identity and zero matrices respectively, the $\mathbf{L}_\omega$ matrix is given by $\mathbf{L}_\omega = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{sinc(\theta)}{sinc^2(\theta/2)}\right)[\mathbf{u}]_\times{}^2$, and $[\mathbf{t}]_\times$ is the skew symmetric matrix associated with vector $\mathbf{t}$.

V.6.3.4 Grasping task

Along this task, the robot's arm moves to grasp the desired item. The same number of DOF is constrained, and the same technique is used to define the grasping task as in the previous case of pre-grasping, but the desired gripper pose is changed to the object's pose. Thus, the task's error will be extracted from the relative pose between the gripper and the item ${}^{g}\mathbf{M}_{o}$ which is also regulated to zero (this pose is calculated using the relation: ${}^{g}\mathbf{M}_{o} = ({}^{n}\mathbf{M}_{g})^{-1}\,{}^{n}\mathbf{M}_{c}\,{}^{c}\mathbf{M}_{o}$).

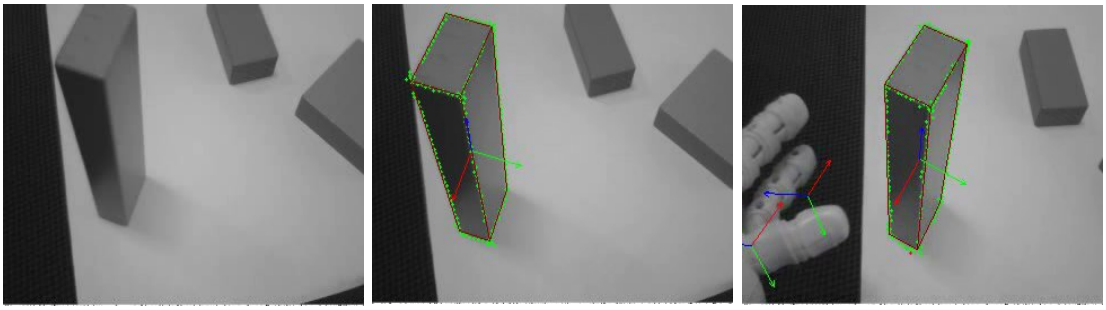
After arriving to the desired pose, the robot's gripper closes to catch the item. We should note that the visibility and pre-grasping tasks are executed in parallel, and once the pre-grasping finished, the grasping task is executed automatically.

Furthermore, a task is completed when the error norm reaches a predefined threshold value. This threshold varies with respect to the executed task: during the grasping task of small objects, a high precision is necessary unlike in the case of pre-grasping task. In our experiments, the threshold is predefined to 5 mm in the camera's image for the visibility task. In case of pre-grasping and grasping tasks, the precision is predefined to 5 mm and 1 mm respectively and 3 degrees for the orientation.

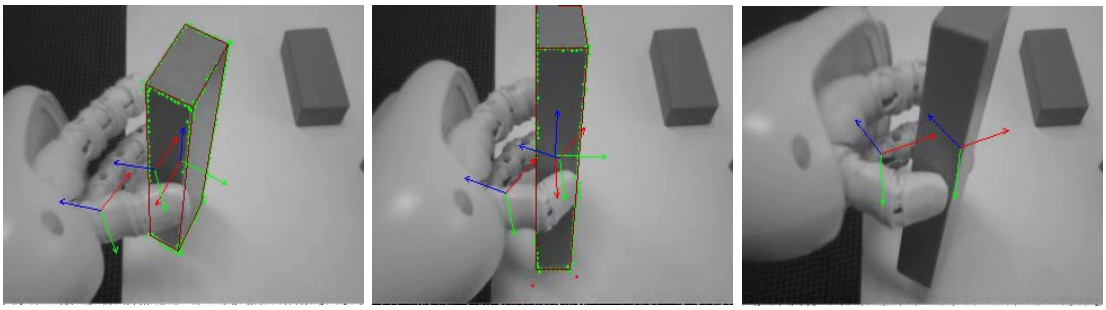
V.6.3.5 Experimental results

Experiment photos of visibility and grasping tasks executed by Nao robot are presented in Fig. V.34 and correspond to the tasks introduced in the above paragraphs Fig. V.34a, shows the item to grasp before launching the MBT to detect and track it (Fig. V.34b). The visibility task is used to center the object on the camera's image, and the pre-grasping task is executed in Fig. V.34c, where we can identify the different frames on the robot's arm and gripper in addition to the object's frame. Afterwards, the gripper's frame approaches the object's one when executing the grasping task (Fig. V.34d). Finally, the gripper closes and the manipulation task is completed (Fig. V.34e-V.34f).

The variation of the error in each task is presented in Fig. V.35: the first graph represents the horizontal and vertical position error during the head servoing task (visibility task), initially



a - Object to grasp b - Tracking initialization c - Tracking and grasping



d - Pre-grasp position e - Grasping object f - Successful object grasping

Figure V.34 – Photos of tracking and grasping tasks showing the arm's, gripper's and object's frames

the object is at a distance of approximately 30 cm from the center of the camera's image, we remark that this error is successfully regulated to zero during 29 sec with a precision of 5 mm.

For the second and third graphs, we represent the variation of pre-grasping and grasping tasks errors on X, Y and Z components (in Nao's frame), and the Yaw angle of the gripper orientation: the robot's hand is initially at an approximate distance of 25 cm from the predefined pre-grasping position, and the gripper is rotated of 180 deg with respect to the object. During this task, the position error and the Yaw angle are regulated exponentially to zero in 31 sec. Finally, we recall that these (pre) grasping tasks are successfully executed with a precision of 5 mm and 1 mm respectively and 3 deg for the orientation.

To ensure the robustness of the proposed visual servoing technique for manipulation tasks and the efficiency of the used control laws. The experiment on the previously presented tasks (tracking, head servoing and object grasping) has been repeated for several times with the same initial conditions in the next section where two error regulation strategies are proposed to improve the execution time of the manipulation process.

An improvement on the classical exponential error regulation strategy is discussed in the next section, where two propositions for faster task execution are presented and tested on the Nao robot in case of manipulation tasks.

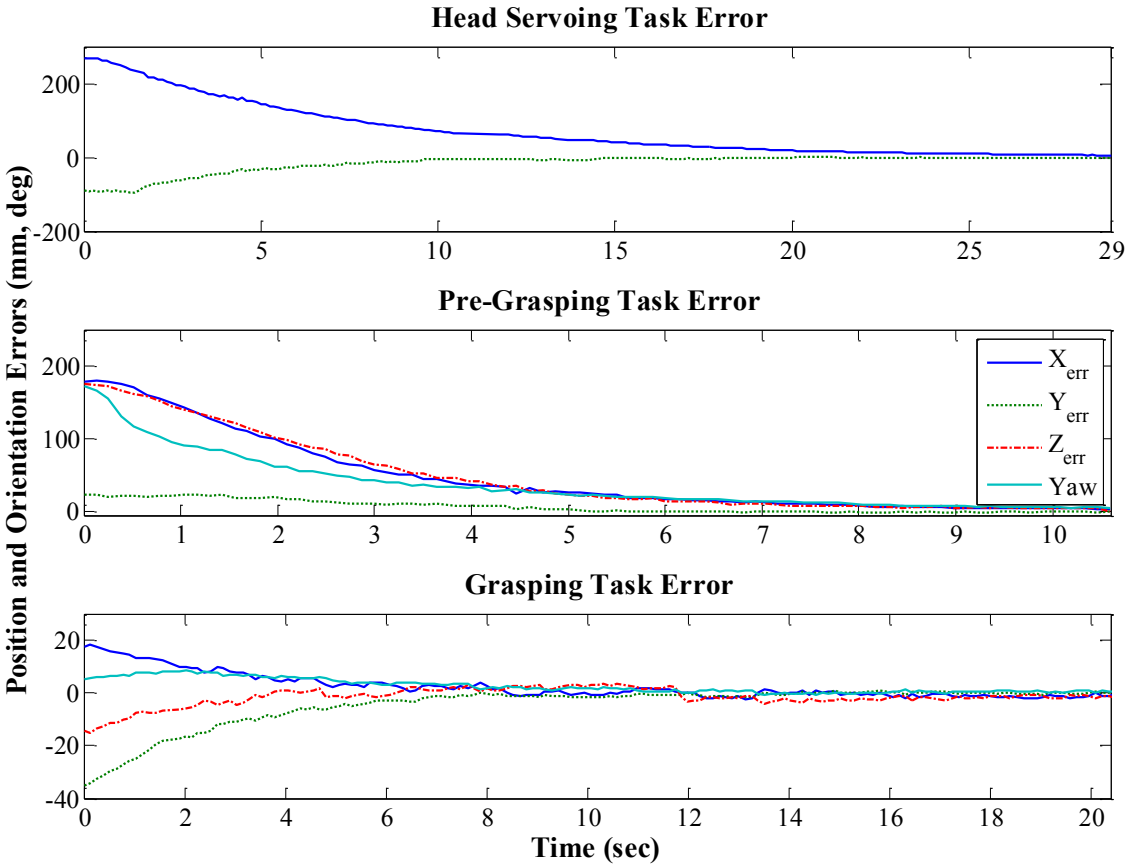


Figure V.35 – Experimental results: Visibility task, Pre-Grasping task and Grasping task errors

V.6.4 Error Regulation Strategies for Visual Servoing Tasks

In the previous section, the control law is designed so that the feature vector $\mathbf{s}$ reaches a desired value $\mathbf{s}^*$, using a special case of the error decrease (exponential regulation). In visual servoing, for a desired decrease $\dot{\mathbf{e}}$ of the error $\mathbf{e} = (\mathbf{s} - \mathbf{s}^*)$, the general control law used to define a task could be derived from (IV.8) and is given by:

$$\dot{\mathbf{q}} = \mathbf{J}_s^+ \dot{\mathbf{e}} \tag{V.17}$$

V.6.4.1 Classical exponential decrease

When executing service tasks, and especially object's manipulation, the main goal is to carry out these tasks as precisely and as quickly as possible. Thus to decrease the task error rapidly and to arrive to an acceptable precision without losing system's stability. Classically, an exponential decrease (V.18) is used to decrease the error when applying this control law:

$$\dot{\mathbf{e}} = -\lambda\, \mathbf{e} \tag{V.18}$$

with λ a proportional gain that is usually tuned to minimize the time to convergence.

In this case, the error follows an asymptotic exponential decrease to zero. However, this choice leads to a larger regulation time t_f (red curve in Fig. V.36). In addition, this gain should be tuned to reduce the convergence rate of the main task while preserving the stability of the system [MC10] and reducing error oscillations near convergence.

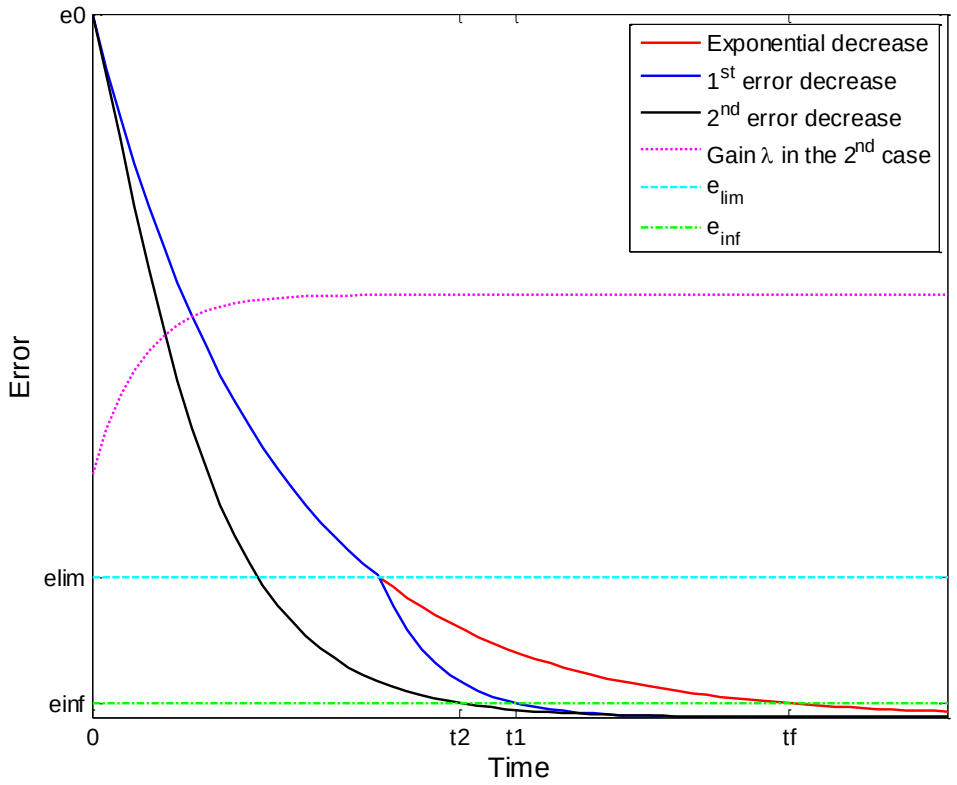


Figure V.36 – Comparison between the proposed strategies of error regulation

Moreover, when this classical regulation is applied into visual servoing tasks, and especially object's tracking, an increase of the gain λ may lead to tracking loss due to large camera velocities, or robot's joint velocity saturation. To avoid these issues, a small gain value is generally used to initialize the tracking and maintained constant till the end of the task. However, in the other hand, this consideration leads to an increase of the convergence time.

In fact, near task convergence, the displacement of the robot's camera and control points become smaller, thus a larger gain may be used to ameliorate task performance. Thus, a first proposal is to increase the value of λ after task initialization. Note that the gain used initially is optimal for the task initialization and cannot be increased from the beginning. Actually the use of a higher value leads to tracking crash and task failure due to the fast motions.

V.6.4.2 First proposition of error regulation

The goal is thus to decrease the convergence time of the control law to apply manipulations tasks rapidly without losing the system stability. Thus, we introduced a varying reduction of the error (V.19) which acts as an exponential decrease (with a gain λ_0) from the initial value ($\mathbf{e}_0$) until arriving to a specified threshold ($\mathbf{e}_{lim}$) near the equilibrium where it switches to a faster exponential decrease of the error (with a gain $\lambda_1 > \lambda_0$) to carry out the task as fast as possible and to arrive to the desired precision $\left(\mathbf{e}_{inf}\right)$ which indicates the task accomplishment (blue curve in Fig. V.36).

The decrease of the error is thus given by:

$$\dot{\mathbf{e}} = \begin{cases} -\lambda_0\, \mathbf{e} & for \quad \|\mathbf{e}\| \geq \mathbf{e_{lim}} \\ -\lambda_1\, \mathbf{e} & for \quad \|\mathbf{e}\| < \mathbf{e_{lim}} \end{cases} \tag{V.19}$$

As presented in Fig. V.36, this function is continuous and allows a decrease of the convergence time from t_f with the classical exponential decrease given in (V.18) to t_1 with this method.

The improvement in time owned to this method, in function of the initial (λ_0) and final (λ_1) exponential gains, initial values of error e_0, desired moment of switching e_{lim} and the desired precision e_{inf}, can be calculated theoretically by:

$$\frac{t_1}{t_f} = \frac{\lambda_0}{\lambda_1} + \left(1 - \frac{\lambda_0}{\lambda_1}\right) \frac{log\left(\frac{e_0}{e_{lim}}\right)}{log\left(\frac{e_0}{e_{inf}}\right)} \tag{V.20}$$

V.6.4.3 Second proposition of error regulation

To avoid discontinuity in error regulation velocity and possible task instability due to the switch from a low gain value to a higher one, another formulation of the error regulation can be used to have a smoother curve and to further decrease the time to convergence.

The formulation given by (V.21) uses the error norm to increase the gain value when the error decreases:

$$\dot{\mathbf{e}} = -\lambda\, \mathbf{e} \quad \text{with} \quad \lambda = \lambda_0 + a\, exp\left(-b\ \|\mathbf{e}\|\right) \tag{V.21}$$

where a, b and λ_0 are positive constant scalar values.

In contrast to the classical method where the gain value is constant during error regulation, in this case the value of λ begins with a small value, to ensure task initialization and stability, and then increases when the error norm arrives near zero. The gain variation for this proposition is illustrated by the magenta curve in Fig. V.36.

The representation of the corresponding error regulation (black curve) shows that the error variation function is continuous and decreases the convergence time from t_f with the classical exponential decrease to t_2.

V.6.4.4 Experimental results

To visualize the effect of the varying decrease of the error, presented previously, theses tasks are executed on parallel using the three propositions of error decrease given in (V.18), (V.19) and (V.21). The initial conditions and parameters used for each method are given in Table V.5.

Parameters	Visibility Task	Pre-Grasping Task	Grasping Task
$\mathbf{e}_0$	0.3 m	0.25 m 180 deg	0.005 m 15 deg
$\mathbf{e}_{inf}$	0.005 m	0.005 m 15 deg	0.001 m 3 deg
$\mathbf{e}_{lim}$	0.05 m	0.05 m	0.005 m
λ, λ_0	0.02	1.2	1.2

Table V.5 – Initial conditions and parameters values used in the error regulation strategies comparison

a- Exponential regulation of the error

First of all, we present the experimental results using the control law (V.17) and the classical exponential decrease of the error given by (V.18).

The regulation of the error in each task are already presented in Fig. V.35. The first graph represented the horizontal and vertical position error during the head servoing task (visibility task), initially the object is on a distance of approximately 300 mm from the center of the camera's image, we remark that this error is successfully regulated exponentially to zero during almost 29 sec with a precision of 5 mm.

For the second and third graphs, we represented the pre-grasping and grasping tasks errors on X, Y and Z components (in Nao's frame), and the Yaw angle of the gripper orientation. During these tasks, they are regulated exponentially to the desired precision during 31 sec.

b- First proposition of error regulation

In Fig. V.37a, we used the control law (V.17) with the first proposition of error regulation given by (V.19) with the same initial conditions and parameters (Table V.5). The switching between the two modes is executed when arriving to $\mathbf{e}_{lim}$.

Referring to the error regulation curves, the visibility task is executed in 21.7 sec and the grasping tasks in 18.6 sec with a 40 % time improvement with respect to the previous method. This result is a bit different from the theoretical one using the relation (V.20) which gives an improvement of 45 % due to experimental reasons.

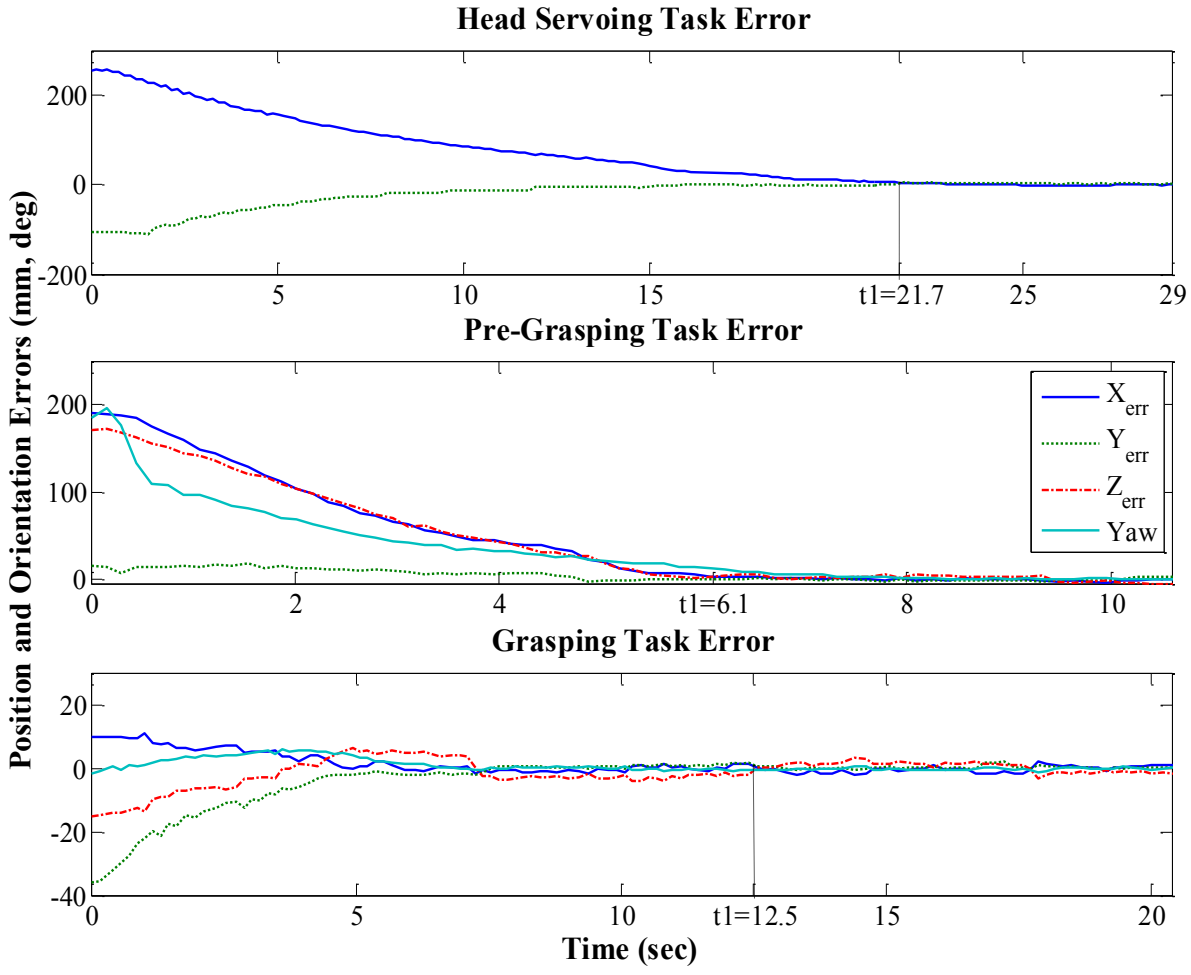


Figure V.37 – Experimental results in case of the first strategy of error regulation

c- Second proposition of error regulation

The control law (V.17) is used with the second proposition of error regulation given by (V.21). Referring to the variation of the error in each task, presented in Fig. V.38a, the visibility task is executed in 11.8 sec with a 59 % time improvement over the classical one. The pre-grasping and grasping tasks are executed in 15.1 sec with a 51 % time improvement.

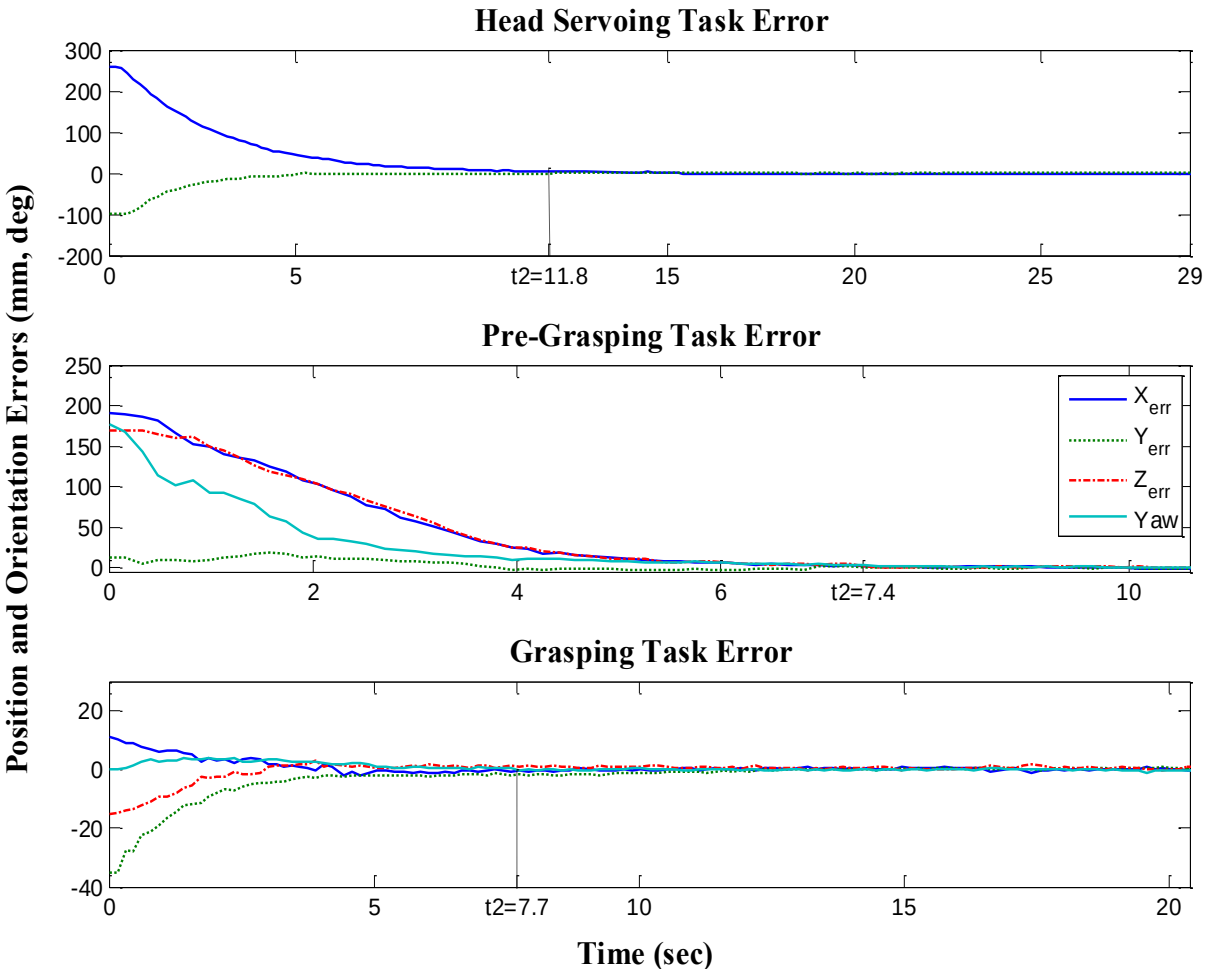


Figure V.38 – Experimental results in case of the second strategy of error regulation

V.6.4.5 Conclusion and Discussion on Manipulation Tasks

To ensure the feasibility and efficiency of the proposed error regulation strategies, and their improvement in convergence time, the previously presented experiments have been repeated 40 times (for each method) with the same initial conditions. We calculated the average of the required convergence time in each case and the improvement over classical one (Results are given in Table V.6).

Referring to the results of the three strategies and the represented graphs (Fig. V.37 and V.38), we remark that the manipulation task is executed faster with a good improvement of the convergence time between 40 % and 55 %. Furthermore, the system is stable after finishing the desired tasks and not perturbed by the defined varying gains.

By comparing the defined gains for each task, we remark that in the first proposition (V.19) the gain begins with a small value λ_0 then increases to λ_1, otherwise in this case (V.21) the gain begins with a relatively high value and increases all the time until stabilizing when arriving to a small error value (magenta curve in Fig. V.36).

Thus, the first method (V.19) is good for the visibility task because of the low gain at the beginning of the tracking but the switch to high gain may leads to some oscillations when convergence of the grasping task. Otherwise, a good behavior is remarked near convergence for the second proposition, but it begins with a relatively high gain which may influence on the tracking initialization due to high camera's motion, furthermore parameter tuning is very necessary for avoiding oscillations or instability.

Method	Visibility Task	Pre-Grasping Task	Grasping Task
Case (V.18)	26.3 sec	10.96 sec	17.41 sec
Case (V.19)	15.56 sec 40.1 %	5.43 sec 50.4 %	9.06 sec 48 %
Case (V.21)	11.74 sec 55.4 %	6.41 sec 41.5 %	8.26 sec 52.6 %

Table V.6 – Average convergence time (in seconds) and improvement (in percent) of the proposed methods over the classical method

V.7 Conclusion

In the previous chapter, we presented the multi-control points approach which controls the redundant systems at the task level, using generalized task to ensure the method's adaptability to various robots. In fact it consists on controlling the system/environment interactions using different types of embedded and external sensors.

In this chapter, the developed approach is validated into several platforms to perform multiple service tasks. The HRP-2 and Nao robots are used to perform (self) localization and manipulations tasks in indoor environments.

V.7.1 (Self) Localization Tasks

The localization issue is addressed using two different approaches: The first one consists on using only the embedded cameras to localize the robot, while walking, using a rough 3D model of a part of the environment (room's door in our case). On the other hand, the second approach uses external Kinect sensor to track a partial model of the robot (torso and head) using the cloud tracking technique.

Furthermore, additional modules are developed to improve the robustness and performance of these methods. In the first approach, an automatic re-initialization of the tracking is used to prevent localization task from failure, and for the second one, a veering compensation is performed to correct the veering behavior of the robot caused by backslash, friction, or motor power.

Experimental results show the successful execution of the desired tasks in the two cases. For the self localization case, results show that the robot can be successfully tracked with a precision of 6 cm for a walk of 1 m. Although the error may be relatively large with respect to other localization methods, but it is acceptable and sufficient when dealing with indoor locomotion for manipulation tasks. This method is used in the following publications [AMCM13c, AMCM13b].

To improve this method, an additional module could be used to dynamically compensate the camera's motions during robot's walking (using data from the robot's accelerometer/gyrometer), or by fusion with other sensors data.

Better results are obtained for the localization using external sensor with a significant improvement of the navigation method over other odometry-based techniques. In addition, the accomplished localization precision of less than 2 cm allows to successfully retrieve objects from the floor. This work leads to the following publications [AMCM13a, CAMM12].

The continuity of this work will focus on the integration of further environmental sensors in the system, and to cooperate with the robot's proprioceptive sensors. On the other hand, an additional control of the oscillatory motion of the robot during walking may largely improve the robot's behavior and the execution time of the desired task.

Further improvements are also possible: obstacle detection with the Kinect sensor would allow the robot to navigate robustly in a cluttered environment. The workspace of the robot

would be scalable by using more Kinect sensors in a networked system. Finally, multimodal user interaction can be achieved by gesture and voice recognition.

V.7.2 Manipulation Tasks

The manipulation task is applied in two different platforms: HRP-2 and Nao humanoid robots. For the first robot, the kinematic multi-control points approach and the directional projection method for redundancy resolution (defined in the previous chapters) are used to control the redundant system when applying different simultaneous tasks. During task execution, the robot is fully controlled to maintain its equilibrium and vertical position while performing the manipulations tasks using the two robot's hands. Simulation results show the successful and robust execution of the manipulation scenario with the HRP-2 robot using the developed approaches as presented in [AMMM10].

In the second part, a concrete scenario of Nao humanoid mobile robot executing manipulation tasks in an everyday life environment is presented. The experimental results point out the possibility and efficiency of using the MBT techniques to apply real-time 3D visual servoing on a simple humanoid robot. The results of tracking and grasping tasks, which are tested several times, show that this method is robust to camera occlusion by the robot's hand, and robust to slight object movement due to hand-object collision. This technique is used in the following publications [AMCM13c, AMCM13b, AMSCM11].

Furthermore, a faster execution of this task can be done using other error regulation, than the classical exponential one, to improve the task's convergence time. Therefore, two new strategies are developed to improve the task's performance. Experimental results show an improvement between 40% and 55% without losing the system's stability. This work is presented in [AMCM12].

Future works will concentrate on the extension of these regulation strategies from the autonomous object grasping to other service robotic tasks. Moreover, their implementation on other complex platforms with different objects should be accomplished to ensure the robustness of this method. For the experimental part, other sensor's feedbacks should be used to improve the manipulation reactivity against dynamic or unusual changes in the environment, especially after grasping the object where many occlusions and brutal motions appears.

These tasks should also be improved and tested on other robotic platforms. In fact, the mechanical and software architecture of Nao is very constraining, especially for the narrow camera field of view, the inability of simultaneous use of the two cameras and the constrained workspace of the hands.

Therefore, future works will focus on the improvement and implementation of these methods on other platforms with different objects of complex shapes with mobile or articulated elements. Other sensors' feedbacks can also be used to improve the manipulation robustness and reactivity against dynamic or unusual changes in the environment.

VI

General Conclusion

In this chapter, we present the conclusions and contributions of this work. The main objective is to study the redundancy of a general robotic platform, and to find the best technique which allow the control law to benefit from such redundancies to apply several simultaneous and prioritized tasks while taking into account the system's constraints.

The studied problems and the developed techniques are validated using several platforms (planar, LWR, Nao, HRP-2), in simulation and in real-time. They are ado used to apply several assistive and service tasks such as localization and manipulation.

VI.1 Conclusions and Contributions

The research described in this thesis is concerned with two main issues in the control of redundant robots. The first issue, presented in Chapters II and III, concerned the redundancy resolution and task sequencing techniques, while the second issue, presented in Chapters IV and V, concerned the real-time application of several service tasks on humanoid robots.

VI.1.1 Redundancy Resolution and Task Sequencing

VI.1.1.1 Redundancy Classification

In addition to the robot's inherited redundancy which arise from the original mechanical design, other types of redundancies occur due to the interaction between the robot and his surrounding (especially in manipulation and grasping tasks), or due to the architecture of the environment's objects that may include extrinsic and intrinsic degrees of freedom (such as the articulated objects). Furthermore, additional redundancies arise from the sensory tools, they are related to the presence of more sensing or measuring elements than theoretically necessary, they are usually used when high reliability is necessary or due to limitations in the sensor's workspace.

Therefore, based on the diversity in redundant systems and on the variety in redundancy interpretations, the first contribution is the development of an extensive classification of the

various types of redundancies which may be present in a general robotic platform. This organization is illustrated by the definition of several redundancy indices with relevant examples from various robotic domains.

VI.1.1.2 Classification of Redundancy Resolution Methods

Afterwards, various methods for redundancy resolution and task sequencing are discussed and presented in a unified notation to simplify the comparison. The schematic diagram in Fig. VI.1 summarizes the different classes of control methods that are used to solve system's redundancy at the kinematic level, and presents some examples of the applied tasks and constraints.

Based on this classification, we notice that most of the existing redundancy resolution and task sequencing techniques are constructed either by gathering the desired tasks in a unified set and using a static/dynamic weighting to manage their priority, or by stacking these tasks in a continuous formalism for sequencing/switching between the different tasks. Furthermore, an extension of the stability condition of the control law leads to an increase in the number of available DOF and thus an improvement in the performance of the system .

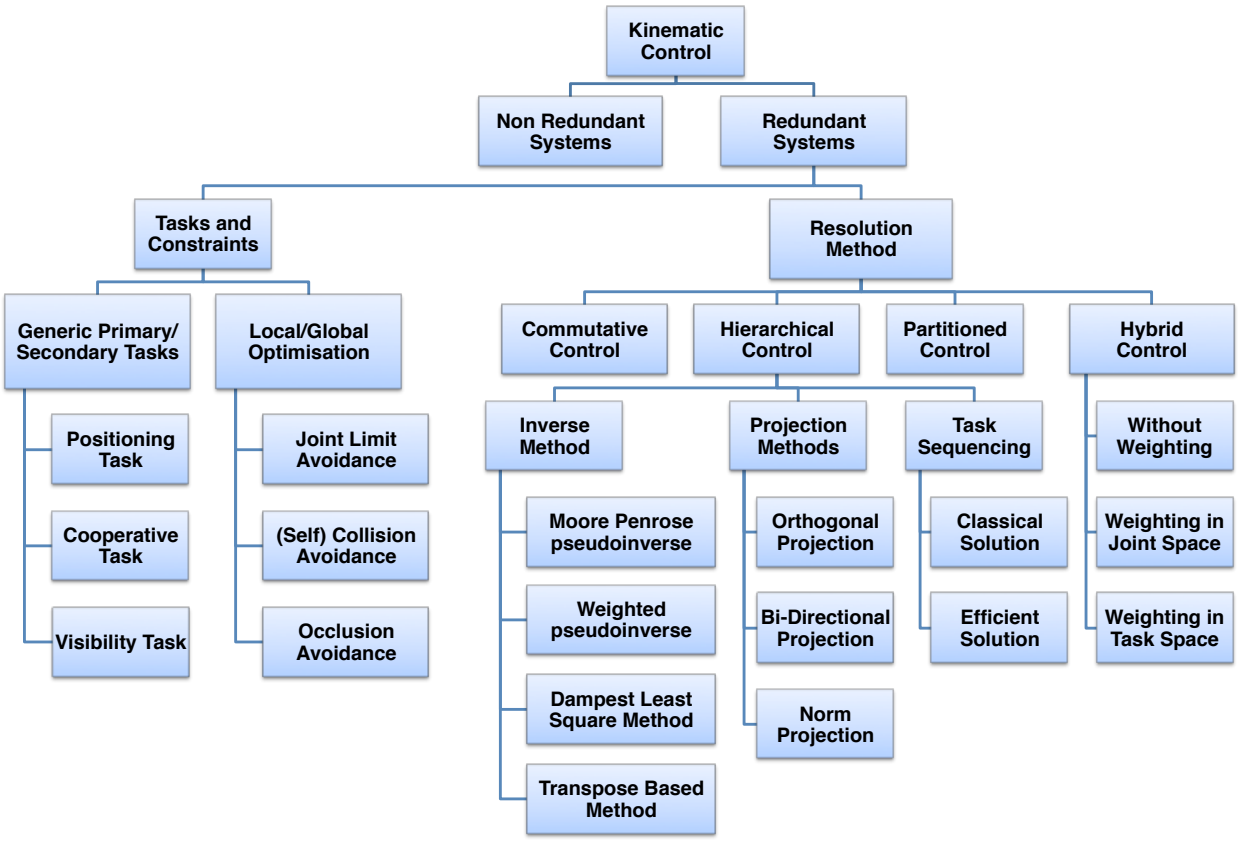


Figure VI.1 – General representation of the kinematic redundancy resolution methods

VI.1.1.3 Comparison of Kinematic Redundancy Resolution Techniques

To execute several simultaneous tasks with a specific hierarchy and priority, a comparison between the appropriate redundancy resolution techniques is carried out, using simulation on different types of redundant planar robots and the 7-degrees-of-freedom Kuka robot.

Various performance criteria are defined and multiple task configurations are applied to show the advantages and drawbacks of each control law and the possible applications where it can be suitably applied. Therefore, using this comparison results, the most useful and optimal control solution is appropriated depending on the number of system's and task's degrees of freedom, in addition to the type of the applied tasks and constraints.

VI.1.1.4 New Redundancy Resolution Techniques

To benefit from the advantages of the orthogonal and directional projection methods in the different cases, a new projection operator is defined to control the contribution of the motions produced by the secondary task in the control law. In addition, it helps the main task to be completed faster by controlling the projection's direction of the additional task. Moreover, this projector uses a smooth transition function to pass between the directional projection (where all secondary task components are considered) and the classical orthogonal one near convergence to perform the exact regulation of the secondary task.

A problem in the directional projection method, and consequently in the defined projector, is noticed at the joint velocities level: a discontinuity appears in several cases due to the discontinuity of the space where the tasks errors are projected (space of singular values). This discontinuity leads to oscillations of the projector when passing between the different conditions in the control law definition, and thus vibration of the system.

Accordingly, a new simple and generalized projector is defined by considering the stability condition on the overall error vector without projection of its components into that space. Therefore, it allows a decrease in the number of parameters which should be tuned, and a perfect execution of the desired tasks in an acceptable time.

VI.1.2 Service Task Application on Humanoid Robots

VI.1.2.1 Kinematic/dynamic multi control points approach

To control a general multi-arm redundant system, we developed the kinematic and dynamic multi control points approach which consists on studying and monitoring the robot/ environment interaction on several points of the system.

This general framework uses the previously discussed redundancy resolution techniques to perform the desired scenario which is decomposed in a set of generalized generic tasks (to ensure the method's adaptability to different robotic platforms). Various types of embedded and external sensors (especially vision ones) are used to perform a precise and robust sensor-based control.

The developed approach is validated on several platforms to perform multiple service

tasks. Thus, the presented generic tasks and their corresponding control laws are applied in localization, navigation and manipulation tasks on HRP-2 and Nao robots using several embedded and external sensors.

VI.1.2.2 Localization and navigation on Nao robot

The localization issue is addressed using two different approaches: the first one consists on using only the embedded cameras to localize the robot, while walking, using a rough 3D model of a part of the environment; and the second approach uses external Kinect sensor to track a partial model of the robot using the cloud tracking technique.

Furthermore, additional modules are developed to improve the robustness and performance of these methods. For the first approach, an automatic re-initialization of the tracking is used to prevent localization task from failure, and for the second one, a veering compensation is performed to correct the veering behavior of the robot caused by backslash, friction, or motor power.

Experimental results show the successful execution of the desired tasks in the two cases: for the self localization case, results show that the robot can be successfully tracked while walking; and better results are obtained for the localization using external sensor with a significant improvement of the navigation method over other odometry-based techniques. Moreover, the precision of the latter localization method allows to successfully retrieve objects from the floor.

VI.1.2.3 Manipulation tasks on HRP-2 and Nao robots

The manipulation task is applied on two different platforms: HRP-2 and Nao humanoid robots. For the first system, the kinematic multi-control points approach and the directional projection method for redundancy resolution are used to control this redundant system when applying different simultaneous tasks. In fact, during task execution, the robot is fully controlled to maintain its equilibrium and vertical position while performing the manipulations tasks using its two hands. Simulation results show the successful and robust execution of the manipulation scenario with the HRP-2 robot using the developed approaches.

In the second part, the kinematic multi-control points approach is also used to perform a concrete scenario of Nao humanoid mobile robot executing manipulation tasks in an everyday life environment. The experimental results point out the possibility and efficiency of using the MBT techniques to apply real-time 3D visual servoing on a simple humanoid robot.

This method is robust to camera occlusion by the robot's hand, and robust to slight object movement due to hand-object collision. Furthermore, a faster execution of this task can be done using other error regulation, than the classical exponential one, to improve the task's convergence time.

VI.2 Limitations and Future Work

VI.2.1 Redundancy Resolution and Task Sequencing

The theoretical and simulation studies of the different methods for redundancy resolution and task sequencing will facilitate the future development of new techniques to improve the performance of the existing ones or to implement new solutions of the redundancy problem.

In fact, the development of new techniques can be achieved, for example, by defining new behavior of the executed tasks, extending the stability condition to increase the DOF of the system, or using other inverse techniques than the classical pseudo-inverse one.

Several existing methods consist on the execution of the lower priority tasks while not perturbing the higher priority one. However, in case of equal priorities between several constraints or tasks, a less restrictive solution could be found by projecting the secondary task into the null space of multiple tasks simultaneously. Such procedure allows a minimization of the error norm of several tasks, and not only the main one, while performing the secondary tasks.

Other alternatives for redundancy resolution control are based on replacing the classical Moore-Penrose pseudo-inverse by other generalized inverse techniques using, for example, the dampest-least square inverse, matrix transpose, or other weighted pseudo-inverse techniques. Although these methods may be simpler and computationally cheaper than the classical pseudo-inverse, their use in redundancy resolution and task sequencing may lead to imprecise execution of the desired tasks.

On the other hand, the possibility to define other projection operators in the hierarchical control is based on the use of other Lyapunov functions to prove the system's stability. In fact, in addition to the classical function based on error norm, other functions could be defined to optimize different parameters, but in all cases a continuous projection space should be defined to prevent the system's oscillations.

Finally, for the developed projection operators in the third chapter, in spite of the acceptable behavior in simulation on the planar robots; their efficiency should be validated on real robotic platforms for significant tasks execution.

VI.2.2 Application of Multi-Control Points Approach

The different results of this dissertation will be directly used in the control and experimental parts of ARMS project [ARM] which studies the robotization of bovine muscle separation in meat cutting and transformation processes. A multi-arm robotic system is considered to achieve simultaneously four main mechanical actions (handling, pulling, pushing and/or cutting), on three different kinds of meat constituting objects: rigid (such as bones), rigid-articulated (such as knee-joints) and deformable (such as meat muscles). Furthermore, several sensors and active perception system are integrated to extract relevant information in real time from the muscle and the robotic tasks environment.

The classification of the different types of redundancies in the robotic platform, and the identification of the applied tasks and system's constraints, allow for choosing the appropriate

redundancy resolution method to be considered in the multi control points approach which is used to control this platform at the kinematic and dynamic levels.

In the presented works, the multi control points approach is implemented on several platforms to only execute various tasks at the kinematic level. In fact, the used platforms can not be controlled at the torque level and thus did not allow the use of the dynamic version of this approach. Therefore, the platform of ARMS project will allow the implementation and improvement of the dynamic approach, in addition to the direct application of the desired scenario at the kinematic level.

On the other hand, for the performed localization and manipulation tasks on Nao and HRP-2 robots, an additional control of the oscillatory motion of the robot during walking may largely improve the robot's behavior and the execution time of the desired task. Furthermore, an additional module could be used to dynamically compensate the camera's motions during robot's walking, to integrate further environmental sensors in the system, and to cooperate with the robot's proprioceptive sensors.

Other sensors' feedbacks could be used to improve the manipulation reactivity against dynamic or unusual changes in the environment, especially after grasping the object where many occlusions and brutal motions appears. The limited workspace of the robot would be also scalable by using more vision sensors in a networked system.

A

Resumé Etendu

Contents

A.1 Motivations et Objectives

La redondance est un large concept qui n'est pas limité à la robotique. Il peut être littéralement défini par "*l'état de ne plus être nécessaire ou utile*", ou d'une perspective d'ingénierie par "*l'inclusion des composants supplémentaires qui ne sont pas strictement nécessaires pour fonctionner*".

Par conséquent, tous les systèmes robotiques, du simple bras manipulateur au système multi-bras en passant par les robots humanoïdes, présentent plusieurs types de redondances qui peuvent apparaître non seulement dans leur structure mécanique, mais aussi dans leur interaction avec les objets qui les entourent dans l'environnement. De plus, l'utilisation excessive des capteurs externes et internes amène à une redondance dans les données mesurées. Cependant, une telle redondance peut être utile lorsque le système nécessite une haute fiabilité ou lorsque l'espace de travail des capteurs est très limité.

Un ou plusieurs types des redondances mentionnées peuvent exister simultanément lors de la commande des plates-formes robotiques. En fait, leur présence dépend des composants utilisés (robots, objets, capteurs, ...), de l'application qui définit les tâches élémentaires appliquées, ainsi que du niveau désiré de la précision et de la robustesse de la loi de commande.

D'autre part, lors de l'exécution d'une tâche sur tels systèmes redondants, il existe une infinité de solutions possibles. Ainsi, le choix de la solution la plus pertinente donne lieu à la définition de nouvelles lois de commande pour gérer les mouvements internes du système en appliquant des tâches secondaires, en ajoutant certaines contraintes sur le comportement du système, ou en optimisant (localement ou globalement) certaines performances désirées.

En ce qui concerne les contraintes, deux types peuvent être identifiées: d'une part, la contrainte d'égalité classique qui, par exemple, impose une configuration donnée du système, et d'autre part, la contrainte unilatérale qui est utilisée pour limiter les valeurs de la vitesse et accélération articulaires du robot par exemple, ou pour éviter la collision avec des obstacles dans l'environnement ou l'auto-collision entre les parties du robot.

A.1.1 Motivations et Problèmes

Malgré leur présence dans tous les systèmes robotiques et dans toutes les applications, les différents types de redondances sont rarement pris en compte dans le schéma de commande du système. Néanmoins, la diversité de ces types de redondances conduit au développement de plusieurs méthodes de résolution de la redondance. Ces techniques permettent de choisir la meilleure solution parmi l'infinité de possibilités et de gérer les priorités et les interactions entre la tâche principale et les tâches secondaires ou les contraintes.

Cependant, depuis les travaux de Liegeois [Lie77], la plupart des approches publiées sont basées sur l'optimisation locale avec une résolution cinématique de la redondance, en utilisant la méthode classique de projection orthogonale dans l'espace nulle de la tâche principale. Malgré la simplicité de ce schéma, son inconvénient est dû au caractère local du processus d'optimisation qui peut conduire à des résultats insatisfaisantes sur les longues tâches. Par ailleurs, cette technique utilise une condition de stabilité très restrictive qui ne permet pas au système de profiter complètement de la redondance disponible.

Plus récemment, plusieurs approches cherchent à améliorer la performance du système en étendant son espace libre pour appliquer des tâches secondaires. Cependant, l'absence d'une comparaison claire entre les performances de ces méthodes, en plus de la manque d'étude sur leurs comportements dans plusieurs cas particuliers et critiques, limite leur utilisation à quelques applications.

Par conséquent, selon le système commandé, les tâches appliquées, et les types de redondance existants, le choix de la méthode la plus pertinente pour la résolution de la redondance doit être fait. En effet, des critères de performance explicites doivent être définis et pris en compte lors de ce choix. En outre, lorsque le robot ne peut pas appliquer toutes les tâches/contraintes désirées, le système doit rester stable et un comportement acceptable doit être défini, par exemple les tâches et les contraintes doivent être entièrement ou partiellement exécutées, ou le système doit être bloqué et aucune des tâches est appliquée.

A.1.2 Objectifs

Le but de cette thèse est de ne plus considérer la redondance comme un problème à résoudre ou cas à éviter, mais plutôt d'apprécier la redondance comme un avantage dont on doit en bénéficier pour améliorer le comportement et la performance du système.

Dans cette étude, on s'intéresse alors à l'étude du comportement d'un système redondant lors de l'exécution de plusieurs tâches simultanées avec une hiérarchie et priorité spécifique. Par conséquent, on procède à une comparaison entre les différentes techniques existantes pour la résolution de la redondance afin de mieux identifier leurs avantages et leurs inconvénients et de déterminer les applications appropriées pour chacune. Pour contrôler un système robotisé général, une approche cinématique/dynamique est élaborée et validée par l'application de plusieurs tâches de service et d'assistance sur des robots humanoïdes (HRP-2 et Nao) et sur une plateforme multi-bras.

Dans le **Chapitre II**, on adresse la définition et la classification des différents types de redondances qui peuvent être présents dans une plateforme robotique générale. En outre, les différentes lois de commande pour la résolution de la redondance et l'enchaînement des tâches sont présentées avec une notation unifiée (avec plus de détails sur les techniques de commande hiérarchique).

Après avoir classé les différentes techniques, on choisit, dans le **Chapitre III**, celles qui sont appropriées pour réaliser les objectifs de cette étude, et on les compare en fonction de plusieurs critères de performance. Cette comparaison est effectuée par simulation sur différents types de robots planaires et sur le robot LWR Kuka à 7 degrés de liberté (ddl). En ce qui concerne les tâches appliquées, des tâches de positionnement sont exécutées sur les deux types de robots comme tâches principales et secondaires. En outre, des contraintes d'égalité et d'inégalité sont étudiées: le premier type peut être considéré, par définition, comme équivalent à une tâche générique, et le second correspond à une restriction ou un obstacle dans l'espace de travail du robot.

Pour comprendre plus profondément les techniques d'enchaînement et d'exécution si-

multanée de plusieurs tâches, on considère plusieurs cas d'études où le système robotique peut ou ne peut pas exécuter toutes les tâches désirées. Dans ce dernier cas, la meilleure performance du robot consiste à maintenir la stabilité du système, l'exécution de la tâche principale et l'exécution au mieux des tâches/contraintes secondaires selon la priorité spécifiée.

Compte tenu des résultats de cette comparaison, on peut choisir la solution la plus pertinente et optimale de la loi de commande. Elle permet d'appliquer les tâches désirées sur le système redondant en fonction du nombre de ddl de la tâche et du système, et de la nature des tâches et des contraintes appliquées. Enfin, l'amélioration des techniques existantes sera également abordée pour surmonter les inconvénients de ces méthodes et pour résoudre les problèmes rencontrés lors de la simulation. En fait, la comparaison théorique et l'étude des performances des différentes méthodes conduira à la définition de nouvelles méthodes généralisées et simples pour la résolution de la redondance.

Ensuite, après avoir adresser les techniques de résolution de la redondance, on développe, dans le **Chapitre IV**, l'approche cinématique et dynamique de multi-points de contrôle qui est un formalisme général pour contrôler les systèmes redondants. Contrairement aux approches classiques de commande dans l'espace cartésienne et articulaire, ce formalisme est basé sur une technique de commande référencée multi-capteurs qui consiste uniquement à étudier et contrôler l'interaction entre le robot et l'environnement dans plusieurs points du système.

Lors de l'exécution des tâches de service et d'assistance sur les robots redondants, notamment les humanoïdes, les mêmes tâches sont toujours exécutées. Ainsi, on définit les tâches les plus récurrentes sous une forme générique pour maintenir la capacité d'adapter l'approche de commande présentée à tout système robotisé.

Pour réaliser une commande référencée capteur précise et robuste, on utilise divers types de systèmes de vision qui sont déjà intégrés dans le robot ou fixés dans l'environnement. Ils sont utilisés pour exécuter plusieurs applications de la robotique de service qui sont décomposées en plusieurs tâches élémentaires génériques. En outre, les techniques de résolution de la redondance, précédemment présentées, sont utilisées pour commander le système robotisé lors de l'exécution de ces tâches en tenant compte de leurs priorités relatives.

Enfin, ce formalisme est utilisé dans le **Chapitre V** pour effectuer plusieurs applications sur trois plateformes différentes: un système multi-bras pour la séparation des muscles et la découpe de la viande, et deux robots humanoïdes HRP-2 (en utilisant le simulateur OpenHRP) et Nao (en temps réel). Plusieurs techniques d'asservissement visuel sont utilisées pour suivre le robot et les objets de l'environnement et pour effectuer la localisation, la navigation, le suivi d'objets, et les tâches de saisie.

Pour ce résumé étendu, la même structure est adoptée avec 4 sections qui correspondent aux 4 chapitres de la thèse.

A.2 Résolution de la Redondance Cinématique

A.2.1 Classification des redondances

Cette partie présente une classification globale des différents types de redondances qui peuvent être présents dans n'importe quel système robotique ou qui apparaissent durant l'exécution des tâches. Cette classification prend en compte alors l'architecture du système et son interaction avec l'environnement.

En fait, la redondance est étudiée à trois niveaux: le système robotique, l'interaction robot-environnement et les capteurs utilisés. Pour faire cette classification, on considère:

— 'n' le degré de mobilité du robot
— 'm' le degré de mobilité de l'organe terminal
— 't' la dimension de la tâche
— 'N_a' le nombre d'articulations motorisées du robot

Ainsi, on peut identifier les cas suivants:

Cas de la Non-Redondance: $n = m$

Dans ce cas, le robot a le même degré de mobilité que de degré de mobilité de son organe terminal. C'est le cas des robots **non-redondant**.

On note que dans le cas où il y a une diminution de la dimension 'm' dans des configurations spécifiques ('n'>'m'), le robot est considéré dans une configuration singulière ou dégénérée.

Cas de la Redondance Cinématique: $n > m$

"Un manipulateur est intrinsèquement redondant lorsque la dimension de l'espace opérationnelle est inférieure à la dimension de l'espace articulaire".

Par conséquent, lorsque 'n' est conçu pour être supérieur à 'm', le système est **cinématiquement redondant**. Dans ce cas, le modèle géométrique inverse a un nombre infini de solutions. De plus, dans telles situations, la forme du robot peut être modifiée sans changer la configuration de l'effecteur, ce mouvement est connu par: mouvement dans l'espace nul ou mouvement interne.

Cas de la Redondance de la Tâche: $n > t$

Lorsque la dimension de la tâche 't' est plus petite que le degré de mobilité du robot, on a alors une **redondance de tâche** (ou de la redondance fonctionnelle).

Cette redondance dans l'interaction entre le robot et son environnement apparaît surtout dans la définition de la tâche appliquée qui peut être réalisée par différentes stratégies. Ainsi, la redondance de la tâche n'est pas définie seulement par l'architecture du système, ni par la forme de l'objet seul, elle est plutôt caractérisée par l'interaction entre eux, donc, elle change en fonction des paramètres donnés par l'utilisateur et par la tâche désirée.

Cas de la Redondance d'Actionnement: $N_a > n$

Lorsque le nombre d'articulations motorisées est plus grand que le degré de mobilité du robot, donc une **redondance d'actionnement** est présente.

Par exemple, ce type de redondance se produit lorsque deux ou plusieurs manipulateurs saisissent un objet commun, les manipulateurs et l'objet forment une chaîne cinématique fermée de la même façon qu'un manipulateur parallèle. Mais à la différence des mécanismes parallèles, chaque bras est composé d'un manipulateur sériel, donc entièrement actionné. Par conséquent, le système est composé de plus d'actionneurs que de degrés de mobilité de l'objet.

Cas de la Hyper Redondance : $n >> m$

Il existe plusieurs systèmes qui sont classés comme très ou **hyper redondant** [MSK96]. De tels systèmes ont une dimension de l'espace articulaire qui est beaucoup plus grande que la dimension de l'espace de l'effecteur, c'est à dire '$n >> m$'.

Cas de la Redondance Métrologique:

Il n'y a pas de définition claire et exhaustive de la **redondance métrologique** qui est applicable à tous types de structures et de capteurs. Mais, en général, une telle redondance est liée à la présence de plus d'éléments de mesure que théoriquement nécessaire, elle est généralement utilisée lorsque une haute fiabilité est nécessaire, en raison de limites dans l'espace de travail du capteur [CKP+10] ou, comme dans le cas de robots parallèles où on motorise des articulations passives pour connaitre la configuration du robot.

A.2.2 Résolution de la redondance cinématique

La présence de la redondance dans un système robotique amène à un haut niveau de dextérité qui peut être utilisé pour optimiser de nombreux critères de performance et pour effectuer des tâches secondaires. Afin d'atteindre cet objectif, les différents types de redondance doivent d'abord être identifiées et classées, puis une stratégie appropriée doit être élaborée pour profiter de leur présence dans le système par l'application de plusieurs tâches simultanées (tout en respectant la priorité et la compatibilité entre eux).

A cause de la complexité des robots redondants, plusieurs techniques ont été développées pour contrôler de tels systèmes en utilisant une variété de lois de commande. En fonction de la stratégie de commande utilisée, les différentes méthodes peuvent être groupées en quatre catégories principales:

— **La commande partitionnée** est utilisée pour appliquer des tâches indépendantes (sans couplage dans l'espace articulaire).

— **La commande commutative** utilise une loi de commande spécifique pour chaque cas du système. La complexité de cette méthode réside dans la définition de l'instant et la condition de passage entre les tâches.

— **La commande hybride**: consiste à introduire des tâches/contraintes secondaires dans le problème de cinématique inverse en utilisant la matrice Jacobienne étendue, avec des matrices de pondération dans l'espace articulaire ou dans l'espace de la tâche.

— **La commande hiérarchique**: consiste à exécuter des tâches/contraintes secondaires (en utilisant des matrices de projections) sans perturber la régulation de la tâche principale.

A.2.3 Commande hiérarchique

Cette partie aborde la commande hiérarchique, elle présente et compare trois méthodes différentes: la projection orthogonale, la projection directionnelle et la solution de norme minimale.

A.2.3.1 Projection orthogonale

La méthode classique de projection orthogonale ($\mathbf{P_e}$) projette le vecteur représentant la tâche secondaire $\mathbf{z}$ dans l'espace nul (le noyau) de la jacobienne $\mathbf{J}$ de la tâche principale ($\mathbf{e}$) en utilisant la loi de commande suivante:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}} + (\mathbf{I_n} - \mathbf{J}^{+}\mathbf{J})\,\mathbf{z} \tag{A.1}$$

avec $\mathbf{P_e} = (\mathbf{I_n} - \mathbf{J}^{+}\mathbf{J})$ le projecteur orthogonal dans l'espace nul de $\mathbf{J}$, pour que $\mathbf{z}$ soit réaliser au mieux sans perturber l'exécution de la tâche principale $\mathbf{e}$.

Ainsi, les tâches/contraintes secondaires n'ont pas d'effet sur la convergence de la tâche principale dans l'espace cartésien. Pour une régulation exponentielle de l'erreur de la tâche principale, la stabilité du système peut être vérifiée à l'aide de la fonction de Lyapunov classique basée sur la norme d'erreur $\left(\mathcal{L}(\mathbf{e}) = \frac{1}{2}\|\mathbf{e}\|^2 = \frac{1}{2}\mathbf{e}^{\top}\mathbf{e}\right)$.

La principale limitation de cette méthode est que seul les ddl qui ne sont pas contrôlés par la tâche principale peuvent être exploités pour exécuter d'autres contraintes. Lorsque la tâche principale est plus compliquée, elle utilise plus de dll, et alors il est plus difficile d'appliquer les contraintes secondaires. De plus, si la tâche principale utilise tous les ddl du robot, aucune contrainte secondaire ne peut être prise en compte.

A.2.3.2 Projection directionnelle

La méthode de projection directionnelle ($\mathbf{P_z}$) consiste à ne pas interdire les mouvements de la tâche secondaire qui accélèrent l'exécution de la tâche principale. En fait, cette méthode maintien la stabilité du système en empêchant la loi de commande secondaire d'augmenter l'erreur de la tâche principale. De plus, cela améliore les performances de la tâche secondaire en augmentant le nombre de ddl disponibles.

Ces conditions peuvent être explicitées par une décroissance plus rapide de la fonction de Lyapunov que dans le cas de la projection orthogonale ($\dot{\mathcal{L}}_{(\mathbf{P_z})} < \dot{\mathcal{L}}_{(\mathbf{P_e})}$).

Contrairement à la première méthode qui projette toutes les tâches secondaires dans le noyau de la tâche principale, cette approche distingue deux cas: si la tâche secondaire va dans

le même sens que la tâche principale, elle est maintenue tant qu'elle n'augmente pas l'erreur de la tâche principale et vérifie la condition de stabilité, sinon, si la tâche secondaire va dans la direction opposée à celle de la tâche principale, elle est projetée dans l'espace nul de la tâche principale.

En utilisant la décomposition en valeurs singulières SVD de la Jacobienne ($\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$) et la fonction de Lyapunov classique, l'opérateur de projection directionnelle est établi en considérant les deux cas ci-dessus dans l'espace des valeurs singulières.

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}} + \mathbf{P}_{\mathbf{z}}\,\mathbf{z} \tag{A.2}$$

$$\text{avec } \mathbf{P}_{\mathbf{z}} = \mathbf{V}\,\mathbf{P}_{(\tilde{\mathbf{z}})}\mathbf{V}^\top \;,\; \mathbf{P}_{(\tilde{\mathbf{z}})} = \begin{bmatrix} p_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & p_n(\tilde{z}) \end{bmatrix} \;,\; p_i(\tilde{\mathbf{z}}) = \begin{cases} 1 & \text{if} \quad i > m \;\text{ or }\; \tilde{z}_i = 0 \\ 1 & \text{if} \quad \tilde{z}_i\,\tilde{l}_i\,\sigma_i < 0 \\ 0 & \text{if} \quad \tilde{z}_i\,\tilde{l}_i\,\sigma_i > 0 \end{cases}$$

$$m = rang(\mathbf{J}) \;,\; \tilde{\mathbf{z}} = \mathbf{V}\mathbf{z} \;,\; \mathbf{\Sigma} = diag(\sigma_i) \;,\; \widetilde{\nabla\mathcal{L}} = \mathbf{U}^\top\nabla\mathcal{L} \;,\; \text{et } \widetilde{\nabla\mathcal{L}} = (\tilde{l}_1, ..., \tilde{l}_m).$$

Cependant, même si cet opérateur de projection améliore les performances du système, le nombre de degrés de liberté peut être insuffisant pour respecter toutes les contraintes.

A.2.3.3 Solution de norme minimale

Au lieu de réduire la valeur des composantes de l'erreur ($\dot{\mathbf{e}} = -\lambda\,\mathbf{e}$), la méthode de projection basée sur la norme minimale considère une décroissance exponentielle de la norme de l'erreur ($\dot{\eta} = -\lambda\,\eta$). Ensuite, un projecteur orthogonal est utilisé pour projeter la tâche secondaire dans le noyau de la Jacobienne basée sur la norme de la tâche principale. La loi de commande est alors définie par:

$$\dot{\mathbf{q}} = \mathbf{J}_\eta^{+}\,\dot{\eta} + \mathbf{P}_\eta\,\mathbf{z} \tag{A.3}$$

$$\text{avec } \mathbf{J}_\eta = \gamma\|\mathbf{e}\|^{\gamma-2}\mathbf{e}^\top\mathbf{J} \in \mathbb{R}^{1\times n} \;,\; \mathbf{J}_\eta^{+} = \frac{1}{\gamma\|\mathbf{e}\|^{\gamma-2}(\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e})}\mathbf{J}^\top\mathbf{e}$$

$$\text{et } \mathbf{P}_\eta = \mathbf{I}_\eta - \mathbf{J}_\eta^{+}\mathbf{J}_\eta = \mathbf{P}_{\|\mathbf{e}\|} = \mathbf{I}_n - \frac{1}{\mathbf{e}^\top\,\mathbf{J}\,\mathbf{J}^\top\,\mathbf{e}}\,\mathbf{J}^\top\,\mathbf{e}\,\mathbf{e}^\top\,\mathbf{J}$$

Dans ce cas, les tâches/contraintes secondaires n'ont pas d'effet sur la norme de l'erreur de la tâche principale dans l'espace de la norme.

A cause de la singularité du projecteur à proximité de zéro, une stratégie appropriée est utilisée pour assurer le passage de ce projecteur à l'opérateur de projection classique dès que l'erreur se rapproche de zéro.

La stabilité du système peut être vérifiée à l'aide de la fonction de Lyapunov classique basée sur la norme d'erreur. Le principal avantage de cet opérateur est qu'il est toujours au moins de rang $(n-1)$ ce qui lui permet d'être utilisé même si la tâche principale est de rang plein. Cependant, le passage à la projection classique peut conduire à un comportement insatisfaisant et, dans certains cas, à l'instabilité.

A.2.3.4 Discussion

En comparant les différentes méthodes de résolution de la redondance et d'enchaînement des tâches, on remarque que la plupart de ces méthodes sont réalisées soit en regroupant les tâches ensemble et en utilisant une pondération statique/dynamique pour gérer la priorité entre eux (commande hybride), soit en intégrant ces tâches dans un formalisme continu pour l'enchaînement ou la commutation entre les différentes tâches (commande partitionnée, commutative et hiérarchique).

La commande hiérarchique est un cas particulier de la deuxième catégorie, qui consiste à projeter des tâches moins prioritaires dans l'espace nul de celles de plus haute priorité. Dans ce cas, deux techniques ont été utilisées pour définir l'opérateur de projection:

1. La première considère le projecteur qui répond à la condition de stabilité de Lyapunov classique (projecteur orthogonal $\mathbf{P_e}$) ou étendu (projecteur directionnel $\mathbf{P_z}$) qui est basée sur la norme de l'erreur.
2. La seconde projette directement la norme de l'erreur dans l'espace nul des tâches les plus prioritaires (projecteur basé sur la norme minimale $\mathbf{P}_\eta$), mais des problèmes de stabilité apparaissent dans ce cas et la loi de commande doit revenir à la projection classique des composantes de l'erreur à proximité de la convergence.

Dans certains cas, ces dernières méthodes ont une meilleure performance que celle classique, en particulier, en raison de l'augmentation du nombre de ddl disponibles pour l'application des tâches secondaires. Malgré ces améliorations, il existe plusieurs limitations liées à la définition des tâches appliquées, et au passage à une autre loi de commande près de la convergence par exemple. En outre, il n'existe pas de définition claire des applications où ces techniques de résolution de la redondance peuvent améliorer le comportement du système.

Par conséquent, dans la section suivante, une comparaison entre quatre techniques de résolution de la redondance est effectuée par simulation sur différents types de robots et sous plusieurs conditions. Différents critères de comparaison et de performance sont également définis et utilisés pour démontrer les avantages et les inconvénients de chaque loi de commande et les applications possibles pour chacune d'entre elles. Par ailleurs, une discussion sur les résultats de la simulation conduira à la définition de deux nouvelles méthodes généralisées pour la résolution de la redondance.

A.3 Comparaison des Méthodes de Résolution de la Redondance

Les différentes méthodes de résolution de la redondance et d'enchaînement des tâches sont résumées dans le tableau A.1, et comparées par rapport à plusieurs critères: la possibilité d'appliquer plusieurs tâches simultanément, le couplage entre les tâches, avoir une hiérarchie (priorités statiques/dynamiques) entre les tâches, et enfin la simplicité de réglage des paramètres dans la loi de commande.

A l'exception de la commande commutative, toutes les méthodes d'enchaînement des

tâches permettent l'exécution de plusieurs tâches simultanément. Cependant, pour gérer la priorité entre ces tâches, peu de lois de commande (comme la commande hybride et hiérarchique) peuvent être utilisées, la plus simple est la méthode de projection orthogonale où aucune paramétrisation n'est nécessaire. D'autre part, dans le cas d'un changement dynamique de la priorité entre les tâches, seule la commande hybride avec pondération dans l'espace des tâches peut être utilisée. Pour les autres méthodes, une re-définition ou ré-empilement des tâches exécutées dans la loi de commande est nécessaire.

Malgré les avantages de toutes les méthodes présentées, d'après les objectifs envisagés par cette thèse, quelques lois de contrôle sont seulement adéquates pour résoudre la redondance dans notre cas. En fait, on s'intéresse à la commande d'un système redondant multi-bras lors de l'exécution de plusieurs tâches simultanées avec une hiérarchie et priorité spécifique. Par conséquent, en se référant au tableau A.1, seule la méthode de commande hiérarchique et la méthode de commande hybride avec une pondération dans l'espace des tâches sont utiles dans

Type de Commande			**Tâches simultanées**	**Couplage entre tâches**	**Hierarchie**	**Type de priorité**	**Réglages des paramètres**
Commande partionnée (II.3)			✓	×	×	-	✓
Commande Commutative (II.4)			×	×	×	-	×
Commande Hybride	**Jacobienne Augmentée (II.6)**		✓	✓	×	-	✓
	Pondération dans l'espace articulaire	**Partitionée (II.7)**	✓	×	×	-	×
		Étendue (II.8)	✓	✓	×	-	×
	Pondération dans l'espace des tâches (II.9)	h_i **constant**	✓	✓	✓	Statique	×
		h_i **variable**	✓	✓	✓	Dynamique	×
Commande Hiérarchique	**Projection Classique P_e (II.13)**		✓	✓	✓	Statique	✓
	Projection Bidirectionalle P_z (II.26)		✓	✓	✓	Statique	×
	Méthode de la norme minimale P_η (II.32)		✓	✓	✓	Statique	×

Table A.1 – Comparaison entre les lois de commande pour la résolution de la redondance et l'enchaînement des tâches

la commande de tels systèmes.

Par conséquent, l'efficacité de l'utilisation de ces méthodes pour résoudre la redondance cinématique est abordée pour trouver la technique la plus appropriée qui doit être utilisée pour contrôler un système redondant général selon le degré de redondance du robot et l'application souhaitée.

La première partie adresse la méthodologie de comparaison: elle présente les robots utilisés, décrit les tâches appliquées et rappelle les lois de commande des méthodes comparées. Ensuite, dans la deuxième partie, on les critères de comparaison sont définis, en plus des différentes configurations du robot et des tâches appliquées. Les résultats de la simulation des méthodes considérées dans le cas de robots planaires et du robot LWR KUKA sont discutés dans la dernière partie.

A.3.1 Méthodologie de la comparison

A.3.1.1 Présentation des conditions d'étude

Pour faire une simple et large comparaison entre les différentes techniques de résolution de la redondance, plusieurs types de robots planaires sont utilisés dans la simulation. On considère 'l 'comme la dimension de la tâche(s) secondaire(s) , et'$r = n - m$' comme le degré de redondance du robot (rappelons que le robot a 'n' articulations et un effecteur de mobilité 'm); par conséquent, le comportement des techniques de résolution de la redondance seront étudiées au cas des robots planaires dans trois cas différents:

- $l < r$: **Système avec assez de ddl**
  Le robot a plus de ddl que la dimension des tâches secondaires appliquées.

- $l = r$: **Système avec un nombre exact de ddl**
  La tâche secondaire utilise exactement tous les ddl du robot.

- $l > r$: **Système sans assez de ddl**
  La tâche secondaire a besoin de plus de ddl que ceux disponibles dans le système.

A.3.1.2 Définition des tâches

En ce qui concerne les tâches appliquées, seul le positionnement de l'effecteur et de l'une des articulations intermédiaire du robot est considéré. Le choix de ces tâches, avec un nombre fixe de ddl, facilite l'interprétation du comportement du robot lors de l'application des méthodes choisies, et permet de définir des configurations particulières et significatives de tâches. Par conséquent, deux tâches sont définies avec priorité décroissante (tâche $\mathbf{T}_1$ a plus de priorité que la tâche $\mathbf{T}_2$):

- $\mathbf{T}_1$ pour contrôler la position et l'orientation de l'organe terminal.

- $\mathbf{T}_2$ pour contrôler la position d'une articulation intermédiaire du robot.

A.3.1.3 Présentation des robots

Pour les robots planaires, l'organe terminal a 3 dégrés de mobilité ($m = 3$), et la tâche secondaire $\mathbf{T}_2$ a toujours 2 ddl ($l = 2$). Alors, la comparaison est faite dans les cas suivants:

- Un robot 7R planaire avec 4 degrés de redondance (Système avec assez de ddl, $l < r$).
- Un robot 5R planaire avec 2 degrés de redondance (Système avec un nombre exact de ddl).
- Un robot 4R planaire avec 1 degré de redondance (Système sans assez de ddl, $l > r$).

Articulation	σ	α	d	θ	r
1	0	0	0	θ_1	0
2	0	0	1	θ_2	0
3	0	0	1	θ_3	0
4	0	0	1	θ_4	0
5	0	0	1	θ_5	0
6	0	0	1	θ_6	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	0	0	1	θ_N	0
N+1	0	0	1	0	0

Table A.2 – Paramètres D-HM d'un robot N-R planaire générique

Les paramètres de Denavit-Hartenberg Modifié (paramètres D-HM) [KK86] des robots planaires utilisés sont donnés dans le Tableau A.2 dans le cas général de robot N-R planaire (l_i= 1 m), et Fig. A.1 représente le cas de robot 5-R planaire.

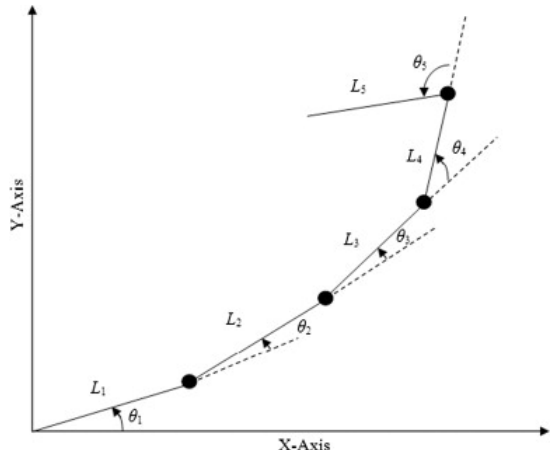


Figure A.1 – Schéma général du robot 5R-planaire

On rappelle que les tâches sont définies comme des couples $(\dot{\mathbf{e}}_\mathbf{i}, \mathbf{J_i})$ avec des priorités décroissantes, où $\dot{\mathbf{e}}_\mathbf{i}$ est la décroissance désirée de l'erreur entre la valeur actuelle et désirée, et $\mathbf{J_i}$ la jacobienne correspondante. On a choisi une décroissance exponentielle classique de l'erreur, donc $\dot{\mathbf{e}}_\mathbf{i} = -\lambda\ \mathbf{e_i}$ où λ est un gain constant qui contrôle la vitesse de convergence de la tâche.

A.3.1.4 Choix des lois de commande

On s'intéresse à l'étude du comportement du système lors de l'exécution de tâches simultanées avec une hiérarchie et une priorité spécifique. Ainsi, en référant au Tableau A.1, la commande hiérarchique et la commande hybride avec une pondération dans l'espace de la tâche peuvent seulement être utilisées. Par conséquent, les lois de commande suivants sont choisis pour être comparées:

Projecteur Orthogonale $\mathbf{P_e}$:
Dans le cas de deux tâches, la relation est donnée par:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_\mathbf{1} + (\mathbf{J_2}\ \mathbf{P_e})^+(\dot{\mathbf{e}}_\mathbf{2} - \mathbf{J_2}\ \dot{\mathbf{q}}_\mathbf{1}) \quad \text{avec} \quad \mathbf{P_e} = \mathbf{I_n} - \mathbf{J_1^+}\ \mathbf{J_1}$$

Projection Directionelle $\mathbf{P_z}$:

La loi de commande est donnée par:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_\mathbf{1} + \mathbf{P_z}\ \mathbf{J_2^+}\ \dot{\mathbf{e}}_\mathbf{2}$$

avec $\mathbf{P_z}$ l'opérateur de projection directionelle defini dans (A.2).

Solution de norme minimale $\mathbf{P}_\eta$:
La solution avec la méthode de norme minimale et la stratégie de passage à la projection orthogonale sont utilisées dans ce cas:

$$\dot{\mathbf{q}} = \mathbf{J_1^+}\ \dot{\mathbf{e}}_\mathbf{1} + (\mathbf{J_2}\ \mathbf{P}_\alpha)^+\ (\dot{\mathbf{e}}_\mathbf{2} - \mathbf{J_2}\ \dot{\mathbf{q}}_\mathbf{1})$$

La commande hybride avec la matrice de pondération H:
La loi de commande, definie par [KC11a], est utilisée:

$$\dot{\mathbf{q}} = (\mathbf{H}\ \mathbf{J_{ext}})^+\ \mathbf{H}\ \dot{\mathbf{e}}_\mathbf{ext}$$

aver $\mathbf{e_{ext}} = [\mathbf{e_1}\ ;\ \mathbf{e_2}]$ et $\mathbf{J_{ext}} = [\mathbf{J_1}\ ;\ \mathbf{J_2}]$ sont respectivement l'erreur de la tâche étendue et la Jacobienne étendue, et $\mathbf{H} = \text{diag}(h_i)$ la matrice de pondération.

Le schéma de commande qui est utilisé pour exécuter les deux tâches désirées, avec les techniques présentées pour l'enchaînement des tâches et la résolution de la redondance, est représenté dans la Figure A.2 ci-dessous:

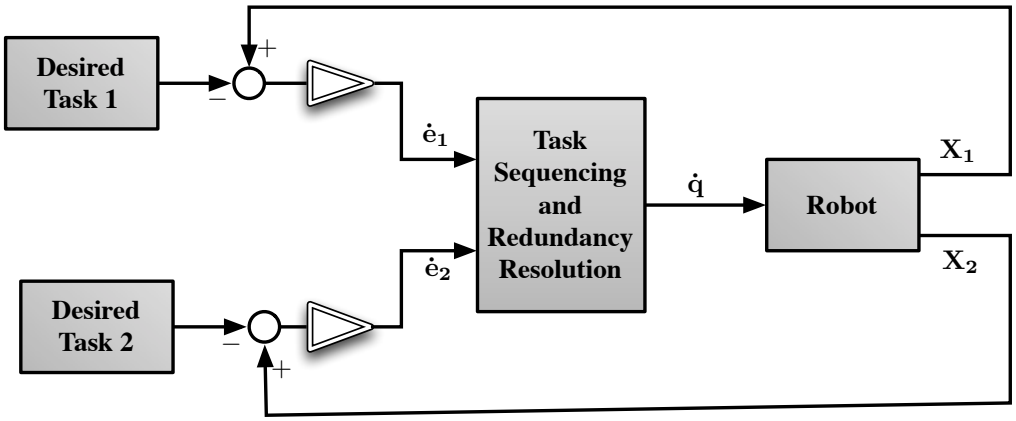


Figure A.2 – Représentation générale du schéma de commande

A.3.2 Critères de Comparaison

Les critères suivants sont considérés pour comparer le comportement du robot pour les différentes techniques de commande.

Trajectoires des points de contrôle:
La trajectoire des points de contrôle (PC) est étudiée pour vérifier si la décroissance exponentielle désirée, de l'erreur des tâches principales et secondaires, est respectée (trajectoire linéaire dans le plan).

Rang de la matrice de projection/pondération:
En analysant la variation du rang de la matrice de projection (noté **rank**(**P**)), on peut déterminer le nombre de ddl libres et qui peuvent être utilisés pour exécuter la(les) tâche(s) secondaire(s). En fait, si le nombre de ddl utilisé par la tâche principale augmente, le rang du projecteur diminue et moins de ddl peuvent être utilisés par le robot pour exécuter la tâche secondaire.

En cas de commande hybride, le rang de la matrice de pondération (noté **rank**(**H**)) représente le nombre des poids non nuls. Lorsque la contrainte est respectée, les pondérations correspondantes sont nulles et la matrice diminue de rang. D'autre part, si la contrainte n'est pas respectée, la loi de commande tente de déplacer le point de contrôle correspondant au domaine souhaité en utilisant une valeur non nulle pour les pondérations correspondantes, ainsi le rang de la matrice de pondération augmente.

Temps et ordre de convergence:
Pour les mêmes valeurs du gain pour la régulation de l'erreur, le temps de convergence ($\mathbf{t_{conv}}$) est utilisé pour comparer la vitesse de convergence des différentes techniques, et l'ordre de convergence de la tâche principale/secondaire appliquée(s).

Il correspond au moment où l'erreur de tâche devient inférieure à un seuil désiré (10^{-7} dans les simulations).

Indices de performance:
Une des méthodes quantitatives qui sont utilisées pour évaluer la performance du robot est la mesure de la manipulabilité du robot dans toutes les directions, ce qui est l'un des indices de performance clés de la dextérité globale des manipulateurs redondants.

Par conséquent, plusieurs indices de manipulabilité ont été utilisés dans l'étude de la cinématique des manipulateurs redondants: [SC82] a proposé un nombre de conditionnement, [Yos84] a proposé un indice de la dextérité globale des robots manipulateurs redondants, et a défini la manipulabilité ellipsoïde [Yos90] qui est utilisée pour calculer la mesure de manipulabilité w_m comme suit:

$$w_m = \sqrt{\det(\mathbf{J}\mathbf{J}^\top)} = \sigma_1.\sigma_2...\sigma_m$$

avec les scalaires $\sigma_1, \sigma_2, ...\sigma_m$ sont les 'm' valeurs singulières de $\mathbf{J}$ ordonnées du maximum au minimum.

[Yos90] a défini aussi le nombre de conditionnement qui est défini par le rapport entre la valeur singulière maximale et minimale.

$$w_{cond} = \frac{\sigma_1}{\sigma_m} \geq 1$$

Ces deux indices sont fusionnés dans un seul paramètre de singularité (PS) qui décrit le comportement cinématique global du robot, il est défini par:

$$W_{ps} = \sqrt{\frac{w_{cond}}{w_m}} = \sqrt{\frac{1}{\sigma_2...\sigma_m^2}} \tag{A.4}$$

Plus W_{ps} est élevé, plus le conditionnement est élevé, ou plus la manipulabilité est élevée, par conséquent, il se rapproche de la singularité.

Ce paramètre de singularité W_{ps} est utilisé dans la comparaison des simulations. Dans le cas de la commande hiérarchique deux indices de performance (W_{ps1} et W_{ps2}) pour la tâche principale et secondaire respectivement seront évaluées, et dans le cas de commande hybride, seul le paramètre de singularité qui correspond à la Jacobienne étendue $\mathbf{J_{ext}}$ est considéré.

Énergie cinétique globale:

Dans certains cas, le temps de convergence n'est pas l'indice le plus important, mais l'énergie consommée est considérée comme critère primordial [Ata07]. Cela peut être le cas lorsque la quantité d'énergie disponible est limitée. De nombreuses applications nécessitent un système de commande de mouvement pour déplacer un objet (charge) à partir du position de repos à une distance/orientation finale désirée dans une durée de temps fixe, et ensuite de retourner l'objet à sa position initiale.

En général, pour effectuer la tâche souhaitée, une loi de commande peut être conçu en utilisant un indice de performance optimal qui évalue l'énergie cinétique E_c utilisée tout au long de de l'exécution de la tâche:

$$E_c = \sum_{i=1}^{n} \frac{1}{2}\, \dot{\mathbf{q}}_\mathbf{i}^\top\, \dot{\mathbf{q}}_\mathbf{i} \tag{A.5}$$

aver $\dot{\mathbf{q}}_\mathbf{i}$ est la vitesse instantanée de la i^{me} articulation.

Cas des tâches non admissibles et incompatibles:

La robustesse d'une loi de commande dépend de son comportement et sa réaction dans le cas des configurations particulières ou irrégulières. En effet, grâce à des imprécisions dans la modélisation du système, ou à cause d'entrées erronées, la tâche secondaire peut atteindre une position désirée qui est en dehors de l'espace de travail du système.

Par conséquent, le comportement du système est évalué dans le cas où la pose désirée pour la tâche secondaire est inaccessible, son effet sur la tâche principale et sur la stabilité du système est également étudié.

La possibilité d'appliquer tels cas dans l'approche de commande permet, par exemple, d'utiliser un critère secondaire du déplacement du point de contrôle dans une direction désirée (qui est définie par une pose qui est hors de l'espace de travail du robot).

On s'attend alors, pour une pose désirée bien définie de la tâche principale, si la tâche secondaire (T_2) est inaccessible, la tâche de priorité plus élevé (T_1) est réglée à la valeur désirée sans aucune perturbation.

De plus, la réaction du système à une commande erronée du même point de contrôle pour deux positions désirées différentes est un cas intéressant à étudier. En fait, le comportement du système est également étudié lorsque deux tâches incompatibles sont commandées pour déplacer le même point de contrôle à deux position/orientation différentes. On s'attends à ce que le système reste stable, et la tâche avec une priorité plus élevée (T_1) soit exécuté.

Un tel cas se produit en raison de conflits entre les contraintes définies ou les critères de performance souhaités du système. Tenant compte d'un grand nombre de contraintes et critères qui sont nécessaires d'être pris en compte dans la loi de commande, cela peut conduire à l'apparition de mouvements qui forcent le point de contrôle pour se déplacer dans des directions opposées. Donc l'étude du comportement du système en cas de tâches incompatibles est nécessaire pour l'évaluation de la capacité des différentes techniques de résolution de la redondance à envisager de tels cas.

A.3.3 Résultats de la comparaison

La comparaison générale des différentes méthodes est résumée dans le Tableau A.3.

Projection orthogonale:

Les résultats des simulations montrent que la méthode classique de projection orthogonale $\mathbf{P_e}$ a le meilleur comportement: une trajectoire linéaire des points de contrôle, une décroissance exponentielle de l'erreur totale, un temps de convergence acceptable et une absence de réglage des paramètres. Cependant, des problèmes d'instabilité et de haute vitesses articulaires surviennent lorsque les tâches ne sont pas bien définies (dans le cas de tâches non admissibles ou incompatibles).

Projection directionnelle:

La méthode de projection directionnelle $\mathbf{P_z}$ est applicable lorsqu'une trajectoire linéaire des points de contrôle n'est pas indispensable, mais plus critique, c'est qu'elle donne le meilleur comportement si un nombre insuffisant de ddl est disponible pour exécuter toutes les tâches désirées. En outre, c'est la seule méthode qui maintiennent la stabilité du système en cas de tâches secondaires non admissibles, et en cas d'incompatibilité entre les tâches contrôlées.

De plus, pour la méthode de projection directionnelle, malgré l'amélioration dans le temps de convergence de la tâche principale de l'ordre de 5% dans le cas normal (à cause de la condition sur la fonction de Lyapunov pour être plus rapide que celle dans le cas de projection orthogonale), le temps de convergence de la tâche secondaire augmente avec le nombre de tâches appliquées, et ce temps peut ensuite infiniment augmenter si un grand nombre de tâches secondaires est considéré. En fait, cette tâche secondaire est réglée à la pose désirée, mais avec une précision insuffisante ($\approx 10^{-3}$), tandis que 10^{-7} est utilisée comme seuil pour les autres méthodes.

En fait, la valeur importante du temps de convergence pour la tâche secondaire, est due au

	CP Trajectoire	**Variation de l'erreur**	**Rang du Projecteur**	**Temps de Convergence**	$\mathbf{W}_{ps}$	$\mathbf{E}_c$
$\mathbf{P}_e$	Linéaire	Exponentielle	Fixe	Acceptable	Moyen	Faible
$\mathbf{P}_z$	Non linéaire	Non Exponentielle	Variable	Inacceptable	Moyen	Moyen
$\mathbf{P}_\eta$	Non linéaire	Non Exponentielle	Variable	Acceptable	Moyen	Haut
H	linéaire	Exponentielle	Variable	Acceptable	Haut	Faible

	Réglage des Paramètres	**T2 Non Admissible**	**Tâches Incompatibles**	**l = r**	**l > r**
$\mathbf{P}_e$	Facile	Mauvais	Mauvais	Bon	Bon
$\mathbf{P}_z$	Difficile	Bon	Bon	Bon	Meilleur
$\mathbf{P}_\eta$	Moyen	Mauvais	Mauvais	Bon	Bon
H	Difficile	Moyen	Mauvais	Moyen	Moyen

Table A.3 – Comparaison des lois de contrôle pour l'enchaînement des tâches

choix de la méthode d'enchaînement des tâches (II.34) qui ne prend pas en considération la partie de la tâche secondaire qui est exécutée par la tâche principale de la solution minimale, ce qui conduit à une régulation imprécise de la tâche secondaire (comme expliqué dans le paragraphe II.4.2). En effet, ce choix d'enchaînement tâche est seulement considéré pour $\mathbf{P_z}$ en raison de la complexité pour démontrer la stabilité du système avec la méthode d'enchaînement optimale donnée par (II.41).

La méthode de projection directionnelle utilise la décomposition SVD de la jacobienne de la tâche principale ($\mathbf{J_1} = \mathbf{U_1}\mathbf{\Sigma_1}\mathbf{V_1}^\top$) pour projeter la tâche secondaire dans l'espace des valeurs singulières. En fait, la discontinuité dans la décomposition en valeurs singulières, qui apparaît dans la discontinuité des colonnes des matrices $\mathbf{U}_1$ et $\mathbf{V}_1$, conduit à une oscillation entre les différentes conditions dans la définition du projecteur (II.25 et II.29), et cela entraine des oscillations des vitesses articulaires qui peuvent conduire à des accélérations élevées et donc à des vibrations du système.

Solution de norme minimale:

Malgré le rang le plus élevé du projecteur, et alors le plus grand nombre de ddl disponibles dans le cas de la méthode basée sur la norme minimum $\mathbf{P}_\eta$, le principal problème de cette approche réside dans la stratégie de commutation effectuée près de la convergence à cause de la singularité du projecteur.

En fait, le système est bloqué dans la zone de commutation lors de l'application des tâches non admissibles et incompatibles. Contrairement à la méthode de projection orthogonale, la trajectoire des points de contrôle n'est pas linéaire en raison de la décroissance exponentielle de la norme de l'erreur et non pas des composantes de l'erreur.

Commande hybride:

Enfin, la commande hybride avec une matrice de pondération $\mathbf{H}$ dans l'espace de la tâche nécessite un réglage des paramètres, surtout dans le cas des poids h_i dynamiques. De plus, il est difficile de démontrer la stabilité du système. Un comportement acceptable est remarqué dans le cas où assez de ddl sont présents pour exécuter toutes les tâches.

Toutefois, lorsqu'il n'y a pas suffisamment de ddl, le système se rapproche plus des singularités ainsi les vitesses et accélérations articulaires nécessaires pour exécuter le mouvement désiré sont plus élevées. De plus, des problèmes apparaissent en cas de tâches non admissibles et incompatibles. Néanmoins, la trajectoire linéaire et la décroissance exponentielle de l'erreur des tâches sont notées lors de l'application de la commande hybride.

A.3.4 Projecteur unifiant les méthodes orthogonale et directionnelle

La comparaison des résultats de simulation montre qu'il n'y a pas une seule méthode de résolution de la redondance qui a un comportement parfait dans tous les cas étudiés. Cependant, on peut remarquer que les méthodes de projection orthogonale et directionnelle ont des performances acceptables dans des cas complémentaires. En fait, le premier est facilement applicable lorsqu'un nombre suffisant de dll est disponible, et la seconde dans les cas spéciaux où

il n'y a pas assez de ddl, des tâches non admissibles ou incompatibles.

Revenant à la construction théorique de la méthode de projection directionnelle, elle consiste à autoriser les composantes des tâches secondaires qui vont dans le même sens que la tâche principale et ne perturbent pas la stabilité du système. Ainsi, si aucune des composantes des tâches secondaires est activé, cette approche ne permet d'effectuer que la tâche principale, et donc revient à appliquer l'approche classique de projection orthogonale. Par conséquent, ces deux approches peuvent être fusionnées en un seul formalisme général.

De plus, dans plusieurs cas, il n'est pas nécessaire de considérer totalement la tâche secondaire dans la loi de commande, par exemple quand elle correspond à une contrainte qui n'est pas toujours présente. Dans de tels cas, ces tâches/contraintes secondaires peuvent être entièrement activées, partiellement activées (dans une direction désirée) ou même désactivées et projetées dans l'espace nul de la tâche principale (comme dans le cas de la projection orthogonale). Par conséquent, on propose d'ajouter un terme de pondération sur le projecteur de la tâche secondaire qui est réglé en fonction de la valeur de la tâche/contrainte désirée.

De plus, en cas de projection directionnelle, on peut remarquer, dans les résultats de simulation, qu'une discontinuité dans la vitesse articulaire apparaît quand il y a un changement dans le rang du projecteur. En fait, ce comportement est dû à la variation des éléments du projecteur qui changent entre deux valeurs (0 et 1) en fonction du signe de l'erreur de la tâche principale, et de la vitesse articulaire de la tâche secondaire (projetée dans l'espace des valeurs singulières). Ainsi, une transition plus douce entre ces deux valeurs permet d'améliorer le comportement du système.

La loi de commande proposée est alors définie par:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}} + \mathbf{P}_{\mathbf{w}}\,\mathbf{z} \tag{A.6}$$

$$\text{avec}\ \ \mathbf{P}_{\mathbf{w}} = \mathbf{V}\,\mathbf{P}_{(\tilde{\mathbf{z}})}\mathbf{V}^{\top}\ \ ,\ \ \mathbf{P}_{(\tilde{\mathbf{z}})} = \begin{bmatrix} p_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & p_n(\tilde{z}) \end{bmatrix}\ \ ,\ \ p_i(\tilde{\mathbf{z}}) = \begin{cases} 1 & \text{if} \quad i > m \ \text{ or } \ \tilde{z}_i = 0 \\ w & \text{if} \quad \tilde{z}_i\,\tilde{l}_i\,\sigma_i < 0 \\ 0 & \text{if} \quad \tilde{z}_i\,\tilde{l}_i\,\sigma_i > 0 \end{cases}$$

$$m = rang(\mathbf{J})\ \ ,\ \ \tilde{\mathbf{z}} = \mathbf{V}\mathbf{z}\ \ ,\ \ \mathbf{\Sigma} = diag(\sigma_i)\ \ ,\ \ \widetilde{\nabla\mathcal{L}} = \mathbf{U}^{\top}\nabla\mathcal{L}\ \ ,\ \text{et}\ \ \widetilde{\nabla\mathcal{L}} = (\tilde{l}_1, ..., \tilde{l}_m).$$

Une fonction sigmoïde peut être utilisée pour la variation de la valeur de w en fonction de la norme de l'erreur de la tâche secondaire $\|\mathbf{e}_2\|$:

$$w\left(\|\mathbf{e}_2\|\right) = \begin{cases} 1 & \text{si} \quad \|\mathbf{e}_2\| \geq \alpha_1 \\ \frac{1}{1+\exp\left(20\,\frac{\|\mathbf{e}_2\|-\alpha_1}{\alpha_0-\alpha_1}-10\right)} & \text{si} \quad \alpha_0 < \|\mathbf{e}_2\| < \alpha_1 \\ 0 & \text{si} \quad \|\mathbf{e}_2\| \leq \alpha_0 \end{cases} \tag{A.7}$$

avec α_0, α_1 sont deux seuils qui définissent les conditions de début et de fin de la période de commutation, ils doivent être choisis de telle sorte que le système ne converge pas trop vite pendant cet interval. w est donc une fonction décroissante monotone continue comme représenté sur la Fig. A.3.

De plus, l'utilisation d'une telle fonction de transition permet non seulement de passer de la projection directionnelle à celle orthogonale près de la convergence, mais aussi d'assurer la régulation exacte de la tâche secondaire en utilisant la loi de commande définie dans (II.41) au contraire de la méthode directionnelle qui amène à un temps de convergence élevé (car il ne tient pas compte de l'effet de la tâche principale sur la tâche secondaire).

En conclusion, la définition du projecteur $\mathbf{P}_w$ peut être utilisé pour combiner entre les projecteurs directionnel et orthogonal, et permet alors de bénéficier des avantages de ces deux méthodes dans les différents cas tout en maintenant la stabilité du système. Cette méthode permet la convergence exacte des deux tâches dans un temps acceptable, elle diminue également les oscillations dans le rang du projecteur et donne une allure continue des vitesses articulaires.

Les résultats des simulations sur les différents cas montrent une meilleure performance de cette méthode par rapport aux méthodes de projection orthogonale et directionnelle. Enfin, l'utilisation de cette loi de commande donne plus de liberté dans le choix de la méthode de projection, en fonction du comportement souhaité. Par exemple, si tous les conditions (ddl suffisant, des tâches bien définies, ...) sont présents pour l'application de la méthode de projection orthogonale, le choix d'une valeur fixe de $w = 0$ conduit au comportement désiré.

A.3.5 Opérateur de projection généralisé

Pour surmonter le problème de discontinuité dû à la décomposition SVD de la jacobienne de la tâche principale (qui conduit à des discontinuités des vitesses articulaires et l'oscillation du système), on définit dans cette partie un opérateur de projection simple qui bénéficie de la tâche secondaire pour appliquer plus rapidement la tâche principale tout en maintenant la stabilité du système et la régulation exacte des deux tâches (quand assez de ddl sont disponibles).

Cette méthode de résolution de la redondance résout le problème de discontinuité qui apparait à cause de la projection dans l'espace des valeurs singulières. Ainsi, cette méthode généralisée aboutit à un opérateur unique qui intègre plusieurs techniques existantes.

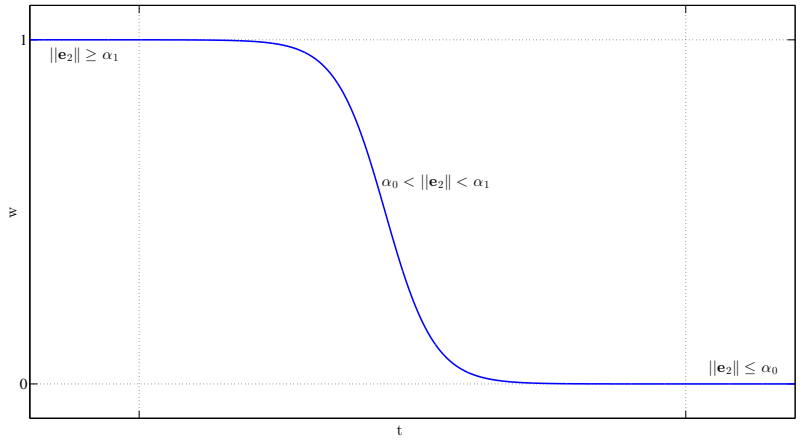


Figure A.3 – Variation de la valeur de pondération w par rapport à la norme de l'erreur $\|\mathbf{e}_2\|$

La loi de commande est définie par:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_1 + \mathbf{P}_\beta \, \mathbf{z} = \mathbf{J}^+ \, \dot{\mathbf{e}} + (\mathbf{I_n} - \beta \, \mathbf{J}^+ \, \mathbf{J}) \, \mathbf{z} \tag{A.8}$$

avec $\beta \in [0, 1]$ un paramètre de réglage qui est utilisé pour assurer la stabilité du système, tout en respectant la condition ci-dessous:

$$(1 - \beta) \, \mathbf{e}^\top \, \mathbf{J} \, \mathbf{z} \leq \mathbf{0} \tag{A.9}$$

En effet, ce projecteur permet au système de bénéficier des avantages de la méthode de projection orthogonale (en prenant $\beta = 1$), surtout la simplicité et la régulation exacte de la tâche secondaire près de la convergence. En outre, il englobe la commande hybride (pour $\beta = 0$) en tenant compte de la tâche secondaire. Enfin, c'est une sorte de projection directionnelle de la tâche secondaire avec des contraintes moins restrictives que la projection classique directionnelle.

Une fonction sigmoïde peut être utilisée pour la variation de la valeur de β en fonction de la norme de l'erreur de la tâche secondaire $\|\mathbf{e}_2\|$ et la condition de stabilité:

$$\beta\,(\|\mathbf{e}_2\|) = \begin{cases} 0 & \text{si} \quad \mathbf{e}^\top \, \mathbf{J} \, \mathbf{z} \leq \alpha_0 \\ \frac{1}{1+\exp\left(-20\frac{\|\mathbf{e}_2\|-a}{b-a}+10\right)} & \text{si} \quad \alpha_0 < \mathbf{e}^\top \, \mathbf{J} \, \mathbf{z} < 0 \\ 1 & \text{si} \quad \mathbf{e}^\top \, \mathbf{J} \, \mathbf{z} \geq 0 \end{cases} \tag{A.10}$$

β est donc une fonction croissante monotone continue où les paramètres a et b sont réglés pour contrôler le comportement et la vitesse de la transition, et α_0 est une valeur de seuil qui définit le début de la période de commutation, elle doit être choisie de telle sorte que le système ne passe pas trop rapidement vers la projection orthogonale (Fig. A.4).

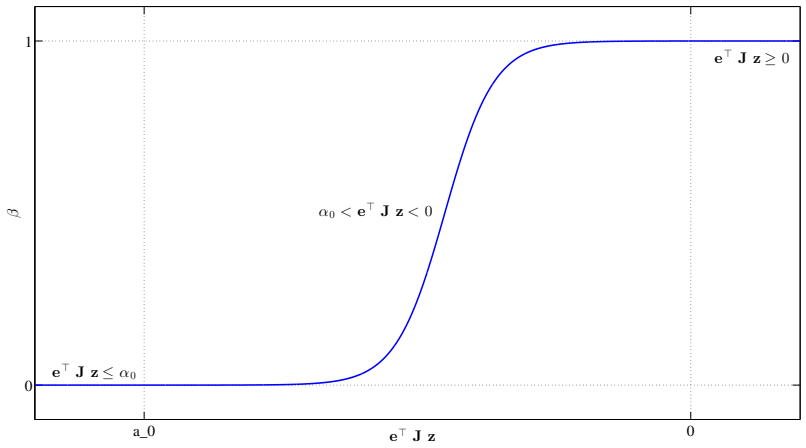


Figure A.4 – Variation de la valeur de pondération β par rapport à la norme de l'erreur $\|\mathbf{e}_2\|$

Les résultats des simulations, sur les différents cas, montrent un comportement acceptable du projecteur généralisé $\mathbf{P}_\beta$ et garantie alors la faisabilité de la loi de commande définie.

A.4 Commande des systèmes multi-bras

L'intelligence d'un système robotisé est caractérisée par trois capacités fonctionnelles. **Contrôle au niveau de la tâche**: le système robotisé doit prendre directement les commandes au niveau des tâches sans aucune planification ou décomposition au niveau des articulations. **Généralisation de la tâche**: les systèmes de contrôle doivent être conçus pour une large classe de tâches qui peuvent être appliquées à différents systèmes robotiques. **Flexibilité aux changements de l'environnement**: le système robotisé doit être capable de gérer et d'intégrer certains événements imprévus ou incertains [XTB96].

D'après cela, une technique générique est nécessaire pour contrôler les systèmes robotiques multi-bras. Par conséquent, on développe, dans cette section, l'approche de multi-points de contrôle aux niveaux cinématique et dynamique.

A.4.1 Présentation de l'approche de multi-points de contrôle

Cette approche est basée sur la définition de plusieurs points de contrôle, répartis sur des parties appropriées du robot. Pour chaque point de contrôle, on étudie et commande son interaction avec l'environnement/robot en utilisant des données parvenues des différents capteurs proprioceptif et/ou exteroceptif.

En étudiant ces interactions, on définit des possibles fonctions de tâches qui contrôlent le comportement du robot durant l'exécution des tâches désirées. Ces fonctions de tâches sont ensuite empilées dans une structure qui gère la priorité et la compatibilité des tâches entre elles en utilisant l'une des méthodes d'enchainement de tâches déjà présentées et comparées.

L'approche de multi-points de contrôle utilise uniquement l'interaction entre plusieurs points de contrôle sur le robot avec l'environnement ou le corps du robot, pour gérer et commander les systèmes redondants. La première étape est alors de choisir les points de contrôle appropriés avant de définir les contraintes et les tâches pertinentes sur le système robotique.

A.4.2 Définition des points de contrôle

Les points de contrôle sont choisis en fonction de l'architecture du robot et les tâches à exécuter. Les plus courants sont le centre de masse (CoM), les points des organes terminaux et les points de contact avec l'environnement. Le point de contrôle CoM est généralement utilisé pour contrôler l'équilibre et de la posture du robot. Ceux sur les effecteurs du système sont utilisés pour définir les tâches d'interaction entre le robot et les objets de l'environnement. De plus, certains points de contrôle sur le corps du robot peuvent être utilisés pour éviter l'auto collision, ou la collision avec l'environnement.

A.4.3 Technique de commande

Indépendamment des capteurs utilisés, on définit le vecteur des primitives (ou vecteur d'information) $\mathbf{s} \in \mathbb{R}^m$. Il peut s'agir d'un ensemble de données fournies par un capteur scalaire (la distance), ou d'un vecteur de données en cas de capteurs plus complexes comme les caméras et les capteurs d'effort.

La valeur de consigne de ce vecteur est $\mathbf{s}^*$, donc l'erreur peut être directement calculée par:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{A.11}$$

La loi de commande générale utilisée est alors donnée par:

$$\dot{\mathbf{q}} = -\lambda \, (\mathbf{L_s} \, {}^{\mathbf{s}}\mathbf{W_{cp}} \, {}^{\mathbf{cp}}\mathbf{J_q})^{+} \, (\mathbf{s} - \mathbf{s}^*) \tag{A.12}$$

Ainsi, pour appliquer une tâche désirée, la vitesse articulaire du robot peut être calculée en utilisant cette loi de commande après avoir déterminé les informations suivantes:

— La valeur actuelle ($\mathbf{s}$) et désirée ($\mathbf{s}^*$) des primitives données par les capteurs.

— La Jacobienne du robot (${}^{\mathbf{cp}}\mathbf{J_q}$) au point de contrôle dans le repère $\mathcal{F}_{cp}$ au point de contrôle.

— La matrice de transformation (${}^{\mathbf{s}}\mathbf{W_{cp}}$) entre le repère du point de controle $\mathcal{F}_{cp}$ et le repère capteur $\mathcal{F}_s$.

— La matrice d'interaction $\mathbf{L_s}$ qui correspond à la primitive $\mathbf{s}$. Elle correspond à la variation de $\mathbf{s}$ par rapport à la vitesse $\mathbf{v_s}$ au point capteur.

A.4.4 Tâches robotiques génériques

La décomposition de l'application désirée en plusieurs tâches robotiques élémentaires et génériques est une première étape de la commande du robot. En outre, chaque tâche générique peut être définie comme un triplet composé de: la fonction de tâche, le point de contrôle et le niveau de priorité. Ainsi, on développe dans cette partie plusieurs tâches génériques essentielles pour tous les domaines d'application de la robotique.

A.4.4.1 Tâche de positionnement

C'est la tâche générique la plus simple, qui peut être généralisée à un grand nombre de tâches robotiques. Elle consiste à déplacer un point de contrôle de sa pose actuelle à une position et/ou orientation désirée. Cette pose désirée peut être un point fixe (tâche de positionnement) ou un point en mouvement (tâche de suivi de cible).

A.4.4.2 Tâche de coopération

Un enjeu important dans le processus de manipulation de robots redondants est la coopération entre les différents effecteurs pour exécuter une tâche désirée, comme par exemple le déplacement d'un objet long en utilisant les pinces de deux robots. Dans ce cas, la fonction de tâche est définie par deux points de contrôle afin de contrôler la configuration relative entre eux et maintenir la position/orientation relative désirée.

On définit pour cette tâche, la pose de deux points de contrôle $\mathbf{s_1}$ et $\mathbf{s_2}$ pour $\mathbf{CP_1}$ et $\mathbf{CP_2}$ respectivement. Ainsi, la pose relative du second point de contrôle par rapport au premier est

considérée comme primitive à contrôler, elle est donnée par $\mathbf{s} = (\mathbf{s_2} - \mathbf{s_1})$, et la configuration désirée est noté $\mathbf{s}^*$. De plus, puisqu'on contrôle généralement la pose relative, donc 6 ddl sont utilisés par cette tâche et la loi de commande est donnée par:

$$\dot{\mathbf{q}} = -\lambda \left[-\mathbf{L}_{s_1}\, {}^{s_1}\mathbf{W}_{cp_1}\, {}^{cp_1}\mathbf{J}_q \qquad \mathbf{L}_{s_2}\, {}^{s_2}\mathbf{W}_{cp_2}\, {}^{cp_2}\mathbf{J}_q \right]^+ (\mathbf{s} - \mathbf{s}^*) \tag{A.13}$$

A.4.4.3 Tâche de visibilité

Le problème de la visibilité des primitives a été largement abordée dans la littérature. L'utilisation des schémas classique d'asservissement visuel 2D et 3D en supposant une mauvaise calibration avec un large déplacement initial de la caméra, la cible peut quitter le champ de vision de la caméra [MCB99, CHPV04]. D'où la nécessité d'avoir des lois de commande qui sont en mesure de conserver ces primitives dans le champ de vision de la caméra pour obtenir un signal de retour fiable pendant le processus d'asservissement.

On aborde alors ce problème en utilisant une fonction de tâche: la tâche de visibilité. En général, le point de contrôle utilisé est l'articulation sur lequel le système de vision est monté. Dans le cas des robots humanoïdes, l'articulation de la tête est considérée. La Jacobienne articulaire ${}^{cp}\mathbf{J}_q$ au niveau du point de contrôle considéré est donnée par des capteurs proprioceptifs.

A.4.5 Commande dynamique des systèmes multi-bras

L'approche basée sur la cinématique a toujours été préférée dans plusieurs applications robotiques en raison de sa simplicité. Cependant, pour les systèmes de second ordre, l'approche basée sur l'accélération est plus attrayante, surtout lorsqu'elle est utilisée en conjonction avec une commande de dynamique inverse qui doit explicitement connaître les accélérations dans l'espace articulaire.

De plus, les accélérations obtenues peuvent être directement utilisées comme signaux de référence (avec les positions et les vitesses correspondantes) pour un contrôleur dynamique dans l'espace de la tâche. Cependant, un système de résolution de la redondance au second ordre est toujours plus exigeant en termes de charge de calcul.

Dans l'approche dynamique, les tâches génériques sont définis à l'aide des lois de commande pour la résolution de la redondance aux niveaux d'accélération (A.14) et de couple (A.15):

$$\ddot{\mathbf{q}} = \mathbf{J}_s^+ \left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s \dot{\mathbf{q}} \right) \tag{A.14}$$

$$\tau = \left(\mathbf{J}_s \mathbf{M}^{-1} \right)^{\#\mathbf{M}} \left(\ddot{\mathbf{e}} - \dot{\mathbf{J}}_s \dot{\mathbf{q}} + \mathbf{J}_s \mathbf{M}^{-1} (\mathbf{C} + \mathbf{G} + \tau_e) \right) \tag{A.15}$$

Quelle que soit la loi de commande utilisée (au niveau de l'accélération ou niveau du couple) et la trajectoire désirée du point de contrôle; afin de définir une tâche au niveau dynamique les données suivantes doivent être déterminées (en plus des éléments $\mathbf{M}, \mathbf{C}, \mathbf{G}, \tau_e$ du modèle dynamique du robot):

A.4.5.1 Vecteurs de primitives s et $\dot{s}$

La valeur du vecteur de la primitive utilisée $\mathbf{s}$ est donnée par des capteurs proprioceptifs ou exteroceptifs. Plusieurs types de primitives peuvent être utilisées comme celles présentées dans le cas de l'asservissement visuel cinématique.

La variation de la primitive utilisée $\dot{\mathbf{s}}$ peut être déterminée en utilisant le modèle de robot ou d'autres techniques basées capteurs:

- En utilisant la relation $\dot{\mathbf{s}} = \mathbf{J_s}\,\dot{\mathbf{q}}$, la variation de la primitive peut être basée sur un modèle estimé à partir des vitesses articulaires et de la Jacobienne du capteur: $\dot{\mathbf{s}} = \mathbf{J_s}\,\dot{\mathbf{q}}$.

- La variation de la primitive peut être calculée en appliquant l'équation: $\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v_s}$, en utilisant la vitesse $\mathbf{v_s}$ au point capteur qui est déterminée par l'utilisation de certaines techniques spécialisées sur les données exteroceptifs du robot comme dans [DAAAM08].

- En cas d'un mouvement lent, une simplification de cette méthode consiste à étudier la variation instantanée de la primitive tel que $\dot{\mathbf{s}} = \frac{d\mathbf{s}}{dt} = \frac{\mathbf{s_i} - \mathbf{s_{i-1}}}{dt}$ avec dt suffisamment petit.

A.4.5.2 Valeurs désirées des primitives $\mathbf{s}^*$ et $\dot{\mathbf{s}}^*$

En fonction de la mobilité de l'objet cible, la pose et la vitesse désirée du point de contrôle, $\mathbf{s}^*$ and $\dot{\mathbf{s}}^*$ respectivement, peuvent être constantes pour un objet fixe ou variable pour des objets mobiles. Dans ce dernier cas, les mêmes méthodes que ci-dessus sont utilisés pour déterminer les valeurs désirées.

A.4.5.3 La Jacobienne capteur $\mathbf{J_s}$

La Jacobienne capteur, exprimée par $\mathbf{J_s} = \mathbf{L_s}\,{}^{\mathbf{s}}\mathbf{W_{cp}}\,{}^{\mathbf{cp}}\mathbf{J_q}$ en utilisant la Jacobienne articulaire ${}^{\mathbf{cp}}\mathbf{J_q}$ au point de contrôle donnée par le modèle de robot, la matrice d'interaction $\mathbf{L_s}$ qui dépend de la primitive choisie, et la matrice de transformation ${}^{\mathbf{s}}\mathbf{W_{cp}}$.

A.4.5.4 Variation de la Jacobienne capteur $\dot{\mathbf{J}}_\mathbf{s}$

Considérant une matrice de transformation ${}^{\mathbf{s}}\mathbf{W_{cp}}$ constante et une matrice d'interaction $\mathbf{L_s}$ constante, la variation de la Jacobienne capteur $\dot{\mathbf{J}}_\mathbf{s}$ est directement liée à la variation de la Jacobienne articulaire $\dot{\mathbf{J}}_\mathbf{q}$ (Hessienne) qui peut être dérivée soit par différentiation numérique ou par formulation analytique [Hou05].

Les matrices Hessiennes sont généralement calculées par des calculs manuels, méthodes finie différenciation, ou par une technique de différentiation automatique comme dans [RASR11, FA12]. Plusieurs outils, comme le logiciel symbolique SYMORO+, sont utilisés pour calculer la valeur du produit de la Hessienne par la vitesse articulaire ($\dot{\mathbf{J}}_\mathbf{q}\dot{\mathbf{q}}$) sous une forme symbolique [KC+97].

A.5 Application aux robots humanoïdes et multi-bras

A.5.1 Commande de la plateforme multi-bras ARMS

Cette application s'inscrit dans le cadre du projet ANR ARMS [ARM] qui étudie la robotisation de la séparation des muscles de pièces de viande bovine. Un système robotique multi-bras est utilisé pour permettre de réaliser et contrôler simultanément quatre actions mécaniques principales (préhension, traction, poussée, et/ou coupe).

Pour concrétiser cet objectif on utilise une plateforme multi-bras composée de 3 robots Kuka LWR à 7 ddl chacun. La préhension de la viande est assurée par le premier, tandis que le deuxième fait la coupe de la viande. Un troisième robot maintient un système de vision sur son organe terminal pour assurer une boucle fermée dans le schéma de commande.

L'approche cinématique de multi-points de contrôle est utilisée pour commander ce système ayant 21 dll en utilisant 10 points de contrôle sur les organes terminaux des robots, la pointe de l'outil de coupe et d'autres points sur le corps des robots. Ces points de contrôle sont utilisés pour appliquer les tâches suivantes:

— **Tâche de coupe**:
Elle utilise la tâche générique de positionnement (à 6 ddl) pour commander le point de contrôle sur la pointe du couteau à suivre la trajectoire qui sépare les deux muscles (la ligne de coupe est actualisé par la déformation de la pièce de viande).

— **Tâche de traction**:
Le point de contrôle sur l'effecteur du deuxième robot est utilisé pour appliquer la tâche générique de positionnement (de 6 ddl) pour la préhension de la viande.

— **Tâche de visibilité**:
Cette tâche générique consiste à déplacer le système de vision fixé sur l'organe terminal du troisième robot pour centre la ligne de coupe dans le centre de l'image dans une orientation et à une distance prédéfinie. Cette tâche utilise alors 4 ddl.

En plus de ces tâches, les contraintes suivantes doivent être aussi respecter dans la commande du système:

— **Evitement d'auto-collision**:
Sur chaque robot, ce type de contraintes est défini pour éviter la collision entre deux points de contrôle (sur l'organe terminal et sur le corps du robot). Cette contrainte utilise 3 ddl pour assurer que la position relative entre les deux points de contrôle soit plus grande qu'un seuil prédéfini.

— **Evitement de collision**:
Six contraintes de ce type sont considérées pour éviter la collision entre les organes terminaux et les points de contrôle sur le corps des robots. Le même principe que précédemment est utilisé pour la définition de ces contraintes à 3 ddl chacune.

— **Evitement d'occultation**:
Cette contrainte consiste à déplacer le point de contrôle sur le troisième robot pour éviter que l'organe terminal du robot de coupe entre dans le champs de vision de la caméra, ce qui peut entrainer une occultation de la ligne de coupe.

— **Evitement de butées articulaires**:
Trois contraintes d'évitement de butées articulaires sont considérées sur les trois robots.

A cause du grand nombre de ddl nécessaires pour exécuter toutes ces tâches simultanément tout en respectant les contraintes du système multi bras à 21 ddl seulement, la méthode de résolution de la redondance basée sur la commande hiérarchique avec le nouveau projecteur défini dans (A.4) est utilisée pour enchainer les 3 tâches principales et les contraintes du système.

Les résultats de simulation sur l'environnement Matlab/Simulink/Adams valide l'efficacité d'utiliser l'approche de multi-points de contrôle avec le nouveau projecteur de la résolution de la redondance. Cela permet d'enchainer plusieurs tâches robotiques tout en respectant les différentes contraintes du système et par suite d'exécuter l'application désirée de découpe et de séparation de muscles.

A.5.2 Auto-localisation du robot Nao pendant la marche

Cette application consiste à localiser le robot humanoïde Nao par rapport à son environnement pendant la marche.

Le robot Nao H25 [GHB+09], développé par Aldebaran Robotics, est un robot bipède avec 25 ddl (5 dans chaque jambe, 1 dans le bassin, 2 dans la tête, 5 dans chaque bras et 2 mains actionnés). Il a des mains à 3 doigts utilisés pour saisir et tenir de petits objets (300 g avec les deux mains).

La tête du robot Nao (v3.2) contient deux caméras CMOS identiques ayant une résolution de 640×480. Elles peuvent capturer jusqu'à 30 images par seconde. La configuration des caméras ne permet pas d'avoir la vision stéréo puisque leurs champs de vision ne se chevauchent pas. La première caméra se dirige vers l'avant et l'autre vers le bas afin de voir le sol en face de Nao. La deuxième est utilisée dans cette application.

Au cours de l'exécution de cette tâche, seul un modèle approximatif de la porte et une partie de la pièce sont utilisés (A.5). En fait, le robot suit la porte et donc il est localisé par rapport à l'environnement tout en marchant. Nous utilisons le modèle de la porte et les lignes de l'espace tout autour pour initialiser le module de suivi basé modèle (MBT développé sous ViSP) qui donne ensuite la pose de la porte dans l'espace de la caméra en temps réel. A partir de la valeur donnée, la pose du robot peut être calculée.

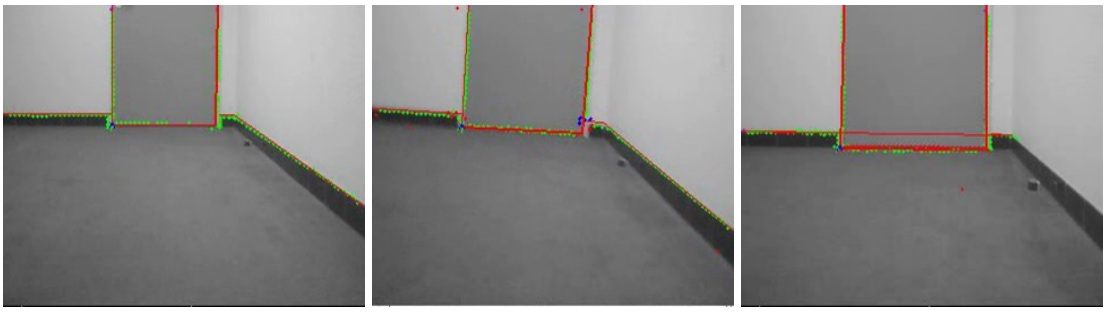


Figure A.5 – Photos de la tâche d'auto-localisation pendant la marche du robot

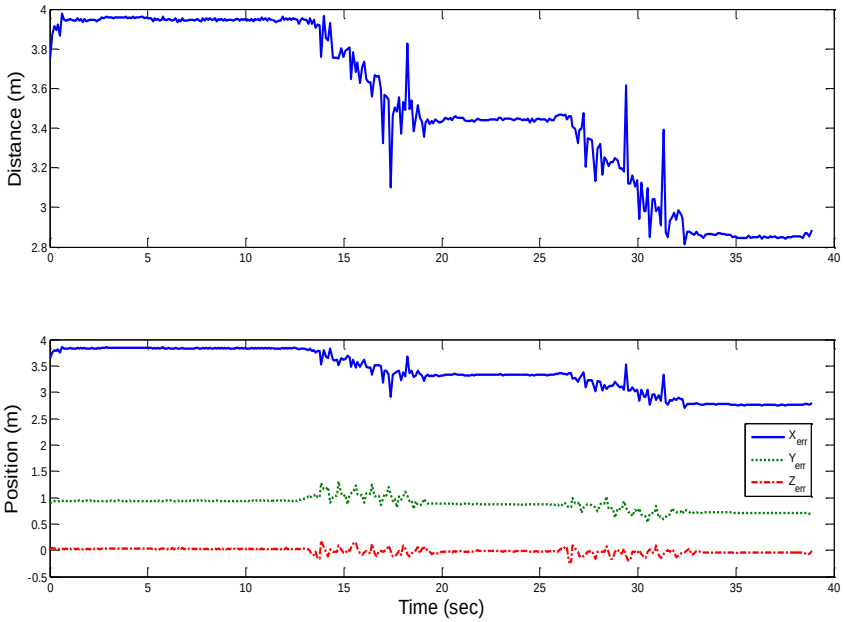


Figure A.6 – Résultats expérimentaux de la tâche d'auto-localisation dans le repère du robot Nao

Le module de suivi est automatiquement ré-initialisé à chaque fois que le suivi a échoué en raison des déplacements brusques de la caméra lors de la marche par exemple. Il utilise la dernière pose trouvée pour réinitialiser le module de suivi. Dans nos expériences, le robot marche en boucle ouverte en direction de la porte pour une distance de 1 mètre.

Les résultats expérimentaux montrent que le robot suit avec succès la porte tout en marchant la distance souhaitée de 1 mètre avec une erreur finale de 6 cm, ce qui nous permet de localiser le robot avec succès. L'erreur peut être relativement grande par rapport aux autres méthodes de localisation, mais elle est acceptable et suffisante puisqu'il s'agit d'une navigation d'intérieur pour des tâches de manipulation. Ces travauxx sont illustrés dans les publications suivantes [AMCM13b, AMCM13c].

A.5.3 Navigation de Nao pour l'application d'une tâche d'assistance

Dans cette application, un capteur externe est fixé dans l'environnement du robot, et utilisé pour localiser le robot pendant la marche et la saisie d'un objet de l'environnement.

Un système de vision 3D externe (Kinect) est utilisé pour effectuer les tâches désirées. C'est un capteur récent, puissant et à relativement faible coût. Il est composé de deux capteurs optiques dont l'interaction permet une analyse de la scène en trois dimensions. L'un des capteurs est une caméra RGB qui a une fréquence vidéo de 30 images par seconde. La résolution de l'image donnée par cet appareil est de 640x480 pixels. Le second capteur a pour but d'obtenir des informations sur la profondeur des objets de l'environnement.

Ainsi, cette méthode s'appuie sur le développement d'une boucle fermée robuste pour le système de navigation des robots humanoïdes dans un environnement domestique en utilisant la localisation par des caméras Kinect. Ainsi, l'objectif principal est de développer un système robuste qui permet de suivre et de localiser le robot lors de la marche, et de contrôler son mou-

vement d'une façon suffisamment précise pour récupérer un objet sur le sol.

Dans ce cas, la primitive considérée est la position/orientation relative entre le robot et la balle sur le sol : $\mathbf{s} = [x, y, \theta_z]^\top$. Pour la valeur désirée de la primitive $\mathbf{s}^*$, la stratégie suivante est envisagée: au début, tandis que l'erreur est supérieure à un seuil donné e_{min}, le robot se dirige vers la ligne joignant sa pose actuelle et la destination finale, ensuite, lorsque l'erreur est inférieure à ce seuil, le robot change vers son orientation finale:

$$s^* = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ atan(y/x) \end{bmatrix} & if \quad \|\mathbf{e}\| > e_{min} \\ \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & if \quad \|\mathbf{e}\| < e_{min} \end{cases} \tag{A.16}$$

Le calcul de la matrice d'interaction $\mathbf{L_s}$ qui sera utilisée dans la loi de commande du robot donne:

$$\mathbf{L_s} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{y}{x^2+y^2} & \frac{-x}{x^2+y^2} & 1 \end{bmatrix} & if \quad \|\mathbf{e}\| > e_{min} \\ \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & if \quad \|\mathbf{e}\| < e_{min} \end{cases} \tag{A.17}$$

De la même manière que les êtres humains [KSL07], les robots bipèdes souffrent de l'incapacité à maintenir une trajectoire droite quand ils marchent sans vision. En fait, des légères différences entre les pas du robot, causées par la friction ou par la puissance variable des moteurs, produisent une déviation de la marche qui peut être estimée et corrigée. Par conséquent, une simple procédure de correction du virage est désormais performée.

La loi de commande finale est alors donnée en fonction du rayon R du cercle d'ajustement et de la pente m de la ligne d'ajustement qui sont utilisés pour calculer la compensation de la vitesse angulaire et latérale respectivement:

$$\mathbf{V_r} = -\lambda \, \mathbf{L_{veering}} \, \mathbf{L}_s^+ (\mathbf{s} - \mathbf{s}^*) \tag{A.18}$$

avec $\mathbf{L_{veering}} = \begin{bmatrix} 1 & 0 & 0 \\ -m & 1 & 0 \\ -R & 0 & 1 \end{bmatrix}$.

Les résultats expérimentaux montrent que lorsqu'on utilise uniquement l'odométrie, le robot ne peut pas atteindre la destination (la pose finale est à 40 cm de celle désirée). Toutefois, la localisation et la navigation en boucle fermée permettent au robot d'atteindre l'objectif avec quelques centimètres de précision: $(x, y, \theta_z) = (0.012, 0.018, 0.07)$. Ce travail a conduit aux publications suivantes [AMCM13a, CAMM12].

A.5.4 Manipulation d'objets avec le robot HRP-2

Dans cette application, on exécute un scénario de manipulation avec le robot humanoïde HRP-2. L'approche cinématique de multi-points de contrôle est appliquée sur HRP-2 pour faire plusieurs tâches simultanées. En fait, toute la structure du robot est contrôlée pour maintenir l'équilibre du système dans une position verticale, et pour saisir des objets rigides à l'aide des mains droite et gauche.

HRP-2 est une plateforme robotique du "Humanoid Robotics Project" dirigé par "Manufacturing Science and Technology Center" (MSTC) en 2002 [HRP]. HRP-2 a 30 ddl dont 2 ddl pour la hanche, 6 ddl pour chaque bras et jambe, 2 ddl pour la tête et 1 ddl pour chaque main.

Pour cette application, on considère 6 points de contrôle sur le corps du robot: les pieds gauche et droit, les poignets gauche et droit, la tête du robot et le centre du masse du robot. Ensuite, après avoir défini le scénario souhaité et les points de contrôle pertinents, on utilise les tâches génériques pour définir les tâches appropriées à appliquer et leurs lois de commande correspondantes. Enfin, le formalisme de "stack of tasks", basé sur la commande hiérarchique avec la méthode de projection directionnelle, est utilisée pour empiler et de gérer la priorité entre ces tâches.

Les tâches appliquées sont résumées dans le tableau A.4, avec les points de contrôle correspondants, le nombre de ddl utilisé et la priorité des tâches. Elles sont simulées sur le logiciel

Priorité	**Nom de la tâche**	**DDL utilisés**	**Points de contrôle**
1	Tâche d'équilibre	3 DDL	Centre de masse
2	Tâche de posture	6 DDL	Pieds gauche et droit
3	Tâche de saisie (main gauche)	3 DDL	Main gauche
4	Tâche de saisie (main droite)	6 DDL	Main droite
5	Tâche de visibilité	2 DDL	Tête du robot
6	Tâche de préhension	2×1 DDL	Préhenseur gauche et droit

Table A.4 – Récapitulatif des tâches simulées avec leurs points de contrôle correspondants, le nombre de degrés de liberté utilisés et leurs priorité

OpenHRP dédié aux robots HRP à l'aide d'une caméra externe AVT MARLIN (F-131B, noir et blanc) qui donne 25 images par seconde pour simuler le système de vision du robot.

La première tâche maintient le centre de masse du robot dans sa position initiale, et la seconde tâche maintient une pose relative fixe entre les jambes du robot. Ces tâches prioritaires sont conservées et respectées pendant tout le processus de manipulation.

La tâche de troisième priorité consiste à saisir un objet quelconque par la main gauche du robot à l'aide d'une caméra externe pour faire le suivi de l'objet (à l'aide de la module MBT de ViSP) qui permet alors de déterminer la position désirée de la tâche. Simultanément, la deuxième tâche de saisie déplace la main droite du robot pour une pose prédéfinie désirée (6 ddl). Pour la dernière tâche, la tâche de visibilité déplace les deux ddl de la tête du robot pour que l'image de l'objet désiré soit dans le centre de l'image de la caméra.

Les résultats des simulations montrent une exécution réussie et robuste du scénario de manipulation avec le robot HRP-2 en utilisant l'approche de multi-point de contrôle pour contrôler le système redondant et la méthode de projection directionnelle pour gérer et appliquer les différentes tâches simultanées. Cette manipulation permet d'exploiter la redondance du robot ayant 30 ddl pour exécuter plusieurs tâches (qui utilisent 22 ddl au total).

A.5.5 Manipulation d'objets avec le robot Nao

Dans cette application, on utilise l'approche cinématique de multi-points de contrôle pour appliquer plusieurs tâches de service sur le robot humanoïde Nao. Le scénario envisagé consiste à intégrer les techniques d'asservissement visuel en temps réel sur le robot humanoïde en boucle fermée, pour effectuer différentes tâches de manipulation. La technique de suivi basé modèle est utilisée en temps réel pour appliquer les tâches d'asservissement visuel 3D sur le robot Nao. Deux points de contrôle sont utilisés: la main gauche et la tête du robot. Plusieurs tâches élémentaires sont utilisées pour effectuer cet objectif: asservissement de la tête pour la tâche de visibilité, la détection, le suivi et la manipulation des objets de l'environnement.

Les résultats expérimentaux montrent une variation acceptable de l'erreur de position horizontale et verticale pendant la tâche d'asservissement de la tête (tâche de visibilité): initialement l'objet est à une distance d'environ 30 cm du centre de l'image de la caméra, on remarque que cette erreur converge à zéro pendant 29 secondes avec une précision de 5 mm.

Pour les tâches de pré-saisie et saisie de l'objet, la main du robot est initialement à une distance approximative de 25 cm par rapport à la position de pré-saisie prédéfinie, et la main est tournée de 180 degrés par rapport à l'objet. Au cours de cette tâche, l'erreur de position et de l'angle de lacet convergent d'une façon exponentielle à zéro en 31 sec. Ces tâches de (pré) saisie sont exécutées avec succès avec une précision de 5 mm et 1 mm respectivement et 3 degrés pour l'orientation.

Les résultats expérimentaux vérifient la possibilité et l'efficacité de l'utilisation des techniques MBT pour appliquer en temps réel un asservissement visuel 3D sur un simple robot humanoïde. Les résultats des tâches de suivi et de saisie, qui ont été répétés plusieurs fois, montrent que cette méthode est robuste à l'occultation par la main du robot, et résiste à un léger

mouvement de l'objet en raison de la collision main-objet. Cette technique a été utilisée dans les publications suivantes [AMSCM11, AMCM13b, AMCM13c].

En outre, une exécution plus rapide de cette tâche peut être effectuée en utilisant un autre type de décroissance de l'erreur, que l'exponentielle classique, afin d'améliorer le temps de convergence de la tâche. Par conséquent, deux nouvelles stratégies sont développées pour améliorer les performances de la tâche. Les résultats expérimentaux montrent une amélioration entre 40% et 55% sans perdre la stabilité du système. Ce travail a été présenté dans [AMCM12].

A.6 Conclusion générale

A.6.1 Conclusions et contributions

La travail de recherche décrit dans cette thèse est composé de deux parties: la première adresse la problématique de la résolution de la redondance et de l'enchaînement des tâches, tandis que la deuxième présente des applications en simulation et en temps réel des tâches de service sur plusieurs robots humanoïdes et multi-bras. Les contributions sont résumées ci dessous:

Classification des redondances
En plus de la redondance qui vient de la conception mécanique d'origine du robot, d'autres types de redondances se produisent à cause de l'interaction entre le robot et son environnement ou à cause des outils sensoriels utilisées dans les applications. La première contribution est le développement d'une classification détaillée des différents types de redondances qui peuvent être présentes ou qui apparaissent dans une plateforme robotique générale. En plus des définitions, cette organisation est illustrée par plusieurs exemples pertinents des divers domaines de la robotique.

Classification des méthodes de résolution de la redondance
Diverses méthodes de résolution de la redondance et de séquences de tâches sont présentées et discutées dans une notation unifiée pour simplifier la comparaison. D'après cette classification, on remarque que la plupart des méthodes existantes pour la résolution de la redondance et l'enchaînement des tâches sont construites soit en regroupant les tâches désirées ensemble dans un seul système étendu et en utilisant une pondération statique/dynamique pour gérer leur priorité, ou par l'empilement de ces tâches avec un processus continu d'enchaînement ou de commutation entre les différentes tâches. De plus, une extension de la condition de stabilité de la loi de commande conduit à une augmentation du nombre de ddl disponibles et donc à une amélioration de la performance du système.

Comparaison des méthodes de résolution de la redondance cinématique
Pour exécuter plusieurs tâches simultanées avec une hiérarchie bien spécifique, une comparaison entre les techniques appropriées de résolution de la redondance est réalisée par simulation sur différents types de robots planaires redondants et sur le robot Kuka LWR à 7 ddl.

Différents critères de performance sont définis et plusieurs configurations de tâches sont appliqués pour montrer les avantages et les inconvénients de chaque loi de commande et les

applications où chacune peut être convenablement appliquée. A partir de ces résultats de comparaison, la commande optimale et la solution la plus utile est calculée en fonction du nombre de ddl du système et de la tâche, en plus du type des tâches et des contraintes appliquées.

Nouvelles techniques de résolution de la redondance
Pour bénéficier des avantages de la méthode de projection orthogonale et directionnelle dans les différents cas d'étude, un nouveau opérateur de projection est défini pour contrôler la contribution des mouvements produits par la tâche secondaire dans la loi de commande. En outre, il aide la tâche principale à être exécuter plus rapidement en contrôlant la direction de la projection de la tâche secondaire. De plus, le projecteur utilise une fonction de transition pour passer entre la projection directionnelle et la projection orthogonale classique près de la convergence pour effectuer la régulation précise de la tâche secondaire.

Un problème dans la méthode de projection directionnelle, et par conséquent dans le premier projecteur défini, est constaté au niveau des vitesses articulaires: une discontinuité apparaît dans plusieurs cas, à cause de la discontinuité de l'espace des valeurs singulières où les erreurs des tâches sont projetées. Cette discontinuité conduit à des oscillations du projecteur lors du passage entre les différents conditions dans la définition de la loi de commande, et ainsi la vibration du système.

Par conséquent, un nouveau projecteur simple et généralisé est définie en considérant la condition de stabilité sur tout le vecteur d'erreur sans projection de ses éléments dans cet espace. Cela permet une diminution du nombre de paramètres qui doivent être réglés, et une exécution parfaite des tâches désirées dans un temps de convergence acceptable.

L'approche cinématique/dynamique de multi-points de contrôle
Pour contrôler n'importe quel système multi-bras redondant, on a développé l'approche de multi-points de contrôle en cinématique et dynamique qui consiste à étudier l'interaction robot/ environnement en plusieurs points du système.

Ce formalisme général utilise les techniques précédemment discutées de résolution de la redondance pour exécuter le scénario désiré, qui est décomposé en un ensemble de tâches génériques (généralisées pour assurer l'adaptabilité de la méthode à différentes plateformes robotiques). Divers types de capteurs intégrés et externes sont utilisés pour réaliser une commande basée capteur précise et robuste.

La localisation et la navigation du robot Nao
La localisation du robot est abordée selon deux techniques différentes: la première consiste à utiliser uniquement les caméras embarquées pour localiser le robot, pendant la marche, en utilisant un modèle 3D approximatif d'une partie de l'environnement, et la seconde approche utilise le capteur Kinect externe pour suivre un modèle partiel du robot en utilisant la technique de suivi basée sur le nuage des points.

De plus, des modules supplémentaires sont développés pour améliorer la robustesse et la performance de ces méthodes. Pour la première approche, une ré-initialisation automatique du suivi est utilisée pour prévenir tâche de localisation de l'échec, et pour la seconde, une compensation du virage latéral et angulaire du robot est effectuée pour corriger la déviation de la

marche causée par la friction, ou la variation de la puissance des moteurs par exemple.

Les résultats expérimentaux montrent la bonne exécution des tâches désirées dans les deux cas: dans le cas de l'auto-localisation, les résultats montrent que le robot peut être suivi avec succès tout en marchant, et des meilleurs résultats sont obtenus pour la localisation à l'aide du capteur externe avec une amélioration significative de la méthode de navigation par rapport aux autres techniques basées sur l'odométrie. De plus, la précision de cette méthode de localisation permet au robot de récupérer des objets sur le sol avec succès.

Tâches de manipulation sur les robots HRP-2 et Nao
La tâche de manipulation est appliquée sur deux robots humanoïdes différents: HRP-2 et Nao. Pour le premier système, l'approche cinématique de multi-points de contrôle et le formalisme de projection directionnelle pour la résolution de redondance sont utilisés pour contrôler ce système redondant lors de l'application des différentes tâches simultanées. En fait, au cours de l'exécution de ces tâches, le robot est entièrement contrôlé pour maintenir son équilibre et sa posture verticale tout en effectuant les tâches de manipulation à l'aide de ses deux mains. Les résultats de simulation montrent l'exécution réussie et robuste du scénario de manipulation avec le robot HRP-2 en utilisant ces approches.

Dans la deuxième partie, l'approche cinématique de multi-points de contrôle est également utilisée pour effectuer un scénario concret sur le robot humanoïde Nao qui consiste à exécuter des tâches de manipulation dans un environnement quotidien. Les résultats expérimentaux soulignent la possibilité et l'efficacité de l'utilisation des techniques MBT pour appliquer,en temps réel, un asservissement visuel 3D sur un robot humanoïde simple.

Cette méthode est robuste à l'occultation de la caméra par la main du robot et robuste aux petits déplacements de l'objet en raison de la collision main-objet. En outre, une exécution plus rapide de cette tâche peut être faite en utilisant d'autres méthodes de régulation, que celle classique exponentielle, afin d'améliorer le temps de convergence de la tâche.

A.6.2 Limitations et travaux futurs

Les études théoriques et la simulation des différentes méthodes de résolution de la redondance et d'enchaînement des tâches permettent de faciliter le développement futur de nouvelles techniques pour améliorer la performance de celles existantes ou pour mettre en œuvre de nouvelles solutions au problème de la redondance.

En fait, le développement de nouvelles techniques peut être réalisé, par exemple, en définissant un nouveau comportement des tâches exécutées, prolongeant ainsi la condition de stabilité pour augmenter les degrés de liberté du système, ou en utilisant d'autres techniques d'inversion que la pseudo-inverse classique.

De plus, il est possible de définir d'autres opérateurs de projection dans la commande hiérarchique est basée sur l'utilisation d'autres fonctions de Lyapunov pour prouver la stabilité du système. En effet, en plus de la fonction classique basée sur la norme de l'erreur, d'autres fonctions peuvent être définies afin d'optimiser les différents paramètres, mais dans tous les cas un espace de projection continu doit être défini pour éviter les oscillations du système.

Pour l'application sur le système multi-bras, l'approche dynamique de multi-points de contrôle peut être utilisé pour commander ce système au niveau du couple (ce qui n'est pas permis pour les autres plateformes à cause de leur architecture de commande).

Pour les tâches de localisation et de manipulation effectuées sur les robots Nao et HRP-2, un contrôle supplémentaire du mouvement oscillatoire du robot lors de la marche peut améliorer largement le comportement du robot et diminuer le temps d'exécution de la tâche désirée. Des modules supplémentaires pourraient être utilisés pour compenser dynamiquement les mouvements de la caméra lors de la marche du robot, intégrer plus de capteurs externes dans le système, et pour coopérer avec les capteurs proprioceptifs du robot.

D'autres capteurs pourraient être utilisés pour améliorer la réactivité de manipulation contre les changements dynamiques ou inhabituels dans l'environnement, en particulier ceux qui apparaissent après la saisie de l'objet où de nombreuses occultations et des mouvements brutaux apparaissent.

List of Publications

Book Chapter

- Abou Moughlbay A., Cervera E., Martinet P., **Model Based Visual Servoing Tasks with an Autonomous Humanoid Robot** (2013), Frontiers of Intelligent Autonomous Systems, Studies in Computational Intelligence Volume 466 (pp. 149-162). DOI: 10.1007/978-3-642-35485-4_12. Springer Berlin Heidelberg.

International Journals

- Abou Moughlbay A., Martinet P., **How to take advantage of redundancy in robotic applications: Review and enhancement**, Journal of Intelligent and Robotic Systems, to appear, 2014.

- Abou Moughlbay A., Long P., Khalil W., Martinet P., **Kinematic Control of a Multi-Arm Redundant System for Meat Cutting** (en cours).

International Conferences

- Long P., Abou Moughlbay A., Khalil W., Martinet P., **Robotic Meat Cutting**, ICT-PAMM Workshop, Kumamoto, Japan, November 9-10th, 2013.

- Martinet P., Abou Moughlbay A., Long P., **Control of Redundancy in complex systems: from theoretical concepts to applications** (2013), International Workshop on Wireless Communications and User centered Services in Pervasive Environments, Hanoi, Vietnam, 19-20 September, 2013.

- Abou Moughlbay A., Cervera E., Martinet P., **Real-Time Model Based Visual Servoing Tasks on a Humanoid Robot** (2013) In S. Lee, H. Cho, K.-J. Yoon, J. Lee (Eds.), Intelligent Autonomous Systems 12 SE - 30 (Vol. 193, pp. 321–333). Springer Berlin Heidelberg.

- Abou Moughlbay A., Cervera E., Martinet P., **Humanoid Robot Localization and Navigation in Domestic Environment using RGBD Sensor** (2013), 1st International Conference on Technology for Helping People with Special Needs (ICTHP-2013), February 18-20, 2013, Riyadh, Kingdom of Saudi Arabia.

- Abou Moughlbay A., Cervera E., Martinet P., **Error Regulation Strategies for Model Based Visual Servoing Tasks - Application to Autonomous Object Grasping with**

Nao Robot (2012) 12th International Conference on Control, Automation, Robotics and Vision (ICARCV) - 5 - 7 December 2012 - Guangzhou, China.

- Cervera E., Abou Moughlbay A., Martinet P., **Localization and Navigation of an Assistive Humanoid Robot in a Smart Environment** (2012) Workshop on Assistance and Service Robotics in a Human Environment, IROS 2012, 12 October 2012, Vila Moura, Algrave, Portugal.

- Abou Moughlbay A., Sorribes J.J., Cervera E., Martinet P., **Model-based Visual Servoing Tasks on the Nao Platform** (2011) IROS Standard Platform Demonstrations - September 25-30, 2011 - San Francisco, California.

National Conferences

- Abou Moughlbay A., Martinet P., Mansard N., **Commande par vision d'un robot redondant multi bras : Manipulation à deux bras avec le HRP2** (2010) 5èmes Journées Nationales de la Robotique Humanoïde, 3 - 4 Juin 2010, Futuroscope, Poitiers, France.

- Abou Moughlbay A., Martinet P., **Commande d'un système multi-bras pour la manipulation d'objets: Application à la commande de HRP-2** (2010) Journées de l'Ecole Doctorale EDSPI, 29-30 Avril 2010, Clermont Ferrand, France.

Project's Reports

- Long P., Abou Moughlbay A., **Report on Collision Avoidance** (2013), Deliverable D5.4 for ARMS project (ANR-10-SEGI-002): A multi arms robotic system for muscle separation.

- Long P., Abou Moughlbay A., **Definition and Implementation of Multi-Arm Control Framework** (2013), Deliverable D5.3 for ARMS project (ANR-10-SEGI-002): A multi arms robotic system for muscle separation.

- Long P., Abou Moughlbay A., **Definition and implementation redundancy criteria** (2013), Deliverable D5.2 for ARMS project (ANR-10-SEGI-002): A multi arms robotic system for muscle separation.

- Long P., Abou Moughlbay A., **State of the art on redundant system modeling and control** (2012), Deliverable D5.1 for ARMS project (ANR-10-SEGI-002): A multi arms robotic system for muscle separation.

Bibliography

[AAA+00] R.O. Ambrose, H. Aldridge, R.S. Askew, R.R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems and their Applications*, 15(4):57–63, 2000. 3

[AAH88] C.H. An, C.G. Atkeson, and J.M. Hollerbach. *Model-based control of a robot manipulator*, volume 16. MIT press Cambridge, MA, 1988. 10

[ACC10] G. Allibert, E. Courtial, and F. Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Transactions on Robotics*, 26(5):933–939, 2010. 119

[ACVB09] B.D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *International Journal of Robotics and Autonomous Systems*, 57(5):469–483, 2009. 9

[Ahm92] S. Ahmed. Issues in repeatability of redundant manipulator control. *Department of Electrical Engineering: Ohio State University*, 1992. 124

[AHN+08] S. Aomura, M. Harada, T. Nagatomo, S. Yanagihara, and M. Tachibana. A study on co-operative motion planning of a dual manipulator system for measuring radioactivity. *Industrial robot*, 35(6):541–548, 2008. 3

[AMCM12] A. Abou Moughlbay, E. Cervera, and P. Martinet. Error regulation strategies for model based visual servoing tasks - application to autonomous object grasping with nao robot. In *12th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Guangzhou, China, 5 - 7 December 2012. 194, 232

[AMCM13a] A. Abou Moughlbay, E. Cervera, and P. Martinet. Humanoid robot localization and navigation in domestic environment using rgbd sensor. In *1st International Conference on Technology Helping People with Special Needs (ICTHP-2013)*, Riyadh - Saudi Arabia, 2013. 193, 229

[AMCM13b] A. Abou Moughlbay, E. Cervera, and P. Martinet. Model based visual servoing tasks with an autonomous humanoid robot. In Sukhan Lee, Kwang-Joon Yoon, and Jangmyung Lee, editors, *Frontiers of Intelligent Autonomous Systems*, volume 466 of *Studies in Computational Intelligence*, pages 149–162. Springer Berlin Heidelberg, 2013. 193, 194, 228, 232

[AMCM13c] A. Abou Moughlbay, E. Cervera, and P. Martinet. Real-time model based visual servoing tasks on a humanoid robot. In Sukhan Lee, Hyungsuck Cho, Kwang-Joon Yoon, and Jangmyung Lee, editors, *Intelligent Autonomous Systems 12*, volume 193 of *Advances in Intelligent Systems and Computing*, pages 321–333. Springer Berlin Heidelberg, 2013. 193, 194, 228, 232

[AMMM10] A. Abou Moughlbay, P. Martinet, and N. Mansard. Commande par vision d'un robot redondant multi bras : Manipulation à deux bras avec le hrp2. In *5èmes Journées Nationales de la Robotique Humanoïde*, Futuroscope, Poitiers, France, 3 - 4 Juin 2010. 194

[AMSCM11] A. Abou Moughlbay, J.J. Sorribes, E. Cervera, and P. Martinet. Model-based visual servoing tasks on the nao platform. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS Standard Platform Demonstrations*, San Francisco, California, 25-30 September 2011. 194, 232

[Ant09] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5):985–994, 2009. 40, 41

[ARM] ARMS. A multi arms robotic system for muscle separation. http://arms.irccyn.ec-nantes.fr/. ANR project (ANR-10-SEGI-002), Last visited on september 2012. 3, 139, 199, 226

[ASEG+08] A. Albu-Schaffer, O. Eiberger, M. Grebenstein, S. Haddadin, C. Ott, T. Wimbock, S. Wolf, and G. Hirzinger. Soft robotics. *IEEE Robotics & Automation Magazine*, 15(3):20–30, 2008. 22

[ASY+93] Y. Asari, H. Sato, T. Yoshimi, K. Tatsuno, and K. Asano. Development of model-based remote maintenance robot system. iv. a practical stiffness control method for redundant robot arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1245–1251, jul 1993. 126

[Ata07] A.A. Ata. Optimal trajectory planning of manipulators: a review. *Journal of Engineering Science and Technology*, 2(1):32–54, 2007. 56, 215

[BB98] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 323–329, 1998. 41

[BDMS84] A. Balestrino, G. De Maria, and L. Sciavicco. Robust control of robotic manipulators. In *9th IFAC World Congress*, volume 5, pages 2435–2440, 1984. 32

[Bed90] N.S. Bedrossian. Classification of singular configurations for redundant manipulators. In *IEEE International Conference on Robotics and Automation*, pages 818–823, 1990. 134

[BHB84] J. Baillieul, J. Hollerbach, and R. Brockett. Programming and control of kinematically redundant manipulators. In *23rd IEEE Conference on Decision and Control*, volume 23, pages 768–774, 1984. 31

[Bil87] J. Billingsley. A system for partitioned control of a robot. *Control Theory and Applications*, 134(5):309–316, september 1987. 24

[BKX08] O. Brock, J. Kuffner, and J. Xiao. Motion for manipulation tasks. *Handbook of robotics*, pages 615–645, 2008. 177

[Blo10] R. Bloss. Robotics innovations at the 2009 assembly technology expo. *International Journal on Industrial Robot*, 37(5):427–430, 2010. 4

[BS94] R.A. Brooks and L.A. Stein. Building brains for bodies. *Autonomous Robots*, 1(1):7–25, 1994. 5

[BSBB06] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. In Henrik I. Christensen, editor, *EUROS*, volume 22 of *Springer Tracks in Advanced Robotics*, pages 195–209. Springer, 2006. 167

[CAMM12] E. Cervera, A. Abou Moughlbay, and P. Martinet. Localization and navigation of an assistive humanoid robot in a smart environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Assistance and Service Robotics in a Human Environment*, Vila Moura, Algrave, Portugal, 12 October 2012. 193, 229

[CB97] E.S. Conkur and R. Buckingham. Clarifying the definition of redundancy as used in robotics. *Robotica*, 15(05):583–586, 1997. 19

[CC98] P. Chiacchio and S Chiaverini. *Complex Robotic Systems*, volume 233 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, London, 1998.

[CC01] A. Crétual and F. Chaumette. Visual servoing based on image motion. *The International Journal of Robotics Research*, 20(11):857–877, 2001. 132, 133

[CCMV08] F. Caccavale, P. Chiacchio, A. Marino, and L. Villani. Six-dof impedance control of dual-arm cooperative manipulators. *IEEE/ASME Transactions on Mechatronics*, 13(5):576–586, 2008. 4

[CCSS91] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal of Robotics Research*, 10(4):410–425, 1991. 29, 41, 126

[CD95] T.F. Chan and R.V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 11(2):286–292, 1995. 26

[CDW+12] C. Chen, X. Duan, X. Wang, X. Zhu, and M. Li. Kinematics analysis and trajectory planning of a multiarm medical robot assisted maxillofacial surgery. In *International Conference on Complex Medical Engineering*, pages 229–234, july 2012. 3

[Cec11] M. Ceccarelli. Problems and Issues for Service Robots in New Applications. *International Journal of Social Robotics*, 3(3):299–312, April 2011. 8

[Cera] E. Cervera. Javiss - java-based visual servo simulator. http://www.robot.uji.es/research/projects/javiss. Last visited on october 2012. 154

[Cerb] E. Cervera. Visual servoing toolbox for matlab/simulink. http://vstoolbox.sourceforge.net. Last visited on october 2012. 154

[Cer06] E. Cervera. A cross-platform network-ready visual servo simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2314–2319, 2006. 154

[CH01] P.I. Corke and S.A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, 2001. 118

[CH06] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006. 11, 104, 111, 115

[CH07] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Transactions on Robotics and Automation*, 14(1):109–118, 2007. 112

[Cha98] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *The confluence of vision and control*, pages 66–78, 1998. 113

[Chi88] S.L. Chiu. Task compatibility of manipulator postures. *The International Journal of Robotics Research*, 7(5):13–21, 1988. 22

[Chi97] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997. 44

[CHPV04] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. Keeping features in the field of view in eye-in-hand visual servoing: A switching approach. *IEEE Transactions on Robotics*, 20(5):908–914, 2004. 118, 119, 224

[CKP+10] D. Corbel, S. Krut, F. Pierrot, et al. Enhancing pkm accuracy by separating actuation and measurement: A 3 dof case study. *ASME Journal of Mechanisms and Robotics (JMR)*, 2(3):1–11, 2010. 12, 22, 206

[CMC06] A.I. Comport, E. Marchand, and F. Chaumette. Statistically robust 2-d visual servoing. *IEEE Transactions on Robotics*, 22(2):415–420, 2006. 27

[Cor96] P.I. Corke. A robotics toolbox for matlab. *IEEE Transactions on Robotics and Automation*, 3(1):24–32, 1996. 154

[Cor05] P.I. Corke. The machine vision toolbox: a matlab toolbox for vision and vision-based control. *IEEE Transactions on Robotics and Automation*, 12(4):16–25, 2005. 154

[COW08] S. Chiaverini, G. Oriolo, and I.D. Walker. *Kinematically redundant manipulators*. Springer, 2008. 25

[CP12] M.P.G. Castro and T. Peynot. Laser-to-radar sensing redundancy for resilient perception in adverse environmental conditions. In *Beyond Laser and Vision: Alternative Sensing Techniques for Robotics Perception, Workshop, Robotics: Science and Systems (RSS)*, 2012.

[Cra89] J.J. Craig. Introduction to robotics. mechanics and control. series in electrical and computer engineering: Control engineering, 1989. 126

[CS92] L.S. Chou and S.M. Song. Geometric work of manipulators and path planning based on minimum energy consumption. *Journal of Mechanical Design*, 114, 1992. 56

[CSC11] A. Cherubini, F. Spindler, and F. Chaumette. A redundancy-based approach for visual navigation with collision avoidance. In *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 8–15, 2011. 24

[CSG89] R. Colbaugh, H. Seraji, and KL Glass. Obstacle avoidance for redundant robots using configuration control. *Journal of Robotic Systems*, 6(6):721–744, 1989. 22

[CSL05] R. Cupec, G. Schmidt, and O. Lorch. Experiments in vision-guided robot walking in a structured scenario. In *IEEE International Symposium on Industrial Electronics*, volume 4, pages 1581–1586, 2005. 167

[DA10] H. Dang and P.K. Allen. Robot learning of everyday object manipulations via human demonstration. In *IEEE International Conference on Intelligent Robots and Systems*, 2010. 177

[DAAAM08] R. Dahmouche, O. Ait-Aider, N. Andreff, and Y. Mezouar. High-speed pose and velocity measurement from vision. In *IEEE International Conference on Robotics and Automation*, pages 107–112, 2008. 133, 225

[DAM12] A. Dingli, D. Attard, and R. Mamo. Turning homes into low-cost ambient assisted living environments. *International Journal of Ambient Computing and Intelligence (IJACI)*, pages 1–23, 2012. 144

[DAMM09] R. Dahmouche, N. Andreff, Y. Mezouar, and P. Martinet. 3d pose and velocity visual tracking based on sequential region of interest acquisition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5426–5431, 2009. 133

[DBDS97] S. Dutre, H. Bruyninckx, and J. De Schutter. The analytical jacobian and its derivative for a parallel manipulator. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 2961–2966, 1997. 134

[DC02] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, july 2002. 151

[DLOG06] A. De Luca, G. Oriolo, and P.R. Giordano. Kinematic modeling and redundancy resolution for nonholonomic mobile manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1867–1873, 2006. 26

[DMB93] K.L. Doty, C. Melchiorri, and C. Bonivento. A Theory of Generalized Inverses Applied to Robotics. *The International Journal of Robotics Research*, 12(1):1–19, 1993. 126

[DN90] R.C. Dorf and S.Y. Nof. *Concise International Encyclopedia of Robotics: Applications and Automation*. John Wiley & Sons, Inc., 1990. 4

[DW12] S. Dong and B. Williams. Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. *International Journal of Social Robotics*, pages 1–12, 2012. 9

[DWBS96] C.C. De Wit, G. Bastin, and B. Siciliano. *Theory of robot control*. Springer-Verlag New York, Inc., 1996. 10

[DZ85] P. Dauchez and R. Zapata. Coordinated control of two cooperative manipulators: The use of a kinematic model. In *15th International Symposium Inustrial Robots*, pages 641–648, 1985. 2

[ECR92] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992. 112

[EMS+09] P. Evrard, N. Mansard, O. Stasse, A. Kheddar, T. Schauß, C. Weber, A. Peer, and M. Buss. Intercontinental, multimodal, wide-range tele-cooperation using a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5635–5640, 2009. 177

[FA12] M. Fairbank and E. Alonso. Efficient calculation of the gauss-newton approximation of the hessian matrix in neural networks. *Neural Computation*, 24(3):607–610, 2012. 133, 225

[FBE+09] F. Faber, M. Bennewitz, C. Eppner, A. Gorog, C. Gonsior, D. Joho, M. Schreiber, and S. Behnke. The humanoid museum tour guide robotinho. In *18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 891–896, 2009. 168

[FC09] R.T. Fomena and F. Chaumette. Improvements on visual servoing from spherical targets using a spherical projection model. *IEEE Transactions on Robotics*, 25(4):874–886, 2009. 114

[FCM00] G. Flandin, F. Chaumette, and E. Marchand. Eye-in-hand/eye-to-hand cooperation for visual servoing. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2741–2746, 2000. 112

[FK75] Fujii and Kurono. Coordinated computer control of a pair of manipulators. In *4th World congress of the international federation for theory of machines and mechanisms (IFToMM)*, pages 411–417, 1975. 2

[FK97] R. Featherstone and O. Khatib. Load independence of the dynamically consistent inverse of the jacobian matrix. *International Journal of Robotics Research*, 16(2):168–170, 1997. 124, 125

[Fle60] TF Fletcher. The undersea mobot. *Report on Nuclear Electronics Laboratory of Hughes Aircraft Company*, 1960. 3

[FLM91] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell. Weighted selection of image features for resolved rate visual feedback control. *IEEE Transactions on Robotics and Automation*, 7(1):31–47, 1991. 113

[FP03] D.A. Forsyth and J. Ponce. Computer vision: A modern approach. *Prentice Hall series in artificial intelligence*, 2003. 114

[FS96] G. Field and Y. Stepanenko. Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2755–2760, 1996. 57

[GFMGS08] C. Galindo, J.A. Fernández-Madrigal, J. González, and A. Saffiotti. Robot task planning using semantic maps. *International Journal of Robotics and Autonomous Systems*, 56(11):955–966, 2008. 150

[GH02] X. Guo and Z. Huang. Analysis for acceleration performance indices of parallel robots. *Zhongguo Jixie Gongcheng/China Mechanical Engineering*, 13(24):2087–2091, 2002. 56

[GH07] N.R. Gans and S.A. Hutchinson. Stable visual servoing through hybrid switched-system control. *IEEE Transactions on Robotics*, 23(3):530–540, 2007. 24, 119

[GHB+09] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, and Brice Marnier et al. Mechatronic design of nao humanoid. In *IEEE Int. Conf. on Robotics and Automation*, pages 769–774, may 2009. 141, 227

[GHRL09] C. Graf, A. Härtl, T. Röfer, and T. Laue. A robust closed-loop gait for the standard platform league humanoid. In *4th Workshop on Humanoid Soccer Robots*, pages 30–37, 2009. 142

[GLW06] X. Guo, S. Liu, and Z. Wang. Analysis for dynamics performance indices of 4-rr (rr) r parallel mechanism. *International Journal of Innovative Computing Information and Control*, 2(4):849–862, 2006. 56

[Goe52] R.C. Goertz. Fundamentals of general-purpose remote manipulators. *Nucleonics (US) Ceased publication*, 10, 1952. 3

[Gos90] C. Gosselin. Stiffness mapping for parallel manipulators. *IEEE Transactions on Robotics and Automation*, 6(3):377–382, 1990. 56

[Gut02] Jens-Steffen Gutmann. Markov-kalman localization for mobile robots. In *6th International Conference on Pattern Recognition*, pages 601–604, 2002. 167

[HAMM07] H. Hadj-Abdelkader, Y. Mezouar, and P. Martinet. Decoupled visual servoing from a set of points imaged by an omnidirectional camera. In *International Conference on Robotics and Automation*, pages 1697–1702. IEEE, 2007. 111

[Has93] K. Hashimoto. *Visual servoing: real-time control of robot manipulators based on visual sensory feedback*, volume 7. World Scientific Publishing Company Incorporated, 1993. 112

[Hay86] S. Hayati. Hybrid position/force control of multi-arm cooperating robots. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 82–89, 1986. 2

[HG05] O. Härkegård and S.T. Glad. Resolving actuator redundancy-optimal control vs. control allocation. *Automatica*, 41(1):137–144, 2005. 22

[HHC96] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996. 111, 112, 114

[HHS07] P. Hsu, J. Hauser, and S. Sastry. Dynamic control of redundant manipulators. *Journal of Robotic Systems*, 6(2):133–148, 2007. 134

[HIA98] K. Hosoda, K. Igarashi, and M. Asada. Adaptive hybrid control for visual and force servoing in an unknown environment. *IEEE Transactions on Robotics and Automation*, 5(4):39–43, 1998. 24

[HJ06] A.H.A. Hafez and C.V. Jawahar. Probabilistic integration of 2d and 3d cues for visual servoing. In *9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, dec. 2006. 27

[HK09] J. Haverinen and A. Kemppainen. Global indoor self-localization based on the ambient magnetic field. *International Journal of Robotics and Autonomous Systems*, 57(10):1028–1035, oct. 2009. 167

[Hor86] B.K.P. Horn. Robot vision, 1986. 114

[Hor06] T.N. Hornyak. *Loving the machine: the art and science of Japanese robots*. Kodansha Amer Inc, 2006. 5

[Hou05] A. Hourtash. The kinematic hessian and higher derivatives. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 169–174, 2005. 133, 225

[HRP] HRP-2. Humanoid robot hrp-2 (promet). http://global.kawada.jp/mechatronics/hrp2.html. Last visited on october 2012. 140, 230

[HS87] J. Hollerbach and K. Suh. Redundancy resolution of manipulators through torque optimization. *IEEE Transactions on Robotics and Automation*, 3(4):308–316, 1987. 124

[HSE+11] K. Houston, A. Sieber, C. Eder, O. Vittorio, A. Menciassi, and P. Dario. A teleoperation system with novel haptic device for micro-manipulation. *International Journal of Robotics and Automation*, 26(3), 2011. 177

[HT01] R. Hirose and T. Takenaka. Development of the humanoid robot asimo. *Honda R&D Technical Review*, 13(1):1–6, 2001. 6

[HWB10] A. Hornung, K. M. Wurm, and M. Bennwitz. Humanoid robot localization in complex indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1690–1695, oct. 2010. 167, 168

[ISMO09] J. Ido, Y. Shimizu, Y. Matsumoto, and T. Ogasawara. Indoor navigation for a humanoid robot using a view sequence. *International Journal of Robotic Research*, 28(2):315–325, 2009. 167

[Jim12] P. Jiménez. Survey on model-based manipulation planning of deformable objects. *Robotics and Computer-Integrated Manufacturing*, 28(2):154–163, April 2012. 8

[JK09] A. Jain and C.C. Kemp. Pulling open novel doors and drawers with equilibrium point control. In *IEEE-RAS International Conference on Humanoid Robots*, pages 498–505, 2009. 177

[JK10] A. Jain and C. Kemp. El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots*, 28:45–64, 2010. 167

[JSDW11] F. Janabi-Sharifi, L. Deng, and W.J. Wilson. Comparison of basic visual servoing methods. *IEEE/ASME Transactions on Mechatronics*, 16(5):967–983, 2011. 112

[JSW97] F. Janabi-Sharifi and WJ Wilson. Automatic selection of image features for visual servoing. *IEEE Transactions on Robotics and Automation*, 13(6):890–903, 1997. 113

[KB87] C.A. Klein and B.E. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987. 56

[KC+97] W. Khalil, D. Creusot, et al. Symoro+: a system for the symbolic modelling of robots. *Robotica*, 15:153–161, 1997. 134, 184, 225

[KC02a] D. Kragic and HI Christensen. Model based techniques for robotic servoing and grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 299–304, 2002. 9

[KC02b] D. Kragic and H.I. Christensen. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory*, 15, 2002. 111, 112

[KC11a] O. Kermorgant and F. Chaumette. Combining ibvs and pbvs to ensure the visibility constraint. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2849–2854, sept. 2011. 27, 54, 213

[KC11b] O. Kermorgant and F. Chaumette. Multi-sensor data fusion in sensor-based control: application to multi-camera visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 4518–4523, 2011. 25

[KCF+12] Chih-Hung King, Tiffany L. Chen, Zhengqin Fan, Jonathan D. Glass, and Charles C. Kemp. Dusty: an assistive mobile manipulator that retrieves dropped objects for people with motor impairments. *Disability and Rehabilitation: Assistive Technology*, 7(2):168–179, 2012. 167

[KD04] W. Khalil and E. Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004. 1

[KE12] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors (Basel, Switzerland)*, 12(2):1437–1454, 2012. 144

[KETJ07] C.C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):20–29, 2007. 9

[Kha87] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987. 11, 27, 104, 124, 126, 135

[KJLH11] S. Kock, H. Jerregård, I. Lundberg, and M. Hedelind. A concept for scalable, flexible assembly automation-a technology study on a harmless dual-armed robot from abb. In *International Symposium on Assembly and Manufacturing*, pages 25–27, 2011. 4

[KK68] G.A. Korn and T.M. Korn. Mathematical handbook for scientist and engineers, 1968. 134

[KK86] W Khalil and J Kleinfinger. A new geometric notation for open and closed-loop robots. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1174–1179. IEEE, 1986. 52, 212

[KKK+02] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics platform for hrp. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2431–2436, 2002. 6

[KMC04] R. Kelly, J. Moreno, and R. Campa. Visual servoing of planar robots via velocity fields. In *43rd IEEE Conference on Decision and Control*, volume 4, pages 4028–4033, 2004. 132

[KOH+06] R. Konietschke, T. Ortmaier, U. Hagn, G. Hirzinger, and S. Frumento. Kinematic design optimization of an actuated carrier for the dlr multi-arm surgical system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4381–4387, oct. 2006. 3

[KSL07] C. S. Kallie, P. R. Schrater, and G. E. Legge. Variability in stepping direction explains the veering behavior of blind walkers. *Journal of Experimental Psychology: Human Perception and Performance*, 33(1), 2007. 172, 229

[KSS11] J. Krüger, G. Schreck, and D. Surdilovic. Dual arm robot for flexible and cooperative assembly. *CIRP Annals-Manufacturing Technology*, 2011. 4

[KW88] K. Kazerounian and Z. Wang. Global versus local optimization in redundancy resolution of robotic manipulators. *The International Journal of Robotics Research*, 7(5):3–12, 1988. 130

[LCK12] P. Long, S. Caro, and W. Khalil. Control of a Lower Mobility Dual Arm System. In *Robot Control*, volume 10, pages 307–312, Dubrovnik, Croatie, September 2012. 21

[Lie77] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on System Man and Cybernetics*, pages 868–871, 1977. 13, 29, 30, 202

[LPB95] ZC Lin, RV Patel, and CA Balafoutis. Impact reduction for redundant manipulators using augmented impedance control. *Journal of Robotic Systems*, 12(5):301–313, 1995. 22

[LWP80] J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980. 11, 104

[LZ11] K. Li and Y. Zhang. Minimum-effort redundancy resolution of robot manipulators unified by quadratic programming. In *IEEE International Conference on Automation and Logistics (ICAL)*, pages 108–113, 2011. 124

[Ma95] S. Ma. Real-time algorithm for quasi-minimum energy control of robotic manipulators. In *21st International Conference on Industrial Electronics, Control, and Instrumentation (IECON)*, volume 2, pages 1324–1329. IEEE, 1995. 57

[Mal04] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1843–1848, 2004. 115

[Man06] N. Mansard. *Enchainement de taches robotiques*. PhD thesis, IRISA, Rennes, 2006. 36, 82

[MBG96] P. Martinet, F. Berry, and J. Gallice. Use of first derivative of geometric features in visual servoing. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3413–3419, 1996. 133

[MC02] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, 2002. 119

[MC04] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 992–997, 2004. 40

[MC09] N. Mansard and F. Chaumette. Directional redundancy for robot control. *IEEE Transactions on Automatic Control*, 54(6):1179–1192, 2009. 29, 32, 34

[MC10] M. Marey and F. Chaumette. A new large projection operator for the redundancy framework. In *IEEE International Conference on Robotics and Automation*, pages 3727–3732, 2010. 29, 37, 187

[MCB99] E. Malis, F. Chaumette, and S. Boudet. 21/2d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999. 112, 116, 118, 224

[McC86] N. McClamroch. Singular systems of differential equations as dynamic models for constrained robot systems. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 21–28, 1986. 2

[MDC+09] L. Marques, J. Dinis, A.P. Coimbra, M.M. Crisóstomo, and J.P. Ferreira. 3d hyper-redundant robot. In *11th Spanish-Portuguese Conference on Electrical Engineering*, Portugal, 2009. Institute of Systems and Robotics Electrical and Computer Engineering. 29

[MDGD97] P. Martinet, N. Daucher, J. Gallice, and M. Dhome. Robot control using monocular pose estimation. *Worshop on New Trends in Image-based Robot Servoing, IROS*, 97:1–12, 1997. 112

[MG91] J.P. Merlet and C. Gosselin. Nouvelle architecture pour un manipulateur parallèle à six degrés de liberté. *Mechanism and machine theory*, 26(1):77–90, 1991. 134

[MGK+96] P. Martinet, J. Gallice, D. Khadraoui, et al. Vision based control law using 3d visual features. In *WAC proceedings*, volume 96, pages 497–502. Citeseer, 1996. 112

[MI87] W. Miller III. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Transactions on Robotics and Automation*, 3(2):157–165, 1987. 177

[MK88] A.A. Maciejewski and C.A. Klein. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems*, 5(6):527–552, 1988. 32

[MKC10] R. Mebarki, A. Krupa, and F. Chaumette. 2d ultrasound probe complete guidance by visual servoing using image moments. *IEEE Transactions on Robotics*, 26(2):296–306, 2010. 111

[MKK09] N. Mansard, O. Khatib, and A. Kheddar. A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, 25(3):670–685, 2009. 29

[MKP+02] F. Marquet, S. Krut, F. Pierrot, et al. Enhancing parallel robots accuracy with redundant sensors. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4114–4119, 2002. 22

[MLS+00] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot. Explicit incorporation of 2d constraints in vision based control of robot manipulators. *Experimental Robotics VI*, pages 99–108, 2000. 26, 118

[MP95] N. Mandal and S. Payandeh. Control strategies for rodotic contact tasks: An experimental study. *Journal of Robotic Systems*, 12(1):67–92, 1995. 9

[MP97] S.A.A. Moosavian and E. Papadopoulos. Control of space free-flyers using the modified transpose jacobian algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1500–1505, 1997. 126

[MP05] G.L. Mariottini and D. Prattichizzo. Egt for multiple view geometry and visual servoing: robotics vision with pinhole and panoramic cameras. *IEEE Transactions on Robotics and Automation*, 12(4):26–39, 2005. 154

[MP07] S.A.A. Moosavian and E. Papadopoulos. Modified transpose jacobian control of robotic systems. *Automatica (Journal of IFAC)*, 43(7):1226–1233, 2007. 126

[MRC09] N. Mansard, A. Remazeilles, and F. Chaumette. Continuity of varying-feature-set control laws. *IEEE Transactions on Automatic Control*, 54(11):2493–2505, 2009. 26

[MSC05] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Transactions on Robotics and Automation*, 12(4):40–52, 2005. 154

[MSCY07] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi. Visually-guided grasping while walking on a humanoid robot. In *IEEE International Conference on Robotics and Automation*, pages 3041–3047, 2007. 177

[MSEK09] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009. 29, 45

[MSK96] H. Mochiyama, E. Shimemura, and H. Kobayashi. Control of serial rigid link manipulators with hyper degrees of freedom: shape control by a homogeneously decentralized scheme and its experiment. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2877–2882, 1996. 22, 206

[Nak90] Y. Nakamura. *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990. 19, 25, 26, 29

[NH86] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986. 31

[NHY87] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987. 129

[NOIK74] E. Nakano, S. Ozaki, T. Ishida, and I. Kato. Cooperational control of the anthropomorphous manipulator 'melarm'. In *4th Int. Symp. Ind. Robots*, pages 251–260, 1974. 2

[NS04] N. Nitzsche and G. Schmidt. A mobile haptic interface mastering a mobile teleoperator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3912–3917, 2004. 177

[NWM11] Bingbing N., Gang W., and P. Moulin. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1147–1153, nov. 2011. 144

[NYC+99] U. Neumann, S. You, Y. Cho, J. Lee, and J. Park. Augmented reality tracking in natural environments. In *International Symposium on Mixed Realities*, volume 24. Ohmsha Ltd and Springer-Verlag Tokyo, 1999. 153

[NYK+06] H. Nakai, M. Yamataka, T. Kuga, S. Kuge, H. Tadano, H. Nakanishi, M. Furukawa, and H. Ohtsuka. Development of dual-arm robot with multi-fingered hands. In *15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 208–213, 2006. 4

[NZ02] B. Nemec and L. Zlajpah. Force control of redundant robots in unstructured environment. *Industrial Electronics*, 49(1):233–240, 2002. 125

[OA99] P.Y. Oh and P.K. Allen. Performance of a partitioned visual feedback controller. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 275 –280, 1999. 24

[OCB05] WS Owen, EA Croft, and B. Benhabib. Acceleration and torque redistribution for a dual-manipulator system. *IEEE Transactions on Robotics*, 21(6):1226–1230, 2005. 124

[OHB10] S. Oflwald, A. Hornung, and M. Bennewitz. Learning reliable and efficient navigation with a humanoid. In *IEEE International Conference on Robotics and Automation*, pages 2375–2380, 2010. 167

[OJGJ02] J. Okamoto Jr and V. Grassi Jr. Visual servo control of a mobile robot using omnidirectional vision. *Mechatronics*, pages 413–422, 2002. 111

[O'N02] K.A. O'Neil. Divergence of linear acceleration-based redundancy resolution schemes. *IEEE Transactions on Robotics and Automation*, 18(4):625–631, 2002. 124

[Ope] OpenHRP. Openhrp3 official site. `http://www.openrtp.jp/openhrp3/en/about.html`. Last visited on october 2012. 145

[PL91] E.D. Pohl and H. Lipkin. A new method of robotic rate control near singularities. In *IEEE International Conference on Robotics and Automation*, pages 1708–1713, 1991. 134

[PMB+09] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *IEEE International Conference on Robotics and Automation*, pages 2250–2257, 2009. 167

[PMdPL07] M. Prats, P. Martinet, A.P. del Pobil, and S. Lee. Vision force control in task-oriented grasping and manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1320–1325, 2007. 10

[PMU+07] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots*, 24(1):1–12, 2007. 126

[PNK92] NP Papanikolopoulos, B. Nelson, and PK Khosla. Full 3-d tracking using the controlled active vision paradigm. In *IEEE International Symposium on Intelligent Control*, pages 267–274, 1992. 27

[PPPK09] C. Park, K. Park, D.I.L. Park, and J.H. Kyung. Dual arm robot manipulator and its easy teaching system. In *IEEE International Symposium on Assembly and Manufacturing*, pages 242–247, 2009. 4

[PSdP08] M. Prats, P.J. Sanz, and A.P. del Pobil. A sensor-based approach for physical interaction based on hand, grasp and task frames. In *Manipulation workshop in robotics science and systems*, 2008. 10

[PSdP10] M. Prats, P.J. Sanz, and A.P. del Pobil. A framework for compliant physical interaction. *Autonomous Robots*, 28(1):89–111, 2010.

[PSdP11] M. Prats, P.J. Sanz, and A.P. del Pobil. The advantages of exploiting grasp redundancy in robotic manipulation. In *5th International Conference on Automation, Robotics and Applications (ICARA)*, pages 334–339. IEEE, 2011. 20

[QCG+09] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *IEEE International Conference on Robotics and Automation - Workshop on Open Source Software*, 2009. 145

[QM10] M. Quinlan and R. Middleton. Multiple model kalman filters: A localization technique for robocup soccer. *RoboCup 2009: Robot Soccer World Cup XIII*, pages 276–287, 2010. 167

[RASR11] A.P. Raju, J. Amarnath, D. Subbarayudu, and MR Reddy. First order derivative computation of jacobian matrices in load flow algorithms using automatic differentiation. In *14th International Symposium on Electrets (ISE)*, pages 81–82. IEEE, 2011. 133, 225

[RC11] R.B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation*, pages 1–4, may 2011. 145, 157

[RL94] D. Reznik and V. Lumelsky. Sensor-based motion planning in three dimensions for a highly redundant snake robot. *Advanced robotics*, 9(3):255–280, 1994.

[Rob] Aldebaran Robotics. Walk control - NAO Software 1.12.5 documentation. http://www.aldebaran-robotics.com/documentation/index.html. Last visited on september 2012. 171, 173

[Ros94] M.E. Rosheim. *Robot evolution: the development of anthrobotics*. Wiley-Interscience, 1994. 5

[Ros06] M.E. Rosheim. *Leonardo's lost robots*. Springer Verlag, 2006. 5

[SBG+08] C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *IEEE International Conference on Robotics and Automation*, pages 3194–3199, 2008. 167

[SC82] J.K. Salisbury and J.J. Craig. Articulated hands force control and kinematic issues. *The International Journal of Robotics Research*, 1(1):4–17, 1982. 55, 215

[SC90] H. Seraji and R. Colbaugh. Improved configuration control for redundant robots. *Journal of Robotic Systems*, 7(6):897–928, 1990. 22

[Sen99] K. Senda. Quasioptimal control of space redundant manipulators. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1877–1885, Portland, OR, 1999. 124

[Ser91] H. Seraji. Task options for redundancy resolution using configuration control. In *30th IEEE Conference on Decision and Control*, pages 2793–2798, 1991. 22

[SG96] K.B. Shimoga and A.A. Goldenberg. Soft robotic fingertips. *The International Journal of Robotics Research*, 15(4):335–350, 1996. 9

[SG07] W. Shen and J. Gu. Multi-criteria kinematics control for the pa10-7c robot arm with robust singularities. In *IEEE International Conference on Robotics and Biomimetics*, pages 1242–1248, 2007. 27

[SHS08] A. Sanfeliu, N. Hagita, and A. Saffiotti. Network robot systems. *International Journal of Robotics and Autonomous Systems*, 56(10):793–797, 2008. 143

[SHV06] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons Hoboken, NJ, 2006. 6

[SK08] B. Siciliano and O. Khatib, editors. *Handbook of Robotics*. Springer, 2008. 2

[SKN+12] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual arm manipulation–A survey. *International Journal of Robotics and Autonomous Systems*, July 2012. 4

[SLBE91] C. Samson, M. Le Borgne, and B. Espiau. *Robot control: the task function approach*, volume 22. Clarendon Press Oxford, England, 1991. 11, 29, 104, 112, 171

[SLCY05] Z. SHI, Y. LUO, H. CHEN, and M. YE. On global performance indices of robotic mechanisms. *Robot*, 27(5):420–424, 2005. 56

[SOC95] J. Seng, K.A. O'Neil, and YC Chen. Escapability of singular configuration for redundant manipulators via self-motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 78–83, 1995. 134

[SOH12] V. Salvucci, S. Oh, and Y. Hori. Analysis of actuator redundancy resolution methods for bi-articularly actuated robot arms. In *12th IEEE International Workshop on Advanced Motion Control*, pages 1–6, 2012. 22

[SS88] L. Sciavicco and B. Siciliano. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 4(4):403–410, 1988. 32

[SS91] B. Siciliano and J.J.E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *5th International Conference on Advanced Robotics*, volume Robots in Unstructured Environments, pages 1211–1216. IEEE, 1991. 42, 129, 130

[SS96] L. Sciavicco and B. Siciliano. *Modeling and control of robot manipulators*. McGraw-Hill Companies, 1996. 10

[SS98] SFP Saramago and V. Steffen. Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system. *Mechanism and machine theory*, 33(7):883–894, 1998. 57

[SS11] E.E. Stone and M. Skubic. Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 71–77, may 2011. 144

[SSV09] B. Siciliano, L. Sciavicco, and L. Villani. *Robotics: modelling, planning and control*. Springer Verlag, 2009.

[SSY+06] N. Sian, T. Sakaguchi, K. Yokoi, Y. Kawai, and K. Maruyama. Operating humanoid robots in human environments. In *RSS Workshop: Manipulation for Human Environments*. Citeseer, 2006. 8

[SVVY09] O. Stasse, B. Verrelst, B. Vanderborght, and K. Yokoi. Strategies for humanoid robots to dynamically walk over large obstacles. *IEEE Transactions on Robotics*, 25(4):960–967, 2009. 150

[SY87] J.J.E. Slotine and DR Yoerger. A rule-based inverse kinematics algorithm for redundant manipulators. *International Journal of Robotics and Automation*, 2(2):86–89, 1987. 32

[Tay79] R.H. Taylor. Planning and execution of straight line manipulator trajectories. *IBM Journal of Research and Development*, 23(4):424–436, 1979. 29

[TBY88] TJ Tarn, AK Bejczy, and X. Yun. New nonlinear control algorithms for multiple robot arms. *IEEE Transactions on Aerospace and Electronic Systems*, 24(5):571–583, 1988. 2

[TFBD01] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots, 2001. 167

[TFM+08] R. A. TÈllez, F. Ferro, D. Mora, D. Pinyol, and D. Faconti. Autonomous humanoid navigation using laser and odometry data. In *IEEE-RAS International Conference on Humanoid Robots*, pages 500–506. IEEE, 2008. 168

[TISH74] K. Takase, H. Inoue, K. Sato, and S. Hagiwara. The design of an articulated manipulator with torque control ability. In *4th Int. Symp. Ind. Robots*, pages 261–270, 1974. 2

[TKD11] C. Toglia, F. Kennedy, and S. Dubowsky. Cooperative control of modular space robots. *Autonomous Robots*, 31(2):209–221, 2011. 22

[TKN06] S. Thompson, S. Kagami, and K. Nishiwaki. Localisation for autonomous humanoid navigation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 13–19. IEEE, 2006. 168

[TPD09] F. Torres, S. Puente, and C. DÌaz. Automatic cooperative disassembly robotic system: Task planner to distribute tasks among robots. *Control Engineering Practice*, 17(1):112–121, 2009. 3

[UD92] M. Uchiyama and P. Dauchez. Symmetric kinematic formulation and non-master/slave coordinated control of two-arm robots. *Advanced Robotics*, 7(4):361–383, 1992. 2

[Ued] Ryohei Ueda. Tracking 3d objects with point cloud library. `http://pointclouds.org/news`. Last visited on june 2012. 146, 157

[VAC+02] M. Vincze, M. Ayromlou, S. Chroust, M. Zillich, W. Ponweiser, and D. Legenstein. Dynamic aspects of visual servoing and a framework for real-time 3d vision for robotics. *Sensor Based Intelligent Robots*, pages 101–121, 2002. 123

[Wam86] C.W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):93–101, 1986. 31

[WE84] W.A. Wolovich and H. Elliott. A computational technique for inverse kinematics. In *23rd IEEE Conference on Decision and Control*, volume 23, pages 1359–1363, 1984. 32

[WES87] B. Wittenmark, R.J. Evans, and Y.C. Soh. Constrained pole-placement using transformation and lq-design. *Automatica*, 23(6):767–769, 1987. 27

[Whi87] D.E. Whitney. Historical perspective and state of the art in robot force control. *The International Journal of Robotics Research*, 6(1):3–14, 1987. 4

[WMB03] S. Wakabayashi, D.F. Magruder, and W. Bluethmann. Test of operator endurance in the teleoperation of an anthropomorphic hand. In *7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003. 9

[WR94] DM Weer and S.M. Rock. Experiments in object impedance control for flexible objects. In *IEEE International Conference on Robotics and Automation*, pages 1222–1227, 1994. 9

[WSN87] L. Weiss, A. Sanderson, and C. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Transactions on Robotics and Automation*, 3(5):404–417, 1987. 12, 111, 112, 115

[WW78] D.J. Wilde and DJ Wilde. *Globally optimal design*. Wiley, 1978. 134

[WWHB96] W.J. Wilson, CC Williams Hulls, and G.S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996. 112

[XLTP09] W.F. Xie, Z. Li, X.W. Tu, and C. Perron. Switching control of image-based visual servoing with laser pointer in robotic manufacturing systems. *IEEE Transactions on Industrial Electronics*, 56(2):520–529, 2009. 119

[XTB96] N. Xi, T.J. Tarn, and A.K. Bejczy. Intelligent planning and control for multirobot coordination: An event-based approach. *IEEE Transactions on Robotics and Automation*, 12(3):439–452, 1996. 104, 222

[XZW10] J. Xiang, C. Zhong, and W. Wei. General-weighted least-norm control for redundant manipulators. *IEEE Transactions on Robotics*, 26(4):660–669, 2010. 26

[YNS95] Y. Yamada, S. Nagamatsu, and Y. Sato. Development of multi-arm robots for automobile assembly. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2224–2229, 1995. 4

[Yos84] T. Yoshikawa. Analysis and control of robot manipulators with redundancy. In *Robotics research: the first international symposium*, pages 735–747. Mit Press Cambridge, MA, 1984. 55, 215

[Yos90] T. Yoshikawa. *Foundations of robotics: analysis and control*. The MIT Press, 1990. 10, 55, 56, 215

[YTU99] WK Yoon, Y. Tsumaki, and M. Uchiyama. An experimental system for dual-arm robot teleoperation in space with concepts of virtual grip and ball. In *International Conference on Advanced Robotics*, pages 225–230, Tokyo, Japan, 1999. 4

[ZCYZ12] Y.N. Zhang, B.H. Cai, J.P. Yin, and L. Zhang. Two/infinity norm criteria resolution of manipulator redundancy at joint-acceleration level using primal-dual neural network. *Asian Journal of Control*, 2012. 124

[ZLWS03] J. Zhang, R. Lumia, J. Wood, and G. Starr. Delay dependent stability limits in high performance real-time visual servoing systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 485–491, 2003. 123

Contributions à l'enchaînement des tâches et à la résolution de la redondance
Application aux robots humanoïdes et multi-bras

Résumé: La redondance est présente dans toutes les plateformes robotiques, elle est soit intrinsèque et explicite dans l'architecture mécanique du robot, ou implicite et n'apparaît que lors de l'exécution de certaines tâches. Cette thèse identifie, classifie les différents types de redondances et traite leur résolution cinématique lors de l'application de plusieurs tâches simultanées sur un système redondant avec une priorité spécifique.

Une approche de multi-points de contrôle et des formalismes de résolution de la redondance ont été développés et validés par l'application de plusieurs tâches de service et d'assistance sur un système multi-bras pour la découpe de la viande et deux robots humanoïdes HRP-2 et Nao. En utilisant différents types de capteurs, notamment des systèmes de vision embarqués et déportés, plusieurs techniques ont été implémentées sur ces plateformes robotiques pour appliquer, en simulation et en temps réel, les tâches désirées (localisation, saisie, navigation, visibilité ..) tout en respectant les contraintes du système (évitement de collision, occultation, butées articulaires, ...).

Mots Clés: Robot redondant, Résolution de la redondance, Système multi-bras, Commande référencée capteurs, Multi-points de contrôle, Manipulation, Localisation, Nao, HRP-2.

Contributions in task sequencing and redundancy resolution
Application to humanoid and multi-arm robots

Abstract: Redundancy is present in all the robotic platforms; it is either intrinsic and explicit in the robot's mechanical architecture, or implicit and appears only when applying specific tasks. This thesis identifies and classifies the different types of redundancies; it also addresses their kinematic resolution when applying several simultaneous and prioritized tasks on a general redundant system.

Multi-control points approach and redundancy resolution formalisms were developed and validated by the application of several industrial, service and assistive tasks on three different platforms: a multi-arm system for meat cutting and two humanoid robots (HRP-2 and Nao). Using various types of embedded and external vision sensors, several techniques were integrated into these robotic platforms to apply, in simulation and real-time, the desired scenarios (localization, grasping, navigation, visibility, ...) while taking into account the system constraints (collision, occlusion, joint limits avoidance, ...).

Keywords: Redundant robot, Redundancy resolution, Multi-arm system, Sensor-based control, Multi-control points, Manipulation, Localization, Nao, HRP-2.

Discipline: Sciences de l'Ingénieur