Reactive control and sensor fusion for mobile manipulators in human robot interaction

Wuwei He

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse III Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

Informatique et Robotique

Présentée et soutenue par :
Wuwei HE

le : vendredi 4 octobre 2013

Titre :

Reactive Control and Sensor Fusion for Mobile Manipulators in Human Robot Interaction

---

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
LAAS-CNRS

Directeur(s) de Thèse :

M. Daniel SIDOBRE

Rapporteurs :

M. Philippe BIDAUD, professeur
M. Jean-Pierre GAZEAU, ingénieur de recherche

Membre(s) du jury :

M. Rachid ALAMI, directeur de recherche
M. Philippe FRAISSE, professeur
M. Patrick DANES, professeur

UNIVERSITÉ TOULOUSE III - PAUL SABATIER
ÉCOLE DOCTORALE SYSTÈMES

THÈSE

en vue de l'obtention du

Doctorat de l'Université de Toulouse
délivré par l'Université Toulouse III Paul Sabatier

Spécialité: Informatique et Robotique
(Computer Science and Robotics)
présentée et soutenue publiquement le 4 Octobre 2013

Reactive Control and Sensor Fusion for Mobile Manipulators in Human Robot Interaction

Wuwei HE

Préparée au Laboratoire d'Analyse et d'Architecture des Systèmes
sous la direction de M. Daniel SIDOBRE

Jury

M. Philippe BIDAUD	Rapporteur
M. Jean-Pierre GAZEAU	Rapporteur
M. Philippe FRAISSE	Examinateur
M. Patrick DANÈS	Examinateur
M. Rachid ALAMI	Examinateur
M. Daniel SIDOBRE	Directeur de Thèse

敬親愛的父母

To my parents

Abstract

In order to share a workspace with humans, a service robot should be able to safely interact within an unstructured environment. In this context, the robot shall adapt its behavior and react to the environment changes and human activities. The robots based on motion planning are not able to adapt fast enough, so we propose a reactive trajectory controller to track targets, react to human activities and prevent event like collisions.

The reliability of the proposed trajectory controller is based on recent fusion techniques to identify movements and detect forces associated to events. We propose to employ a 6D force/torque sensor to estimate the inertial parameters of the manipulated objects, then the parameters are used to complement the visual tracking process and to compute the contact forces between the robot end-effector and the environment. The contact forces are analyzed and classified by using learning techniques to detect different events, such as human grasping the object or collision between the object and the environment.

This work, conducted as part of the European projects DEXMART and SAPHARI, and the ANR projects ASSIST and ICARO, has been integrated and validated on the Jido and the PR2 robot platforms of LAAS-CNRS.

Keywords: Robotics, Trajectory control, Sensor Fusion, Machine Learning, Human-Robot Interaction, Manipulation

Résumé

Afin de partager un espace de travail avec les humains, les robots de services doivent être capable d'interagir dans des environnements peu structurés. Dans ce contexte, le robot doit réagir et adapter son comportement aux évolutions de l'environnement et aux activités des humains.L'utilisation de planificateur de mouvement ne permet pas au robot d'être suffisamment réactif, aussi nous proposons un controleur de trajectoire réactif capable de suivre une cible, de réagir aux changement d'atitudes des humains ou de prévenir les évènements et, en particulier, les collisions.

Pour fiabiliser le contrôleur de trajectoire, nous utilisons des techniques de fusion de données récentes afin d'identifier les mouvements ou de détecter des forces associées à des évènements. Nous proposons d'utiliser un capteur de force six axes pour estimer les paramètres d'inertie des objets manipulés, puis d'utiliser ces paramètres pour compléter le contrôle visuel et calculer les forces de contact entre l'organe terminal du robot et son environnement. L'utilisation de technique d'apprentissage permet d'analyser et de classifier les forces de contact pour détecter différents évènements tels que la saisie de l'objet par un humain ou le contact entre le robot ou l'objet transporté et l'environnement.

Ce travail a été intégré et testé sur les robots jido et PR2 du LAAS-CNRS dans le cadre des projets européens DEXMART et SAPHARI et des projets ANR ASSIST et ICARO.

Mots clés: Robotique, contrôle de trajectoire, fusion de données, apprentissage, interaction homme-robot, manipulation.

Acknowledgement

This Ph.D program at *LAAS-CNRS* has been three years of precious experience for me. During these years, I have learned so much, not only about robotics, but also about how to work and especially how to work in a team. Therefore, I would like to express my gratitude, without trying to make a complete list, to the people who have helped me during these years.

Foremost, I would like to thank Mr. Daniel Sidobre and Processor Rachid Alami to give me this opportunity to work in a prestigious team and in the promising area of robotics. The always right on-the-spot comments and advices of Rachid have been always a powerful push for the work, and the patience and encouragement from Daniel have always been the most helpful. I am very grateful for these years in this group and it has been an invaluable opportunity in my professional career.

A special thanks goes to Mathieu Herrb and Anthony Mallet, our research engineers, for their support and time. When I firstly arrived, my lack of experience with the systems on the robots may have required much patience. And I would like to thank Jerome Manhes for his work on the Bidule.

Then I would like to thank the Ph.D students and Postdocs with whom I have worked. Much appreciation goes to Xavier Broquèrer and Mokhtar Gharbi. They have been the most supportive and with whom I have worked the most for the first year. The time we spent together to test the motion planner on the robot was long but fun. Then there are Mathieu Warnier, Mamoum Gharbi, Amit Kumar Pandey, Jim Mainprice ... with whom we have spent the most stressful and sometimes chaotic time on the robots.

Then I would like to say thanks to the other members of the group who are the best friends one could ever have. Special thanks go also to the friends that I know from *LAAS* and the schools around the *LAAS*: in INSA, SUPAERO and UPS, with whom I have spent the breaks from work on sports and inspiring discussions on everything.

Then, I thank *Jido* and *PR2*, our awesome robots, who have always been very understanding.

Contents

1 Introduction **1**

1.1 Introduction . . . 1
1.2 Motivation . . . 2
1.3 Research Objectives and Contribution . . . 3
1.3.1 Sensor Fusion . . . 3
1.3.2 Learning to Manipulate . . . 5
1.3.3 Sensor-Based Control . . . 5
1.4 Structure of this Manuscript . . . 5
1.5 Publication, Software Development, and Research Projects . . . 6

2 Related Work and Background **9**

2.1 Introduction . . . 10
2.2 Autonomous Mobile Manipulator . . . 10
2.2.1 Robots at *LAAS-CNRS* . . . 10
2.2.2 Software Architecture for Human Robot Interaction . . . 12
2.3 Motion Planning and Control . . . 14
2.3.1 Geometrical Constraints in HRI . . . 15
2.3.2 Robot Motion Representation . . . 15
2.3.3 Path Planning . . . 17
2.3.4 Interpolation . . . 19
2.3.5 Trajectory Generation . . . 20
2.3.6 Grasp Planning . . . 22
2.3.7 Trajectory Control . . . 24
2.4 Sensor Fusion . . . 25
2.4.1 Multisensor Data Fusion . . . 25
2.4.2 Force Sensing . . . 25
2.5 Wavelet and Classification . . . 26
2.6 Conclusion . . . 28

3 Force Vision Sensor Fusion **29**

3.1 Introduction . . . 29

3.2 Force Sensing . . . 31
3.2.1 Inertial Parameters Estimation . . . 31
3.2.2 Excitation Trajectories . . . 34
3.2.3 Online Object Recognition . . . 36
3.2.4 Contact Forces/Torques Computation . . . 37
3.3 Force-Vision Fusion, a Multi-modal Tracking . . . 38
3.3.1 Process Model for Tracking . . . 39
3.3.2 Measurement Models . . . 41
3.4 Nonlinear Kalman Filtering . . . 43
3.5 Simulation and Experimental Results . . . 47
3.6 Conclusion . . . 49

4 Machine Learning for Manipulation **51**
4.1 Introduction . . . 51
4.1.1 Trajectory Control with Continuous Classification . . . 52
4.2 Bidule: a Device for Manipulation Learning . . . 52
4.3 Wavelet Analysis . . . 55
4.3.1 Wavelet Packet Transformation (WPT) . . . 55
4.3.2 Lifting Scheme . . . 57
4.3.3 Feature Formation . . . 59
4.4 Feature Selection . . . 60
4.4.1 Fisher Linear Discriminant Analysis . . . 61
4.4.2 Fisher LDA for Feature Selection . . . 62
4.5 Relevance Vector Machine (RVM) and Classification . . . 64
4.6 Data Acquisition and Results . . . 65
4.6.1 The Experimental Protocol . . . 65
4.6.2 Results and Discussion . . . 66
4.7 Conclusion . . . 70

5 Reactive Trajectory Controller **71**
5.1 Introduction . . . 71
5.2 From Path to Trajectory . . . 72
5.2.1 Basic Concepts of the Trajectory Generation . . . 72
5.2.2 Trajectory Generation From a Given Path . . . 76
5.2.3 Application to Robot Manipulators . . . 79
5.2.4 Planning in the Cartesian Space . . . 81
5.3 Control Primitives . . . 83
5.4 Reactive Trajectory Controller . . . 86
5.4.1 Execution Monitoring . . . 87
5.4.2 Trajectory Control Modes . . . 88

5.5 Results and Comparison . . . 92
5.6 Conclusion . . . 94

6 Conclusion and perspectives **97**
6.1 Conclusion . . . 97
6.1.1 Visual Servoing and Trajectory Based Control . . . 97
6.1.2 Force Sensing and Force Events . . . 98
6.2 Perspectives . . . 98
6.2.1 Sensor Fusion and Learning . . . 98
6.2.2 Trajectory Control . . . 101

A Nonlinear Kalman Filters **103**
A.1 Discrete Kalman Filtering . . . 103
A.2 Extended Kalman Filter . . . 104

B Quaternions and Rotations **107**
B.1 Axis-Angle Representation . . . 107
B.2 Definition of Quaternion . . . 107
B.3 Rotation matrix . . . 109
B.4 Rotations and Compositions . . . 109
B.5 Perturbations and Derivatives . . . 110

C Wavelet Analysis **113**
C.1 Windowed Fourier Transform . . . 113
C.2 Continuous Wavelet Transforms . . . 114
C.3 Discrete Wavelet Transforms . . . 115

D Sparse Kernel Machines **117**
D.1 Support Vector Machine . . . 117
D.1.1 Linear SVM . . . 118
D.1.2 The Kernel Trick and Nonlinear SVM . . . 121
D.2 Relevance Vector Machine . . . 121
D.2.1 Evidence Approximation Theory . . . 121
D.2.2 Evidence Approximation . . . 125

Bibliography **127**

E Résumé en français **137**
E.1 Introduction . . . 137
E.1.1 Motivation and Contribution . . . 137
E.1.2 État de l'Art . . . 140
E.2 Fusion des données de force et de vision . . . 146

E.2.1 Estimation des paramètres d'inertie 146
E.2.2 Fusion force-vision et suivi multi-modal 151
E.2.3 Résultats de simulation et expérimental pour la fusion 154
E.3 Apprendre pour échanger des objets 154
E.3.1 Bidule: un appareil pour apprendre la manipulation 155
E.3.2 Traitement du signal et détection d'événements 157
E.3.3 Sélection des caractéristiques pour la LDA de Fisher 161
E.3.4 Classification et machine à vecteur de pertinence 162
E.4 Acquisition de données et résultats 163
E.4.1 Le protocole expérimental 163
E.4.2 Résultats et interprétations 163
E.4.3 Conclusion 167
E.5 Contrôleur de trajectoire réactif 168
E.5.1 Génération de trajectoire en ligne 168
E.5.2 Primitives de contrôle 168
E.5.3 Contrôleur de trajectoire réactif 171
E.5.4 Modes de contrôle de trajectoire 172
E.5.5 Résults et comparaison 175
E.5.6 Conclusion 177

1 Introduction

The conscious experience of being a subject arises when a single organism learns to enslave itself.
— Thomas Metzinger, *The Ego Tunnel: The Science of the Mind and the Myth of the Self*

Abstract. As an important part of the effort to achieve a socially aware service robot, the challenges of sensor-based control for interactive object manipulation between robot and human are introduced. The work consists in three distinctive parts: force sensor fusion, event detection by force sensor, and reactive trajectory control. The research objectives, contribution and the outline of work are presented. As this work has been a part of several research projects for service robots, some cooperative work in the team is also included.

1.1 Introduction

While conquering the industrial realm for its multiple advantages over a human operator, the robot manipulator has become a commodity even for enterprises which have made their success by a business model based on cheap labor, such that *Foxconn*, an electronics contract manufacturing company, is reported to start to install robot manipulators on their fabrication lines[1]. The tasks that the industrial robots execute are often predefined in a static and structured environment, thus humans are excluded from the workspace.

[1] http://www.reuters.com/article/2011/08/01/us-foxconn-robots-idUSTRE77016B20110801

The researchers in service robotics, on the other hand, have already started to work towards the dream of robots living and working around people and inside human society, which has long been a dream in science fiction, being it novels, films, or comics. Although robots with a personality like *Bender*, or *Gort* exist still only in fictional works, autonomous automated machines (or simply, robots) have quietly begun to enter our home, such as *Hom-bot* of LG, etc. The mobile manipulator as a service robot is not yet available for households to purchase, but is already a reality in the laboratories all around the world. While serving mainly as research platforms, with promising results being published every year, the optimism about service mobile manipulators can hardly be criticized as a children's dream anymore. In fact, having a robot companion seems like a step away for the optimists and the robot fans. They are expected to search and rescue ([Khan 12]), guide people all around in museums ([Yousuf 12]), and help people in hospitals([Devos 12]). And most importantly, socialize and learn from us ([Koenig 10], [Argall 09], [Fong 03]). Amongst many research platforms, one example is *Justin*, a mobile manipulator from DLR, on which two DLR-LightWeight Robot-III arms are mounted [Ott 06b]. The base has four wheels. The other example is PR2, the robot developed by *Willow Garage*. PR2 is fully powered by open source software, with the ROS (Robot Operating System). *LAAS-CNRS* built a platform called *Jido* and developed its own software architecture for all its robots. This software architecture has been adapted for the PR2. We will further present our two stars in the next chapter.

1.2 Motivation

Service robots work in a dynamic and unstructured environment, hence it is impossible to model every action a service robot is to achieve. The robot is expected to behave autonomously, which may raise numerous challenges. Firstly, decision-making taking into account social and ethical codes among people is not an easy task. This level of interaction can be called cognitive([Lemaignan 12]). Secondly, working in a changing environment needs the ability to react. For example, the robot should be able to catch a moving object, react to unpredictable activities carried out by people, and be able to know what is happening at this level. The lower-level interaction, which concerns also a crucial part of security and safety for the agents in the environment and for the robot himself, should also be studied. The emphasize of this thesis is mainly on the lower level of sensor fusion and adaptive control, while being a part of socially aware Human Robot Interaction research projects. More precisely, we focus on the manipulation task of object exchange with human counterparts, should it be the case of giving an object to a person or the case of taking object given by a person. This work does not claim that cognitive interaction and physical interaction are separate matters (which can be seen as a concept derived from the philosophical dualism), instead, the limits of the document are drawn where the contribution ends.

The motivation of this work can be expressed by an analogy: imagine that a person needs to take an object and give it to another person. The environment is dark so they may not always see the object properly (in fact, the vision systems are often unstable and a robot sees things as we see in darkness), and the objects are unknown (heavy or not, which is important for the robot control). And for normal object exchange, the two people are reactive and the manipulation is natural and robust. The giver should predict where the receiver's hands will be, and will release the object when he detects that the object is firmly grasped. Based on a motion planner provided in our team from colleagues' work, this thesis sets out to solve this task. Three major parts are proposed: force sensor fusion, continuous classification for events detection, and a trajectory controller. A scenario is given in Figures 1.1 and 1.2 as an example. In this scenario, the robot shall give an object to a person. The robot needs to grasp the object and give it to the person. When the person grasps the object, the robot should detect it and release the object. The relationships between these different parts can be found in Figure 1.3.

Figure 1.1: A typical object exchange task: robot gives an object to human.

1.3 Research Objectives and Contribution

1.3.1 Sensor Fusion

Working in an unstructured environment means that the robot does not always have a model of the environment. The first challenge is through vision, depth sensors, and others, to model the geometry of the environment and update it online. The geometrical modeling and monitoring is based on the work of the colleagues at *LAAS*. Returning to the object

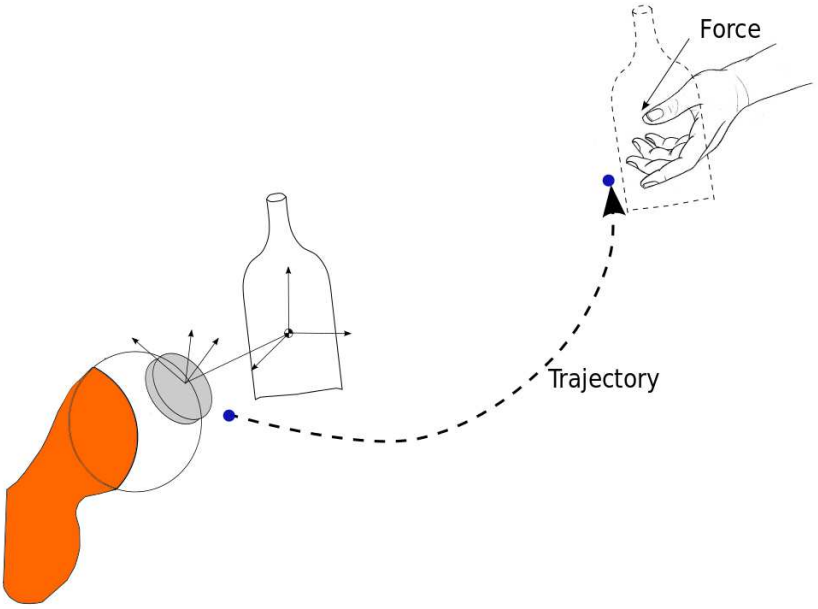


Figure 1.2: When the robot executes the trajectory, it shall estimate the dynamic of the manipulated object, compute the contact forces between the object and human hand, and control the motion to track the exchange point defined in the local frame of the partner's hand.

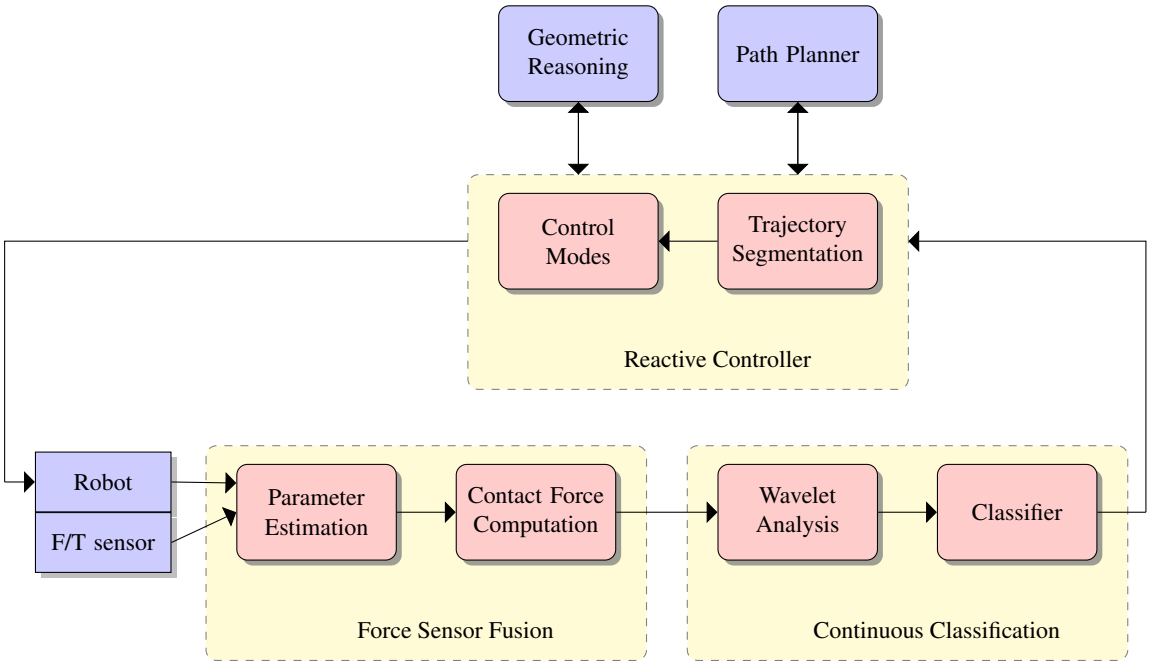


Figure 1.3: Relationship of different contributions of this thesis: force sensing, classification and reactive trajectory controller.

exchange task, we deal firstly with the fact that the robot does not have a dynamic model of the object that it is manipulating, and so the robot needs to build the dynamic model of the object. When we see an object that we do not know, we take it, and move it to feel if it is heavy, and whether the mass is homogeneously distributed, or has liquid inside the object. For the robot arm, the information of the dynamics of an object is even more important because the force control of the arm demands a dynamic model of the object. The advantages of the online estimation of the model are numerous and will be stated in the next chapter. Secondly, geometrical precision for the tracking of manipulated objects and human body parts is crucial for the success of the manipulation and for the safety. The fusion of multiple sensors can benefit the perception, and this is one of the objectives of this thesis.

1.3.2 Learning to Manipulate

This work also presents our effort to design an object to teach the robot how to exchange an object with human counterparts. The driving idea is that in a task where the robot is giving an object to a person, the robot should be able to release the object properly while it is grasped by its counterpart. To achieve this, we propose to record human-human object exchange, and then use wavelet analysis of the force sensor signals and use the Relevance Vector Machine(RVM) to build a robust classifier. The model is then transferred to the robot for Human Robot Interaction. The model can also be used to detect other types of collision between the object and environment, which could be used by the controller. In other words, learning the movement and force control law will also be investigated, although not fully studied and experimented.

1.3.3 Sensor-Based Control

When the robot is able to build the dynamic model of the manipulated object, to track the movement of objects and human body parts, and to detect special events, it needs to finish the tasks while reacting to the movement or events. Based on the work of previous colleagues, we propose a trajectory controller to achieve reactive manipulations. The controller integrates information from multiple sources, and use online trajectory generation as the central algorithm to track a moving target, grasp the moving object and avoid obstacles. The concept of control primitives is proposed and defined, and all tasks are divided into several control primitives. These control primitives enable us to define the control strategies for lower-level controllers as position-velocity control or force control.

1.4 Structure of this Manuscript

Following this introduction, this dissertation begins with the presentation of background and literature review on service manipulators, focusing on force sensor fusion, machine

learning, and reactive trajectory control. Then these topics are presented in separated chapters. Chapter III presents the problem of inertial parameter estimation, 6D target tracking, and sensor fusion techniques to solve the problems. Chapter IV begins with the problem of frequency information extraction, feature selection, and ends with the Relevance Vector Machine as the classifier. In Chapter V, we present the reactive trajectory controller, with the concept of control primitives and how it is used in human robot interactions. Following the three chapters, we give the discussion and conclusion in chapter VI. Because each chapter deals with a different problem, experimental results are given at the end of each chapter. To keep the reading clear and simple, all the mathematical material about nonlinear filtering, quaternions, wavelet analysis, and RVM learning are in the Appendix for references.

1.5 Publication, Software Development, and Research Projects

Author of this document has participated in several research projects during the thesis. The author contributed to the development of several softwares running on the robots, *Jido* and *Pr2*, and maintenance of the robots during the projects:

- Project *DEXMART*[2]. DEXMART is a European project for "DEXterous and autonomous dual-arm/hand robotic manipulation with sMART sensory-motor skills: A bridge from natural to artificial cognition". DEXMART was a large-scale integrating project, which was funded under the European Community's 7th Framework Program from 2008 to 2012.

- Project *SAPHARI*[3], Safe and Autonomous Physical Human-Aware Robot interaction, is a large-scale integrating project which is funded under the European Community's 7th Framework Program from 2011.

- Project *ASSIST (ASSIST: "Étude et développement d'un manipulateur mobile à deux bras pour l'assistance aux handicapés")* of the French agency of research ANR.

- Project *ICARO*[4]. The objective of the project ICARO is to develop tools to improve and simplify interaction between industrial robots and humans and their environment. ICARO is funded by the program CONTINT of ANR from 2011 to 2014.

The author developped several softwares during the thesis:

- *sensFusion-libs*: A C++ library for low-level sensor fusion techniques, including low-pass filters, Kalman filters, and tools for time-frequency analysis.

- *exchange-libs*: A C++ library for force events detection, built for mobile manipulators. It is based on Sparse Vector Machines (Appendix D).

[2] http://www.dexmart.eu/
[3] http://www.saphari.eu/
[4] http://icaro-anr.fr/

- *exchange-genom*: A GenoM module[5] for force events detection. The sensor interface is designed for 6D force/torque sensor and for estimated wrist force/torque.

The author participated in the development of several other softwares:

- *softMotion-libs*: A C++ library for online trajectory generation. It can be tested in the MORSE, the Modular OpenRobots Simulation Engine[6].
- *lwr-genom*: A GenoM module for the control of the KUKA LWR arm of robot *Jido*.
- *pr2-softController*: Together with *soft-controllers*, it provides a bridge between ROS and GenoM modules, offering trajectory execution for the *PR2*.
- *move3d*: The grasp planner is integrated in *move3d*, which is a generic platform for motion planning[7].
- Softwares for Bidule: Bidule is an "intelligent" sensorized device to record human-human object exchange.

Most of the softwares listed above are accessible in *robotpkg*[8].

Publication during the thesis (other publications are submitted) :

- WUWEI HE, DANIEL SIDOBRE AND RAN ZHAO, A Reactive Trajectory Controller for Object Manipulation in Human Robot Interaction, *The 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Reykjavik, Iceland, 2013*
- DANIEL SIDOBRE AND WUWEI HE, Online Task Space Trajectory Generation, *IROS 2012 Workshop Robot Motion Planning Online, Reactive, and in Real-time, invited paper.*
- WUWEI HE AND DANIEL SIDOBRE, Sigma-Point Kalman Filter for Dynamic Force Sensing during Human-Robot I nteraction Manipulation. *13e Colloque National de AIP-PRIMECA, Démarche et innovation dans la conception et la production des systèmes intégrés, Mont Doré, 2012*

[5] https://softs.laas.fr/openrobots/wiki/genom
[6] http://www.openrobots.org/wiki/morse/
[7] https://softs.laas.fr/openrobots/wiki/move3d
[8] http://robotpkg.openrobots.org/

2

Related Work and Background

There is no fixed physical reality, no single perception of the world, just numerous ways of interpreting world views as dictated by one's nervous system and the specific environment of our planetary existence.

— Deepak Chopra

Abstract. This chapter presents an overview of research and development in areas related to this thesis: mobile manipulator, force sensing, wavelet analysis, feature extraction and machine learning, reactive trajectory control and more. For the literature review, although closely related by the application of object exchange between robot and human, a division is made according to the different research areas. The theoretical backgrounds for those areas are large and diverse, so we do not intend to cover all but focus more on applications. As the work presented in this thesis is achieved during several cooperative research projects, the document will be easier to understand only after the background and related work have been presented. For this reason, the theoretical background for several related areas are briefly introduced, including geometrical reasoning, path planning, and software architectures for mobile manipulators.

2.1 Introduction

Service robots, which work in environments like home, hospital and schools with human presence raise challenges on many aspects. Since the tasks for the robot to realize will not be predefined, they are planned for the new situations that the robots have to handle. Clearly, purely motion control with predefined trajectories will be not suitable for these situations. The robot should acquire ability to react to the changing environment. Before presenting the main background of our developments, several aspects need to be discussed and be compared to the state of the art: architecture for autonomy of a service robot, force sensor fusion during physical HRI, machine learning for events detection during a task, trajectory generation and control, including online adaptation and monitoring. In this chapter, we will give a brief review for all of these aspects, starting with the more general problem of physical Human Robot Interaction (HRI).

2.2 Autonomous Mobile Manipulator

Service mobile manipulators have been developed during recent years in many laboratories all around the world, aiming to achieve assistance robots for daily or work tasks for human. Some of the robots are designed to achieve alone complex tasks in social environment. This section gives a short present of the robots at *LAAS-CNRS*, and some research accomplished with these platforms. Although not a part of contribution of this thesis, the review of the research areas is considered necessary to understand this dissertation.

2.2.1 Robots at *LAAS-CNRS*

Mobile manipulators have been developed recent years as platform of research for human robot interactions. For example, *Justin* [Ott 06a] is a robot of the *Institute of Robotics and Mechatronics* of *DLR*, the German Aerospace Center, and *Rosie* is from*Technische Universität München, (TUM)*, which is composed of two KUKA-lightweight LWR-IV arms. In *LAAS-CNRS*, robot *Jido* (Figure 2.1) has been developed, built up with a *Neobotix* mobile platform *MP-L655*, a *Kuka LWR-IV* arm, and a Schunk SAH hand of four fingers. The robot arm has an integrated estimator of external torques on each joints. Figure2.2 presents the important components on the robot, including a 3D vision system and a Kinect for human motion monitoring.

The second mobile manipulator at *LAAS* is a *PR2 (Personal Robot 2)* (Figure2.3), from *Willow Garage* [1], which is a robotics research lab and technology incubator devoted to developing hardware and open source software for personal robotics applications. The robot is an open platform based on *ROS (Robot Operating System)*[2] as the middleware. Software

[1] http://www.willowgarage.com/pages/pr2/overview
[2] http://www.ros.org

Figure 2.1: Jido, robot developed at *LAAS-CNRS*

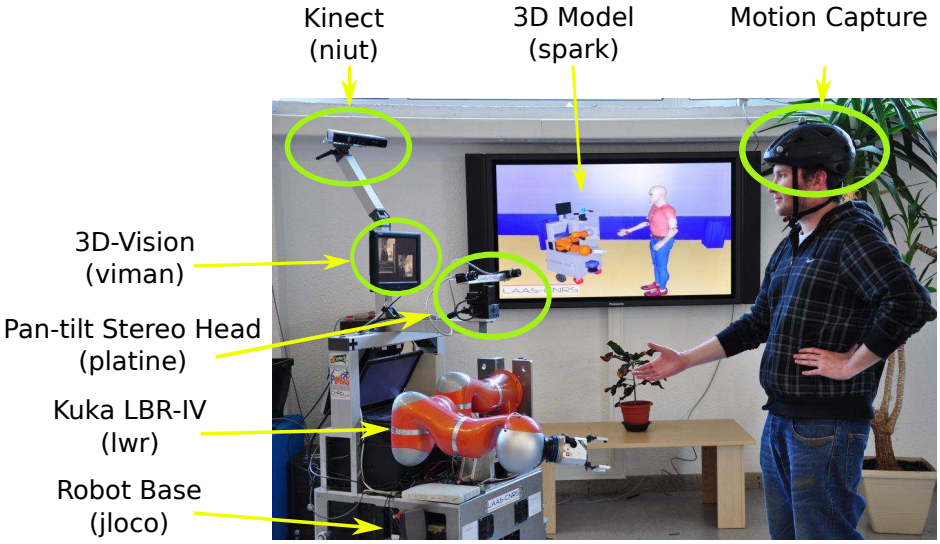


Figure 2.2: On *Jido*, the Kinect monitors the human movement, a 3D vision system is mounted on a platine. The robot model of the scene is displayed on the wall-screen.

modules of type GenoM have been successfully used together with ROS, which enables the same softwares for *Jido* to work on *PR2*. Although the arms of the *PR2* have seven DOF,

it is not equipped to estimate the external torques on joints, neither with the force/torque on wrist.

Figure 2.3: PR2, robot developed by Willow Garage

2.2.2 Software Architecture for Human Robot Interaction

The robots capable of doing HRI must realize several tasks in parallel to manage various information sources and complete tasks of different levels. Figure 2.4 shows the proposed architecture where each component is implemented as a GENOM module. GENOM [Fleury 97] is a development environment for complex real time embedded software.

At the top level, a task planner and supervisor plans tasks such as cleaning the table, bring an object to a person, and then supervises the execution. The module SPARK (**Spa**tial **R**easoning and **K**nowledge) maintains a 3D model of the whole environment, including objects, robots, posture and position of humans [Sisbot 07b]. It manages also the related geometrical reasoning on the 3D models, such as evaluating the collision risk between the robot parts and between the robot and the environment. An important element regarding SPARK that produces cost maps, which describe a space distribution relatively to geometrical properties like human accessibility. The software for perception, from which SPARK updates the 3D model of the environment, are omitted here for simplicity. The module runs at a frequency of 20Hz, limited mainly by the complexity of the 3D vision and of the

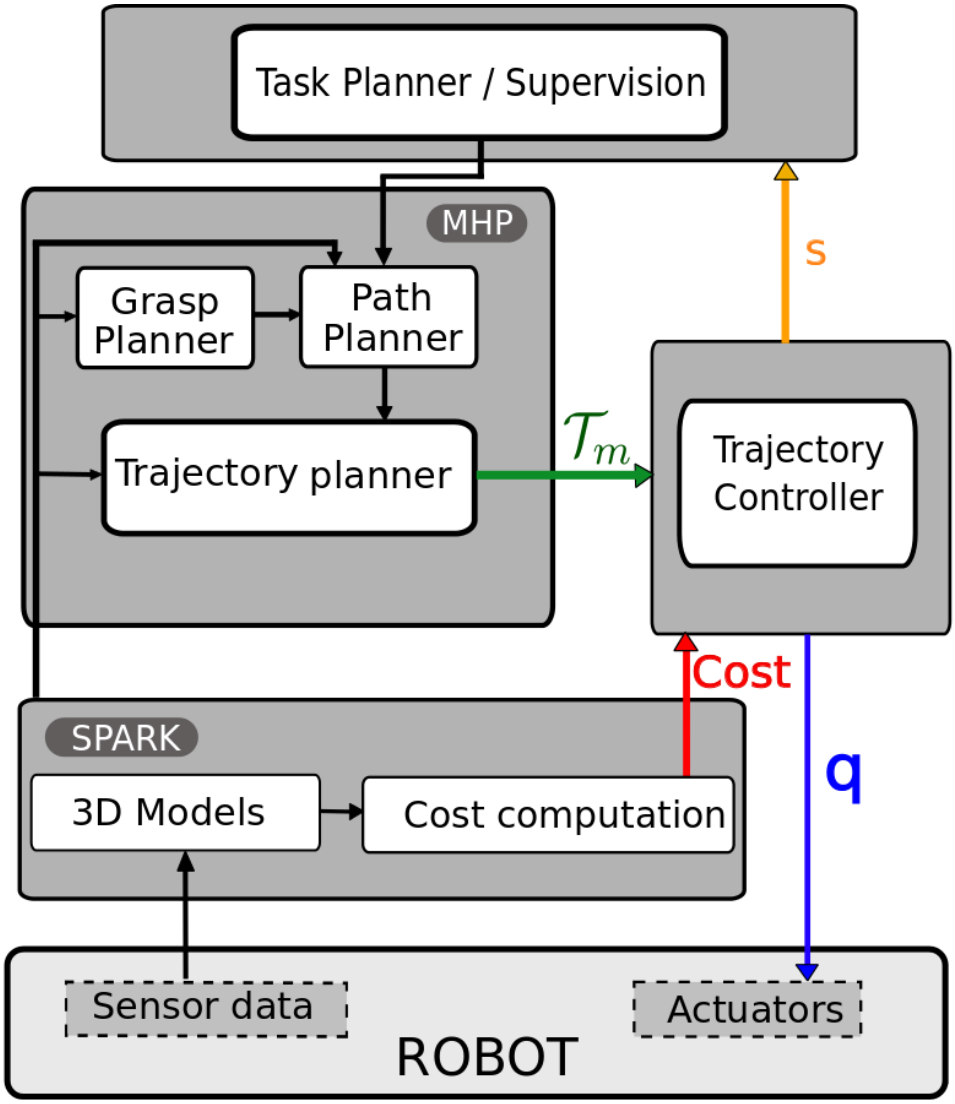


Figure 2.4: Software Architecture of the robot for HRI manipulation. $\mathcal{T}_m$ is the main trajectory calculated initially by MHP. The controller takes also costs from SPARK. The controller sends control signals in joint (q in the figure) to the robot arm controller, and during the execution, the controller returns the state of the controller (s) to the supervisor

perception of human.

Another important module, MHP (**M**otion in **H**uman **P**resence), integrates path and grasp planner. RRT (Rapidly exploring Random Tree) and its variants [Mainprice 10] are used by the path planner. The paths could be described in Cartesian or joint spaces depending on the task type. From the path defined as a broken line, an output trajectory is computed to take the time into account. MHP calculates a new trajectory each time the task planner defines a new task or when the supervisor decides that a new trajectory is needed to react to the changes of the environment.

When the motion adaptation is achieved by path replanning, the robot would switch

between planning and execution, producing slow reactions and movements because of the time needed by the complex path planner. Furthermore, if object or human moves during the execution of a trajectory planned by the module MHP, the task will fail and so a new task or a new path needs to be planned. The human counterpart often finds the movement of the robot unnatural and so not intuitive to interact with.

To overcome the problem of unnecessary replanning, we designed a reactive controller based on trajectory generation, which lies between the high-level software and the low-level controller. The trajectory controller runs at 100 Hz, an intermediate frequency between the one of the MHP planner and the one of the fast robot servo system at about 1kHz. This trajectory controller allows the system to adapt faster the trajectory to the environment changes. The basic logic behind this work is that, not every type of situation changes needs a replanning of path, or even task, which are all resource consuming. The concept is given as Figure 2.5.

The controller integrates information from other modules in the system, including geometrical reasoning and human aware motion planning. The related work of these two parts is presented in the following.

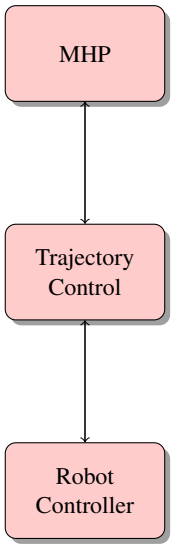


Figure 2.5: Trajectory Controller as an intermediate layer in the software architecture, the servo system on robot is fast, while task planning and path planning are slow.

2.3 Motion Planning and Control

We present firstly the basic notions for geometrical constraints in Human Robot Interaction. Then we discuss the motion of a rigid body in space, then we introduce the techniques of motion planning and control for the mobile manipulators.

2.3.1 Geometrical Constraints in HRI

The presence of humans in the work space of a robot imposes new constraints for the motion planning and control for the navigation and manipulation of a robot. This field has been studied at *LAAS-CNRS*. Among the constraints, the more important are the security, visibility and comfort of human counterpart, two of which are illustrated in Figure 2.6.

For robot motion, the workspace could be associated to many cost maps, each computed by a type of constraint. The first constraint is computed by distance and mainly to guarantee the safety and security of people at motion planning and robot control. In this case, only distance is taken into consideration. This constraint keeps the robot far from the head of a person to prevent possible dangerous collision between the robot and the person. The theory from [Hall 63] shows that the sensation of fear is generated when the threshold of intimate space is passed by other people, causing insecurity sentiments. For this reason, the cost near a person is high while is zero when distant from him.

The second constraint is called visibility, this is to limit firstly the surprise effect to a person while robot is moving nearby. Secondly, a person feels less surprised when the robot is moving in the visible zone, and feels more comfortable and safe.[Sisbot 07a]. While doing robot motion control, this constraint is used to determine if the person is paying attention to the object exchange or not.

Other constraints are also used, which can be found in [Sisbot 07a] and related publications. For example, while planning a point in space to exchange object, this point should be reachable by the person, computed by the length of his arm, and if reaching to this point would produce a confortable posture for the person. A cost of comfort is also computed for every human posture [Yang 04].

When all the cost maps are computed, they are combined by:

$$c(h,x) = \sum_{i=1}^{N} w_i c_i(h,x)$$

in which h is the posture of human and x represent the three-dimensional space in which the cost maps are computed. This combined cost map is used during the motion planning and also for the controller, which is part of the contribution of this thesis.

2.3.2 Robot Motion Representation

We firstly consider the kinematics of a rigid body in a 3D space. Considering a reference frame, F_w, which can be defined as an origin O_w and an orthogonal basis (X_w, Y_W, Z_w). A rigid body B is localized in 3D space by a frame F_B, as shown by Figure 2.7.

Translations and rotations shall be used to represent the relation between these two frames. For the translation, Cartesian coordinates are used, but for rotations, several choices are available:

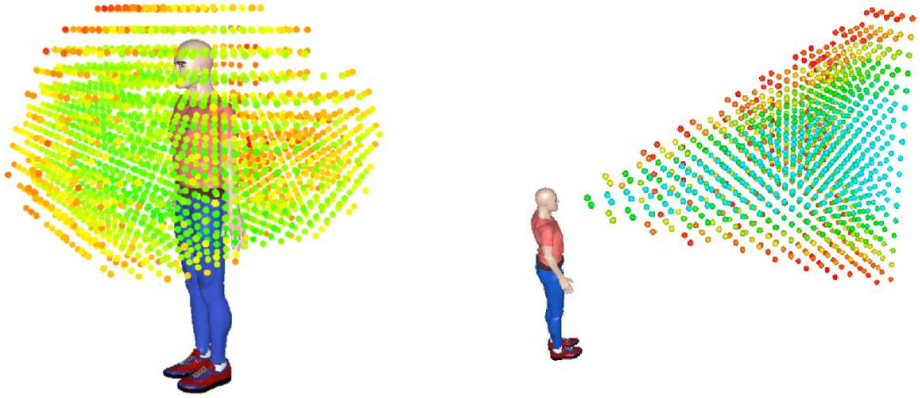


Figure 2.6: Left: 3D reachability map for a human. Green points have low cost, meaning that it is easier for the human to reach, while the red ones, having high cost, are difficult to reach. One application is that when the robot plans to give an object to a human, an exchange point must be planned in the green zone. Right: 3D visibility map. Based on visibility cost, the controller can suspend the execution if the human is not looking at the robot.

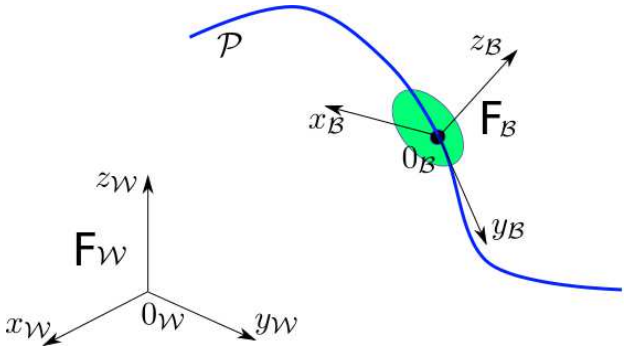


Figure 2.7: Rigid body localization in 3D space

- Axis-angles
- Rotation matrices
- Quaternions

The Homogeneous transform matrices are often used to represent the relative displacement between two frames in computer graphics and robotics, because they allow common operations such as translation and rotation to be implemented as matrix operations. The displace-

ment from frame F_W to frame F_B can be written as:

$$T_{F_W}^{F_B} = \begin{bmatrix} & & & x \\ & R_{3\times3} & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

In which, $R_{3\times3}$ is the rotation matrix, and x, y, z represents the translation. Given a point b which is localized in local frame F_B:

$$^{B}P_b = [X_b\ Y_b\ Z_b\ 1]^T$$

And $T_{F_B}^{F_W}$ represents the transformation matrix between frame F_B to F_W, then the position of point b in frame F_W is given as:

$$^{W}P_b = T_{F_W}^{F_B}{}^{B}P_b$$

If a third frame is given as F_D, and the composition of transformation matrix is also directly given as:

$$T_{F_W}^{F_D} = T_{F_W}^{F_B} T_{F_B}^{F_D}$$

Other representations used in this thesis are quaternion and vector plus axis, the details of which is given in Appendix. Readers can also refer to the literature such as the book of Siciliano [Siciliano 08] for more comparison and discussion on different types of representations on rotations and transformations.

2.3.3 Path Planning

Through this thesis, the robot is assumed to operate in a three-dimensional space ($\mathbb{R}^3$), called the work space ($\mathscr{W}$). This space often contains obstacles, which are rigid bodies and also considered geometrical, written as $\mathscr{WO}_i$, in which i means it is the i^{th} obstacle. And the free space is then $\mathscr{W}_{free} = \mathscr{W} \setminus \bigcup_i \mathscr{WO}_i$, and $\setminus$ is the subtraction operator. Motion planning can be performed in working space also configuration space $\mathscr{Q}$, or called C-space (Figure 2.8), the set of all robot configurations. Configurations are often written as q and geometry in Cartesian space as $\boldsymbol{x}$. (It should be noticed that q is also used as to represent quaternions.) Then the obstacles in the configuration space correspond to configurations where the robot is in collision with an obstacle in the workspace.

A path is a continuous curve. It can be defined in the configuration space or workspace (planning in Cartesian space). Path is different from trajectory in that path does not consider time. Given a parameter $u \in [u_{min}, u_{max}]$, often chosen such that $u \in [0,1]$, a path in configuration space is defined as a curve $\mathscr{P}$ such that:

$$\mathscr{P} : [0,1] \rightarrow \mathscr{Q} \ \ where \ \ \mathscr{P}(0) = q_{start}, \mathscr{P}(1) = q_{goal} \ \ and \ \ \mathscr{P}(s) \in \mathscr{Q}_{free}, \forall s \in [0,1] \tag{2.2}$$

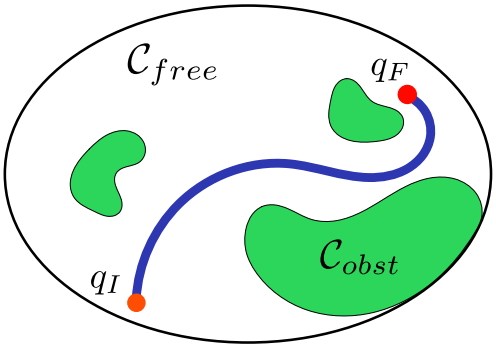


Figure 2.8: Configuration Space and a planned path in C space.

Path planning has been one of the essential problems in robotics. Among numerous papers and books, we have chapter V of *Handbook of Robotics*, by Kavraki and LaValle [Kavraki 08] which provides an introduction to this domain, and the book of LaValle [LaValle 06], in which numerous methods are presented. Figure 2.9 shows an example of the result of path planning, which gives a series of points in the configuration space, linking points q_I and q_F.

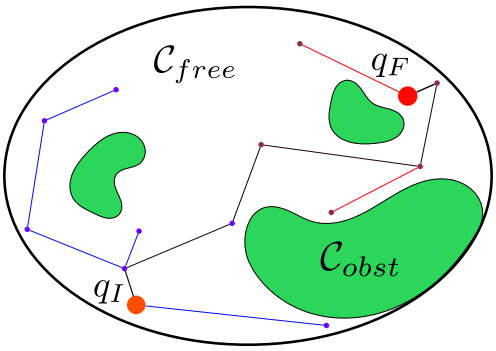


Figure 2.9: Results of path planning (by diffusion) as a series of points in the configuration space. The result is in black.

When the robot shares the workspace with humans, the path planner must take into account the costs of HRI constraints. We perform this planning with the T-RRT method [Jaillet 10] which takes advantage of the performance of two methods. First, it benefits from the exploratory strength of RRT-like planners [LaValle 01] resulting from their expansion bias toward large Voronoi regions of the space. Additionally, it integrates features of stochastic optimization methods, which apply transition tests to accept or reject potential states. It makes the search follow valleys and saddle points of the cost-space in order to compute low-cost solution paths (Fig. 2.10). This planning process leads to solution paths with low value of integral cost regarding the costmap landscape induced by the cost function.

In a smoothing stage, we employ a combination of the shortcut method [Berchtold 94] and of the path perturbation variant described in [Mainprice 11]. In the latter method, a path

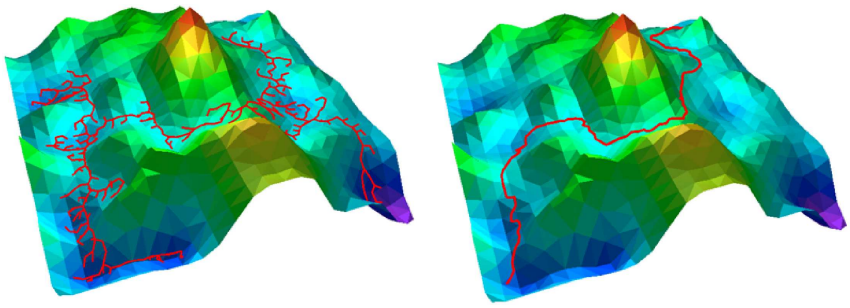


Figure 2.10: T-RRT constructed on a 2D costmap (left). The transition test favors the exploration of low-cost regions, resulting in good-quality paths (right).

$\mathscr{P}(s)$ (with $s \in \mathbb{R}^+$) is iteratively deformed by moving a configuration $q_{perturb}$ randomly selected on the path in a direction determined by a random sample q_{rand}. This process creates a deviation from the current path, The new segment replaces the current segment if it has a lower cost. Collision checking and kinematic constraints verification are performed after cost comparison because of the longer computing time.

The path $\mathscr{P}(s)$ computed with the human-aware path planner consists of a set of via points that correspond to robot configurations. Via points are connected by local paths (straight line segments). Additional via points can be inserted along long path segments to enable the path to be better deformed by the path perturbation method. Thus each local path is cut into a set of smaller local paths of maximal length l_{max}.

2.3.4 Interpolation

A path planner gives path as a set of points. To link these points, an interpolation needs to be computed. Interpolation computes the evolution of the coordinates between two positions in space. For translation in Cartesian space, linear interpolation between two positions P_B and P_C is simply given as:

$$P(u) = P_B + u.(P_C - P_B)$$

where $0 \geq u \geq 1$ is the interpolation parameter. The interpolation of orientations is then more complicated; we can use quaternions or axis-angle representation. Several choices exist for the quaternion interpolation: LERP, SLERP, and NLERP. The easiest way to interpolate between two points is the linear interpolation (LERP). Given the starting point as q_0, ending point q_1, and the interpolation parameter u, $LERP(q_0, q_1, u)$ yields for each u a point along the straight line connecting the two points:

$$LERP(q_0, q_1, u) = q_0 + u.(q_1 - q_0)$$

This interpolation gives points, which are not on the unit sphere. *SLERP* is *LERP* but performed on the unit sphere [Shoemake 85]:

$$SLERP(q_0, q_1, u) = \frac{sin(1-u)\Omega}{sin\Omega} q_0 + \frac{sin(u\Omega)}{sin(\Omega)} .(q_1 - q_0)$$

Or written as:

$$SLERP(q_0, q_1, u) = q_1(q_1^{-1} q_2)^u \tag{2.3}$$

For the trajectories in this thesis, the interpolation is performed with axis-angle representation, passing through quaternions. Between point P_0 and P_1, the displacement from P_0 to P_1 can be written as:

$$Dep_{0\to1} = [P(u), a_{0,1} V_{0,1}]$$

In which, P(u) is the translation interpolation, and the rotations are firstly transformed to quaternion, interpolated in quaternion, then transformed to homogeneous matrix for applications of robot control in Cartersian space.

2.3.5 Trajectory Generation

Trajectory generation computes the time evolution for the robot, in joint space or Cartesian space. Trajectories are then provided as the input to the controller. Trajectories are important because they enable the system to ensure:

- feasibility: the motion can be verified to respect the dynamic constraints of lower-level controllers.
- comfort: the trajectories can be limited on acceleration and jerk to guarantee the comfort for humans.
- optimization: global optimization can be achieved, depending on the objectives.

2.3.5.1 Trajectory Types

Trajectories can be classified as several categories (Figure 2.11). *Mono-dimensional* trajectories correspond to trajectories for systems of only one degree of freedom (DOF), while *Multi-dimensional* for more than one DOF. Compared to the case of mono-dimension, the difficulty for multi-dimensional trajectories is the possible need to synchronize different axis in time. There are point-to-point trajectories, the generation of which are to link two points with a trajectory, and multi-points trajectories, which need to pass through points in the middle too.

From another point of view, trajectories can be planned in joint space or Cartesian space. Joint space trajectories have several advantages:

- Trajectories planned in joint space can be used directly to lower-level controllers without need to compute inverse kinematics.
- No need to deal with the redundant joint or singularities of 7 DOF manipulators or mobile manipulators.
- The dynamic constraints, like maximum acceleration on joints, can be considered while generating the trajectories. While planning in Cartesian space, this should be tested after inverse kinematics.

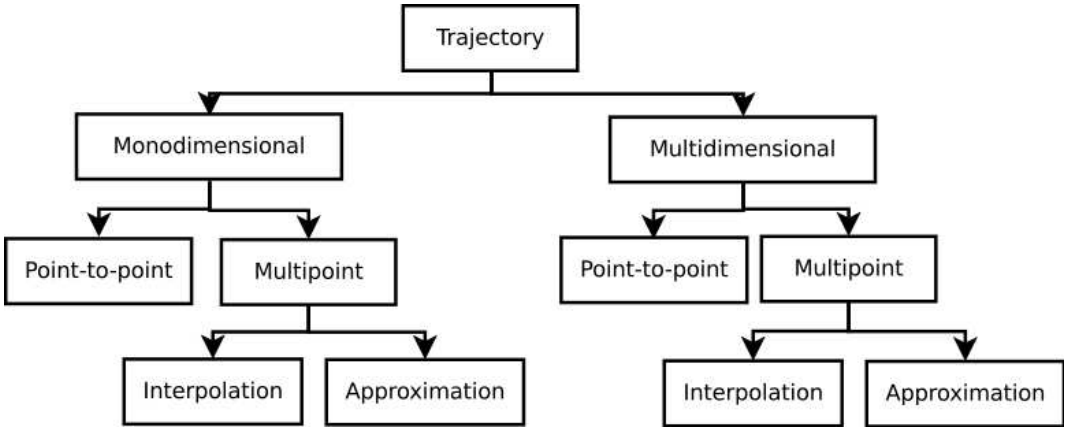


Figure 2.11: Categories of trajectories [Biagiotti 08]

Cartersian space trajectories may produce natural and more acceptable results for people. One example is that to give an object to a person, a trajectory planned in Cartersian space can produce a straight-line movement, which is not easy to guarantee while planned in joint space. The advantage of Cartesian space planning is more evident when the robot needs to manipulate a cup of tea, for example, without spitting it out.

Trajectories can be planned *on-line* and *off-line*. This thesis tries to achieve reactive robot control, so only *on-line* is suitable for the trajectory control level.

2.3.5.2 Trajectory Generation Algorithms

Trajectory generation for manipulators have been discussed in numerous books and papers, among which readers can find [Brady 82], [Khalil 99] and [Biagiotti 08]. Kroger, in his book [Kroger 10b], gives a detailed review on on-line and off-line trajectory generation. Kroger gives a classification of trajectories as in table 2.1 [Kroger 10b]. For each type in the table, the initial and final condition for velocity and acceleration are arbitrary. The final conditions for velocity, acceleration and jerk are noted as V_F, A_F, and A_F, respectively. Types defined in this table are used in this document.

Our approach to build the controller capable of controlling a complete manipulation tasks is based on Online Trajectory Generation. More results on trajectory generation for

Table 2.1: Different types for on-line trajectory generation

	$V_F = 0$ $A_F = 0$ $J_F = 0$	$V_F \in \mathbb{R}$ $A_F = 0$ $J_F = 0$	$V_F \in \mathbb{R}$ $A_F \in \mathbb{R}$ $J_F = 0$	$V_F \in \mathbb{R}$ $A_F \in \mathbb{R}$ $J_F \in \mathbb{R}$
$A^{max} \in \mathbb{R}$	Type I	Type II	-	-
$A^{max} \in \mathbb{R}$ $J^{max} \in \mathbb{R}$	Type III	Type IV	Type V	-
$A^{max} \in \mathbb{R}$ $J^{max} \in \mathbb{R}$ $D^{max} \in \mathbb{R}$	Type VI	Type VII	Type VIII	Type IX

robot control can be found in [Liu 02], [Haschke 08], and [Kröger 06]. Kroger proposed algorithms to generate type III trajectories in table 2.1. Broquere et al. ([Broquere 08b], [Broquere 08c]) proposed type V trajectories, with arbitrary final velocity and acceleration. The difference is important because in this thesis, the controller needs to generate trajectories to join points with arbitrary velocity and acceleration. The details of the algorithm used are given in chapter V to help the readers to understand the document.

2.3.6 Grasp Planning

To plan tasks of picking an object or giving an object to a person, a grasp planner must produce valid grasps on the object. The grasp planner used in this work is based on previous work at *LAAS*, presented in [Bounab 08] and [Saut 12]. The proposed approach does not rely exclusively on a heuristic that can introduce a bias on how the object is grasped. Our objective is to build a grasp list to capture the variety of the possible grasps. The path planner chooses among all the grasps, even in a cluttered environment, for an object with a complex shape. In the following, we illustrate the method with the Schunk Anthropomorphic Hand (SAH) depicted on Fig. 2.12 as it is the one used in our laboratory. It has four fingers. Each finger has four joints except the thumb. Only the three first joints are actuated, the last one being coupled with the third one. The thumb has an additional joint to place it in opposition to the other fingers.

A single grasp is defined for a specific hand type and for a specific object. The object model is supposed to be a triangle mesh: A set (array) of vertices (three coordinates) and a set of triangles (three indices in the vertex array). It is assumed to be a minimum consistent *i.e.* has no duplicate or isolated vertices nor degenerate triangles.

2.3.6.1 Grasp Definition

In the following, we define a grasp by (See Fig. 2.12):

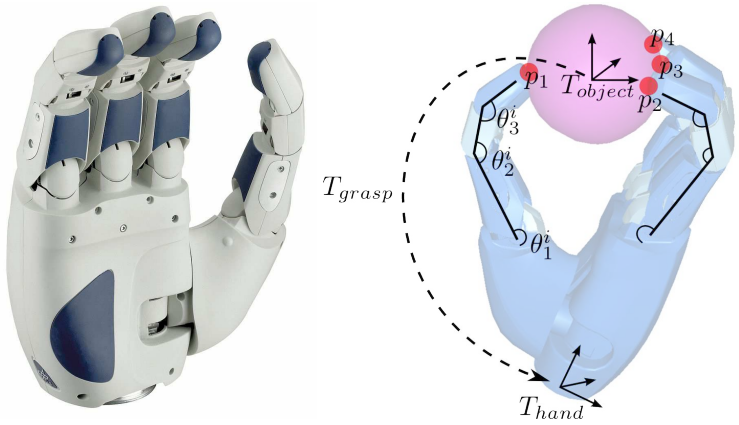


Figure 2.12: Left: The Schunk Anthropomorphic Hand used to illustrate our grasp planning method. Right: A grasp is defined by a transform matrix $T_{grasp} = T_{hand}^{object}$, the finger joint parameters of each finger i $(\theta_1^i, \theta_2^i, \ldots)$ and a set of contact points $(p_1, p_2, \ldots)$.

- A transform T_{grasp} between the object and the hand frame.
- A set of finger joint parameters $(\theta_1^i, \theta_2^i, \ldots)$ where i is the ID of the finger.
- A set of contact points $(p_1, p_2, \ldots)$ that can be deduced from the two previous items.

A contact contains the following information:

- Position: both a 3D vector and a set (triangle index + barycentric coordinates) to store the position.
- Normal: the plane normal of the triangle the contact belongs to.
- Coulomb friction: used further to compute the grasp stability.
- Finger ID: to store which finger is responsible of the contact.
- Curvature: it is interpolated from the curvature of the vertices of the triangles.

As the main concern of the grasp planner is motion planning, it is not possible to rely on the computation of a simple grasp or to compute grasps according to a heuristic that could introduce a bias on the choice of the grasp. It is preferable to compute a grasp list that aims to reflect the best the variety of all possible grasps of the object. Our algorithm applies the following steps that will be detailed further:

- Build a set of grasp frame samples.
- Compute a list of grasps from the set of grasp frames.
- Perform a stability filter step.
- Compute a quality score for each grasp.

More details can be found in the cited papers.

2.3.7 Trajectory Control

Reactive controller for object manipulation is a research topic that is part of the fundamentals of robotic manipulation. Firstly, trajectory generation based approaches where developed. In [Buttazzo 94], results from visual system pass firstly through a low-pass filter. The object movement is modeled as a trajectory with constant acceleration. On this basis, catching position and time is estimated. Then a quintic trajectory is calculated to catch the object, before being sent to a PID controller. The maximum values of acceleration and velocity are not checked when the trajectory is planned, so the robot gives up when the object moves too fast and the maximum velocity or acceleration exceeds the capacity of the servo controller. In [Gosselin 93], inverse kinematic functions are studied, catching a moving object is implemented as one application, a quintic trajectory is used for the robot manipulator to joint the closest point on the predicted object movement trajectory. The systems used in those works are all quite simple and no human is present in the workspace. A more recent work can be found in [Kröger 12], in which a controller for visual servoing based on Online Trajectory Generation (OTG) is presented. The results are promising.

Secondly, the research area of visual servoing provides also numerous results, a survey of which were presented by Chaumette and Hutchinson [Chaumentte 06], [Chaumentte 07] and a comparison of different methods can be found in [Farrokh 11]. Classical visual servoing methods produce rather robust results and stability and robustness can be studied rigorously, but they are difficult to integrate with a path planner, and could have difficulties when the initial and final positions are distant.

Another approach to achieve reactive movements is through Learning from Demonstration (LfD). In [Calinon 04] and [Vakanski 12a], points in the demonstrated trajectory are clustered, then a Hidden Markov Model(HMM) is built. Classification and reproduction of the trajectories are then based on the HMM. A survey for this approach is proposed in [Argall 09]. Although LfD can produce the whole movement for objects manipulation, many problems may arise in a HRI context as LfD demands large set of data to learn, and the learned control policies may have problem to cope with a dynamic and unpredictable environment where a service robot works.

The controller must be capable of dealing with various data in HRI context. Compared to methods mentioned above, approaches based on OTG have the following advantages:

- The integration with a path planner is easy and allows to comply with kinematic limits like the one given by human safety and comfort.
- The path to grasp a complex moving object is defined in the object frame, making sure that the grasping movement is collision free.
- The trajectory based method allows to create a simple standard interface for different visual and servo systems, so easy plug-in modules can be created.

The controller integrates various information from high-level software, such as SPARK and MHP, which were presented previously in this chapter.

2.4 Sensor Fusion

2.4.1 Multisensor Data Fusion

Multisensor data fusion aims to combine information form multiple sensory data or data derived from different sources. The goal of sensor fusion is to obtain information which in some sense better than the situation where the sources are used separately. Essentially, different sensors have their own limitations and fusion algorithms could improve the tasks that the sensors are to achieve. Multisensor tracking is one of the most important area where sensor fusion is used to improve the tracking results. [Hall 04] defined functional roles of multisensor integration, and a four-level category with fusion algorithms. An introduction can been found in [Llinas 98]. A review on system architectures on sensor fusion is provided in [Elmenreich 07]. And Smith in [Smith 06] provides a review on sensor fusion for target tracking, which is also an application for robotics of fusion in this thesis. Luo et al. [Luo 12] gives a review of application on mechatronics, and the classification of techniques according to different application *levels*. The table is given as 2.2, and several techniques used or compared in this document have been added. It shall be noticed that machine learning can be seen as part of sensor fusion, but it is separated from the estimation in this thesis for clarity.

2.4.2 Force Sensing

The estimation of the inertia parameters and external forces are separately studied in the literature. An off-line estimation of inertia parameters of an attached object on an industrial manipulator is studied by An et al.[C.H.An 88]. The approach uses ordinary least-squares estimation and predefined trajectories. The excitation trajectories for the estimation is addressed by Swever et al.[Swevers 97]. Many other approaches are proposed, like [M.Niebergall 01], mainly based on least-square techniques, so they all ignore the errors in the data. *Kubus* et all[Kubus 08a] proposed an on-line method based on recursive total least-squares. Least-square methods are suitable when the inertia parameters are expressed in the force sensor frame. We need the inertia matrix to recognize the object by comparing the inertia parameters with a database of manipulated objects, but its values in the sensor frame are not constant for the same object when the grasp positions or orientations are different. For example, grasping the bottom or the center of a bottle, the inertia matrix in the end effector frame is not the same. In this work, we compute the inertia matrix in the object frame, which change the system to be nonlinear, and same least square method can no longer be used.

Table 2.2: CLASSIFICATION OF FUSION ALGORITHMS

Low level fusion	Medium level fusion	High level fusion
Estimation methods	Classification methods	Inference methods
Recursive: • Kalman filter • Extended Kalman filter • Unscented Kalman filter • Particle filters Non-recursive: • Least squares • Weighted average Covariance based: • Cross covariance • Covariance intersection • Covariance union	• Parametric templates • Cluster analysis • K-means clustering • Learning vector quantization • Artificial neural networks • Support vector machine • Relevance vector machine	• Bayesian inference • Dempster-Shafer theory • Expert system • Fuzzy logic

For the external force sensing, with inertia parameters of the tool or manipulated object known, Uchiyama [Uchiyama 85] estimated external forces/torques with an extended Kalman filter (EKF), which is based on Taylor expansion of the nonlinear system models. Garcia [Garcia 05] proposed a force and acceleration sensor fusion for compliant robot motion control, with a known tool on the manipulator.

2.5 Wavelet and Classification

The problem adressed here is the ability for a robot to detect events from external forces. An object is designed to teach the robot this ability, the problem is then to chose methods to extract features from a time signal and classify different events. Object exchange grasping-releasing synchronization has been studied in various papers. Nagata et al. presented a solution to exchange objects between a human and a four-fingered hand [Nagata 98]. They used a 6-axis force/torque (f/t) sensor mounted on each fingertip to control the grasping force and evaluate modifications in human grasp condition. In the domain of cooperative manipulation with humans [Aggarwal 07, Takubo 02], researchers try to detect the different stages of cooperation like contact and slip.

The works of Nakazawa, Kim and Inooka [Kim 02, Nakazawa 01, Nakasawa 99] are based on a similar approach as ours: the measure of forces during object exchange. The object is designed to be grasped by two fingers, which are the thumb and the forefinger and

the system measures only the forces in two directions. The paper asked a set of questions when building also an object specifically designed to measure hand-over forces. The same questions can define the objective of our learning based approach too:

- "How does the giver know the receiver's contact?"
- "Which one starts to act first for the smooth hand-over?"
- "How do both of them control the grasp forces during the hand-over?"

And they established that "the giver may feel slight vibrations on the fingertip and a change in the weight of the object as the receiver contacts the object". They also showed that the grasp forces are adapted to the weight of the object. Our goal is to model the vibration and the change of weight with time-frequency analysis and use classification techniques. The grasp force will not be studied as we want our learned model to be used for a robot arm with only a wrist force/torque sensor.

Some other interesting results can be found in [Romano 11], in which researchers measure the *vibration* condition to detect the contact with environment when the robot places an object. The vibration condition is defined as a threshold on the high-frequency hand acceleration signal. The threshold should be found by trial and error, which would be especially difficult when multiple types of contact should be classified.

The natural trajectory, in the other hand, used by humans when they exchange objects has been studied by Kajikawa and all [Kajikawa 95] to plan hand-over movement for robots. They summarize the characteristics of the trajectories as follows:

- The receiver tends to begin his motion after the deliverer achieves maximum approach velocity.
- At the start of his motion, the receiver approaches the deliverer with a straight and rapid trajectory, and without accurately determining the direction of his hand.
- The receiver then adjusts the direction of his hand by generating a rotational trajectory.
- Finally the receiver sufficiently decreases the relative velocity to match that of the deliverer.

This can be translated to two trajectories of the deliver and the receiver. Some established techniques can be used to learn the trajectories and the synchronization of them, like based on Hidden Markov Model [Vakanski 12b], or on dynamical system model[Khansari-Zadeh 11], again we will only show the recorded data in Chapter VI. Instead, we focus on the synchronization of the grasping and the finger opening. The methods used in this work are wavelet analysis based, readers can refer to [Mallat 08] for a detailed discussion on wavelets and the comparison to Fourier analysis and its variants. The Wavelet Packet Transformation

(WPT) used for feature extraction are reported in various works, as in [Englehart 99] for myoelectric signal, and in [Learned 95] for underwater acoustic signals.

The Linear Discriminant Analysis are used to select features, a nice explanation of which can be found in the book of Bishop [Bishop 06]. Relevance Vector Machine, used as a classifier in this work, was firstly introduced by Tipping [Tipping 01]. Frequency information based classification, although with different feature formation, is also reported with successful application on touch classification [Sato 12], and especially on image classification, such as in [Rehman 12].

A first design of the object intended to record human-human object exchange was reported in [Sidobre 09], using wavelet analysis to study the forces. The previous work gives interesting results. And in this thesis, we propose a more general solution by learning to build more universal models.

2.6 Conclusion

This chapter presented firstly the robots and the software architecture, and included the literature review of the topic of this thesis and some presentation of the related works at *LAAS-CNRS*. This thesis has been a part of the effort to develop a human-aware, autonomous mobile manipulator. For this reason, the research objectives are all real problems related and cooperation related. The first consequence is that the thesis covers different areas, but all application linked together. Even if the results depend on other components of the system: the control would not be possible without a human-aware motion planner, and the fusion would not achieve any results if not for a 3D vision system. Although time consuming, cooperation and being part of big research project provided an opportunity to learn various related domains and to train the ability to work as part of a team.

Because it covers three different research topics, this document does not aim to provide an exhaustive literature review on all the topics. Nevertheless, comparisons between the chosen approach and other methods are provided. Trajectory control for visual tracking, for example, achieves similar functions as classical visual tracking. Some cited papers are review papers for the area, and provide good reading lists.

3

Force Vision Sensor Fusion

The eye sees only what the mind is prepared to comprehend.

— Henri Bergson

Abstract. The objective of this research work is to achieve a sensor-based reactive control, for which the precision and robustness of perception is an important component. In this chapter, we will discuss two problems in robot perception to improve the interactive manipulation: the force sensing to identify the unknown manipulated object, and kinematics tracking model based on vision and force fusion. For the two problems, the nonlinear Kalman filter is used as the sensor fusion method. Identifying the dynamics parameters of the manipulated object during manipulation is the main challenge for the first problem, while updating the filtering process from different sensors, mainly force and vision, is the focus for the second. The basics of nonlinear Kalman filters are included in appendix.

3.1 Introduction

Robot control in human-robot interactions is a challenging task. Simple planned position control is not adequate in this context. The nature of HRI demands the robot to be reactive during the manipulations, which means the robot should adapt its activities according to the sensory inputs, being visual, tactile, forces/torques, or depth sensors. In this thesis, we call it sensor-based reactive manipulations, which differs from pre-planned manipulations in the

way that the reactions do not need constant involvement of the planning algorithms, several important advantages can be obtained:

- Planning is often time-consuming hence not fast enough for real-time reactions.
- Reactive control based only on the proposed trajectory generator have a fixed computing time, while planning by RRT and similar methods have a varying computation time.
- By building models from the perception data, the manipulation control is directly improved.

In this chapter, we propose to solve two problems of sensor fusion aiming to achieve a controller for sensor-based manipulations. The work on machine learning and control will be discussed in other chapters. Firstly, we discuss the force-sensing problem. The goal of this part is to use a 6D forces/torques sensor mounted on the wrist of the robot arm to estimate the inertial parameters of the tool/load mounted on the robot. Then with the inertial parameters, the external forces/torques on the tool are computed. Secondly, we use the results of force sensing to improve the kinematics tracking by force-vision sensor fusion, including object recognition by inertial parameters, and object localization integrated into visual tracking.

On-line estimation of the inertial parameters of the manipulated object can benefit many aspects of the manipulation, as we will show in the next section. The vision/force fusion does not only improve the performance of the trajectory controller, but also help the facts assessment for the higher level software.

Multi-sensor fusion is a problem for which various methods are proposed in the literature. The review and comparison of different techniques can be found in chapter II. The choice of the technique determines a different choice of the models and the formulation for the problem. For the advantage of clarity in mathematical background and software performance, Kalman Filter based methods are chosen for the two problems at hand. In the following chapters, we will model the problems as a dynamical system, suitable for Kalman filter formulations. It should be noticed that other techniques would use totally different models. For example, when the force sensing part is solved by offline Least Squares, and the tracking part is fused by Dempster-Shafer theory, a different model is needed of the same problem.

The rest of the chapter is organized as follows. Firstly, the problem formulation for force sensing is in the Section 3.2. The multimodal sensor fusion will be discussed in Section 3.3. After this, we will give the algorithm of Unscented Kalman filtering in Section 3.4.

3.2 Force Sensing

3.2.1 Inertial Parameters Estimation

Various applications in service and assembly robotics can benefit from the estimation of inertia parameters, which are the mass, the coordinates of the centre of mass, and the elements of the inertia matrix. Furthermore, complex tools are not always provided with suitable dynamical models so that experimentally estimated inertial parameters can be more precise and more useful. By estimating the inertial model of the tool/manipulated object, the external forces executed directly on the end effector or on the manipulated object should be available to improve the control of object exchange.

The techniques presented in this chapter are also used in the next part of this work, associated to the design of an object intended to teach the robot to synchronize the grasp-release during object exchange, as external forces/torques are needed for the learning procedure.

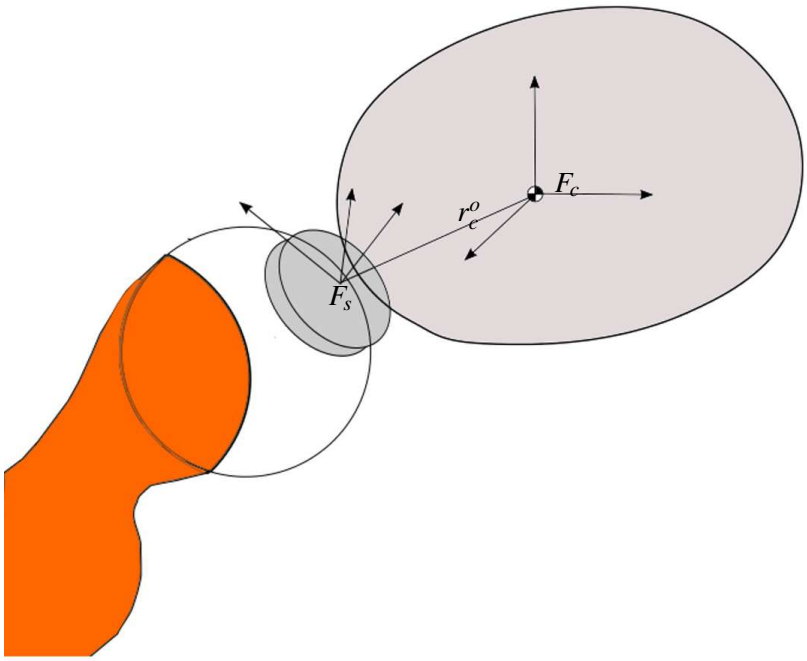


Figure 3.1: Frames for force/torque sensing

We propose a model of the relation between the dynamics of the manipulated object and the robot wrist force/torque in this section. Figure 3.1 shows the important frames for the problem. The hand, the force/torque sensor and possibly the acceleration sensor are all mounted on the end of the robot arm. Firstly, we chose the robot base as the inertial reference coordinates system for this system. This assumption is reasonable because the robot rarely manipulates objects while doing fast navigation. The transform matrices between the sensors and the hand are fixed and measured by experiments. The one from hand to arm is

calculated through forward kinematic transformation.

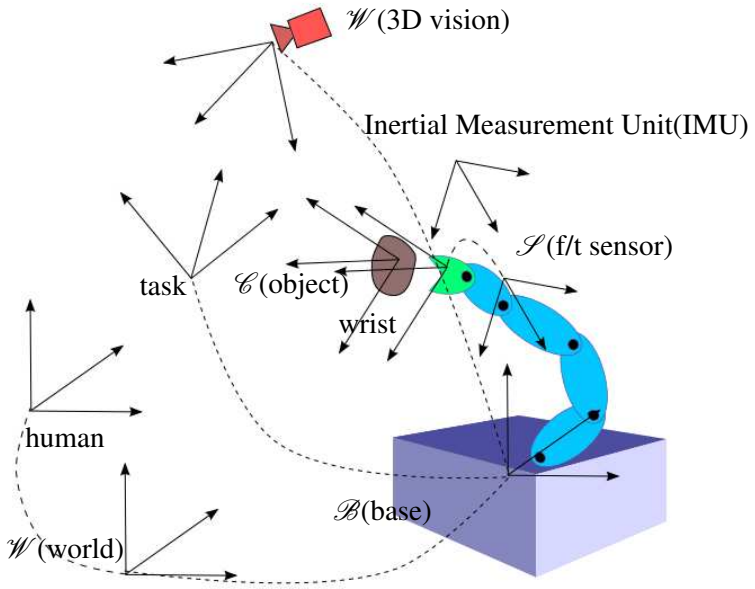


Figure 3.2: The frames for force/torque sensing and vision tracking and relations between them. The dashed lines are associated to the transformation matrices.

Once the position of the manipulated object in the frame of force/torque sensor is estimated, the position in the hand frame is obtained. We skip all the detailed terms of the matrices for the clarity of the document. To simplify the dynamic model, we also make the assumption that the frame of the object is parallel to the frame of hand. This is not a problem because in the next section, we do not try to match the exact inertia matrix but only the characteristics of the matrix are used for object recognition, and the rotation of frame will not affect the localization of the center of mass (COM). A more specific choice of the frames can be accomplished after the first estimation. And different frame rotations will not affect the position of the COM in the force sensor frame and the mass, of course, stays unchanged. The force/torque sensor measures the action and reaction force and torque between the robot arm and the end effector, in our case, a robot hand or a gripper.

Notations:

$\boldsymbol{f}_m^s$: Forces measured by f/t sensor, w.r.t. the sensor frame.

$\boldsymbol{\tau}_m^s$: Torques measured by f/t sensor, w.r.t. the sensor frame.

$\boldsymbol{f}^s$: Real action forces between sensor and object.

$\boldsymbol{\tau}^s$: Real action torques between sensor and object.

$\boldsymbol{f}_{ext}^c$: Contact forces on object, w.r.t. the object frame.

$\boldsymbol{\tau}_{ext}^c$: Contact torques on object, w.r.t. the object frame.

$\boldsymbol{f}_{off}^s$: Sensor offsets on forces.

$\boldsymbol{\tau}_{off}^s$: Sensor offsets on torques.

$\boldsymbol{f}^c$: Total force acting on the center of mass.
$\boldsymbol{\tau}^c$: Total torque acting on the center of mass.

We obtain $\boldsymbol{f}_m^s$, $\boldsymbol{\tau}_m^s$, the measured forces and torques, w.r.t. the sensor frame by:

$$\boldsymbol{f}^s = \boldsymbol{f}_m^s - \boldsymbol{f}_{off}^s \tag{3.1}$$
$$\boldsymbol{\tau}^s = \boldsymbol{\tau}_m^s - \boldsymbol{\tau}_{off}^s \tag{3.2}$$

The static equilibrium equation of the object gives:

$$\boldsymbol{f}^c = \boldsymbol{R}_s^c \boldsymbol{f}^s + \boldsymbol{f}_{ext}^c \tag{3.3}$$
$$\boldsymbol{\tau}^c = \boldsymbol{R}_s^c \boldsymbol{\tau}^s + \boldsymbol{r}_s^c \times (\boldsymbol{R}_s^c \boldsymbol{f}^s) + \boldsymbol{\tau}_{ext}^c \tag{3.4}$$

in which $\boldsymbol{R}_s^c$ is a 3×3 rotation matrix from sensor frame $\mathscr{S}$ to object frame $\mathscr{C}$ 3.2. and $\times$ is the cross product. When the sensor and object frames are parallel, $\boldsymbol{R}_s^c$ becomes the identity matrix. $\boldsymbol{r}_s^c$ is the position vector from $\mathscr{S}$ to $\mathscr{C}$:

$$\boldsymbol{r}_s^c = (c_x \; c_y \; c_z)^T \tag{3.5}$$

We consider the process as two steps:
1. Moving the manipulator in free space, the set of inertia parameters are estimated.
2. With the inertial parameters, the dynamics of the object is simulated, and then the non-contact forces are eliminated from the sensed forces/torques during the manipulation.
When the robot moves in free space, the contact forces/torques are zero, the measured f/t are only inertial f/t:

$$\boldsymbol{f}^c = \boldsymbol{R}_s^c \boldsymbol{f}^s \tag{3.6}$$
$$\boldsymbol{\tau}^c = \boldsymbol{R}_s^c \boldsymbol{\tau}^s + \boldsymbol{r}_s^c \times (\boldsymbol{R}_s^c \boldsymbol{f}^s) \tag{3.7}$$

The Newton-Euler equations, w.r.t. the coordinate frame whose origin coincides with the body's center of mass of the object are

$$m\boldsymbol{a}^o = \boldsymbol{R}_c^o \boldsymbol{f}^c + m\boldsymbol{g}^o \tag{3.8}$$
$$\boldsymbol{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{\tau}^c \tag{3.9}$$

where the inertia matrix is:

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{3.10}$$

Equation 3.8 is written w.r.t. robot base frame $\mathscr{O}$. In which:

$\boldsymbol{R}_c^o$: the transformation matrix from object frame $\mathscr{C}$ to frame $\mathscr{O}$

a^o: the linear acceleration of center of mass of the object w.r.t. frame $\mathscr{O}$. Variables in equation(3.9) are all expressed in frame $\mathscr{C}$, which means that:

$\boldsymbol{\omega}$ is the angular velocities of the object, w.r.t. the local object frame $\mathscr{C}$

Replacing f^c and τ^c in equations (3.8),(3.9) by(3.6),(3.7) and as $\boldsymbol{R}_s^c$ is the identity matrix because of the assumption that frame $\mathscr{C}$ and frame $\mathscr{S}$ are parallel, from which we have also $\boldsymbol{R}_c^o$ is the same as $\boldsymbol{R}_s^o$. We obtain the complete system model for free space movement:

$$m\boldsymbol{a}^o = \boldsymbol{R}_s^o(\boldsymbol{f}_m^s - \boldsymbol{f}_{off}^s) + m\boldsymbol{g}^o \tag{3.11}$$

$$\boldsymbol{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{\tau}_m^s - \boldsymbol{\tau}_{off}^s + \boldsymbol{r}_s^c \times (\boldsymbol{f}_m^s - \boldsymbol{f}_{off}^s) \tag{3.12}$$

In this equation, the linear acceleration $\boldsymbol{a}^o$, the angular velocity $\boldsymbol{\omega}$ and angular acceleration $\boldsymbol{\alpha}$ can be measured by an accelerometer mounted on the robot hand or gripper, and transformed to the proper frame. As stated by Kroger in [Kubus 08b], these parameters can also be obtained from encoder/resolver measurements on the robot arm. $\boldsymbol{R}_s^o$ can be calculated by inverting $\boldsymbol{R}_o^s$, which is available as arm kinematics. $\boldsymbol{f}_m^s$ and $\boldsymbol{\tau}_m^s$ are measured by the force/torque sensor. The parameters to estimate are:

$$\boldsymbol{\varphi} = (m \;\; c_x \;\; c_y \;\; c_z \;\; I_{xx} \;\; I_{xy} \;\; I_{xz} \;\; I_{yy} \;\; I_{yz} \;\; I_{zz} \;\; f_{offx} \;\; f_{offy} \;\; f_{offz} \;\; \tau_{offx} \;\; \tau_{offy} \;\; \tau_{offz})^T \tag{3.13}$$

Where $\boldsymbol{f}_{offx}$ is the offset of force sensor on axis x. In order to represent the dynamic system in a state space form, we chose $\boldsymbol{\varphi}$ to be the state vector. Hence the observation vector is:

$$\boldsymbol{y} = (f_x^s \;\; f_y^s \;\; f_z^s \;\; \tau_x^s \;\; \tau_y^s \;\; \tau_z^s)^T \tag{3.14}$$

then equations (3.11) and (3.12) are considered the observation model, we write them as $\boldsymbol{h}$, then:

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\varphi}_k + \boldsymbol{v}_k \tag{3.15}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\varphi}_k, R_s^o, \boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{a}^o, \boldsymbol{g}^o) + \boldsymbol{n}_k \tag{3.16}$$

$\boldsymbol{v}_k$ and $\boldsymbol{n}_k$ are process noise and observation noise, respectively. The formulation of Kalman filters is included in the appendix.

It's easy to notice that the system is nonlinear, and the parameters in the system model, $R_s^o, \boldsymbol{\alpha}, \boldsymbol{\omega}$, and $\boldsymbol{a}^o$ are given by the sensors and kinematics of the robot. So the dynamics of the object is modeled into a dynamic system, and the inertial parameters are to be estimated. Kalman Filters are suited for this kind of problems. The difficulty here is that the function $\boldsymbol{h}$ is a time-variant nonlinear function. Among all the possible nonlinear filters, we chose the Unscented Kalman Filter (UKF) to solve the problem. As the same filtering technique is used for the next problem of multi-modal sensor fusion, we explain the UKF after the presentation of the model of the tracking problem.

3.2.2 Excitation Trajectories

In the previous section, we build a model for the estimation problem based on a dynamic system. The estimation is viable only when the system is observable. In the system model,

the observer model $\boldsymbol{h}$ has terms of acceleration and angular velocities, which require appropriate excitation trajectories. To find the excitation trajectory, two main issues should be considered:

- The trajectory should pass the stochastic observability test.
- The trajectory is used on a service robot, with obstacles in the working space. Hence a big movement should be avoided and the safety norms for the manipulation should be respected.

The problem to find an optimal solution is theoretically difficult. However, the design for excitation trajectories for robots has been studied in many papers. In this work, we won't try to prove or optimize the design but only use the well-established approaches. A popular approach to the design is based on sinusoidal trajectories in joint space.

$$s_i(t) = \sum_{k=1}^{N} \rho_{i,k} sin(2\pi k f t) + \delta_{i,k} cos(2\pi k f t) \tag{3.17}$$

$$q_i(t) = s_i(t) + q_{i,0} \tag{3.18}$$

In which, $q_{i,0}$ denotes the constant offset position which is used as a start point for the estimation movement. Parameters ρ and δ are to be set so that some predefined cost value is minimized.

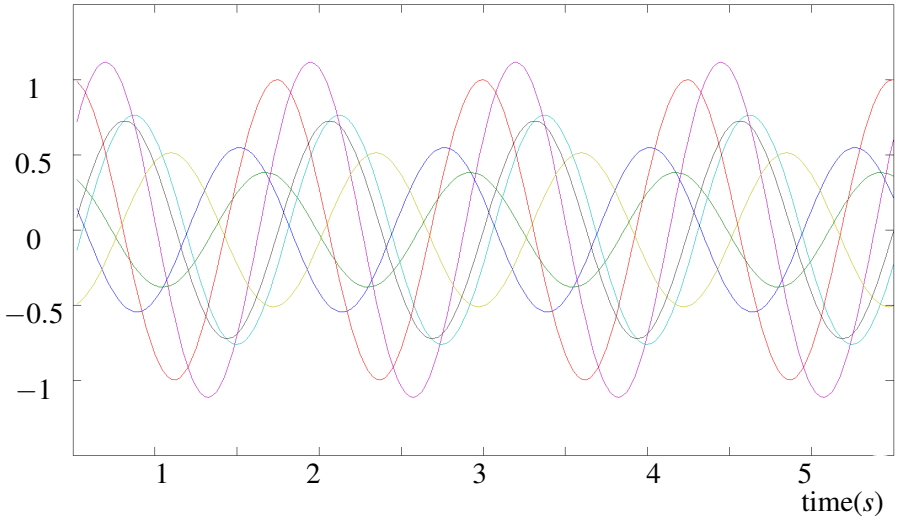


Figure 3.3: Generated sinusoidal trajectories on joints. Sinusoidal trajectories produce smooth trajectories for the robot arm. Time is in sequence of the trajectory controller, which runs at 100Hz. The vertical coordinate is joint positions.

Experiments on the robot also show that planned trajectory in joint-space for a basic manipulation, such as picking up an object, also make the estimation converge, although not as fast as the sinusoidal trajectories. The two sets of trajectories are all tested with

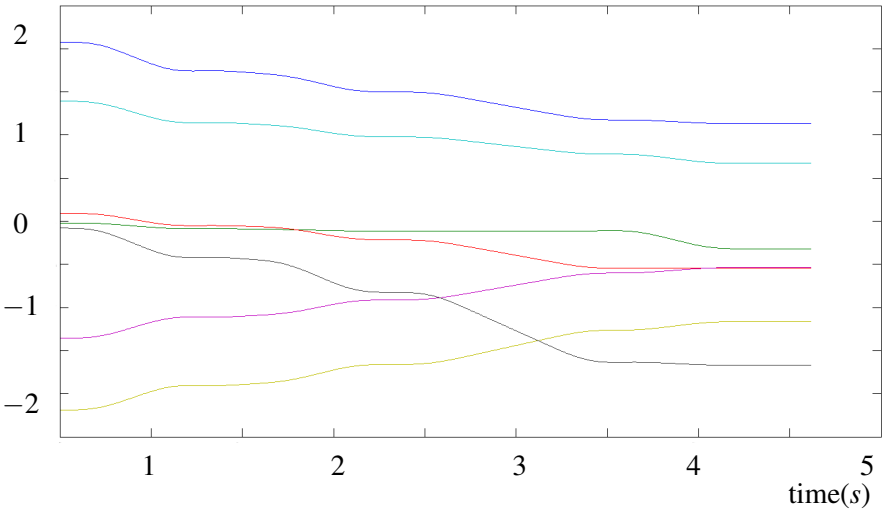


Figure 3.4: Trajectories on joints for a manipulation, planned by a path planner and then smoothed into trajectories. The method used to generate these trajectories are presented in Chapter V. Time is in sequence of the trajectory controller which runs at 100Hz.The vertical coordinate is joint positions.

offline data and the parameter estimation converges. When using the sinusoidal trajectories, the robot arm goes to a predefined position, and moves in a limited obstacle-free space until the estimation converges. While using the second, the robot picks up the object, goes to the rest position, and the estimation converges during the manipulation task.

3.2.3 Online Object Recognition

Object recognition is mostly based on vision sensors, some may use also depth sensors. Once the full set of inertial parameters is estimated, it can also be used as a feature set. There are many reasons to consider this set of features for object recognition. Firstly, there are situations in which vision is not always reliable, especially for a service robot. One example would be that one part of the manipulated object is hidden by the robot hand. Secondly, inertial parameters contain information that would not be possible to have by other sensors. One example is when we pick up a bottle to feel if it is full or empty.

One advantage of the proposed model in the previous section is that once the parameters are estimated, they are expressed in the frame of object itself. If the inertia matrix is expressed in the frame of the f/t sensor, the values of the inertia matrix change with the object position in the gripper. But when the inertia matrix is expressed in the frame of object, the different grasps only rotate the estimated inertia matrix, as shown by Figure 3.5.

In cases when the inertial parameters are not expressed in the object frame, they need to be propagated to the fixed frame. In our case, the parameter set can be used directly for the

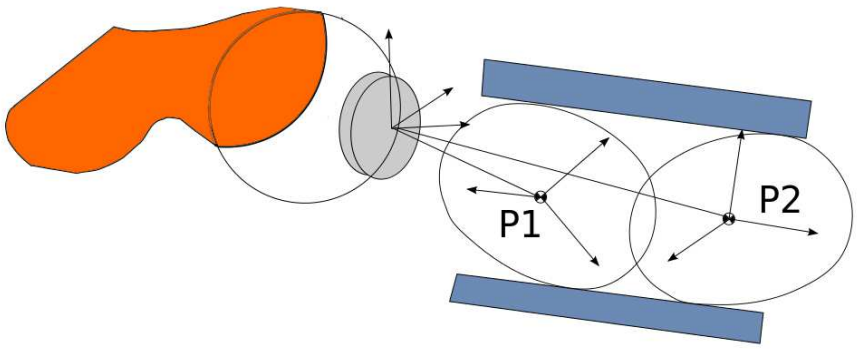


Figure 3.5: Object grasped with two different positions, $P1$ and $P2$. When the inertia matrix is expressed w.r.t. the object frame, the grasping position of the object only rotates the matrix.

classifier:

$$F = [m, \ \boldsymbol{I}] \tag{3.19}$$

Although the object frame has its origin point located at the center of mass, the rotation is still arbitrary. To overcome this problem, we can use only the principal moments of the inertia matrix. So then feature set is:

$$F = [m, \ I_1, \ I_2, \ I_3] \tag{3.20}$$

Where I_1, I_2 and I_3 are the eigenvalues of matrix $\boldsymbol{I}$, which can be computed easily by an eigenvalues decomposition.

3.2.4 Contact Forces/Torques Computation

Once the full set of inertial parameters are estimated, the object dynamics can be simulated. Then, the contact forces/torques are calculated as:

$$\boldsymbol{f}_{contact} = \boldsymbol{f}_{sensor} - \boldsymbol{f}_{inertia} \tag{3.21}$$

$$\boldsymbol{\tau}_{contact} = \boldsymbol{\tau}_{sensor} - \boldsymbol{\tau}_{inertia} \tag{3.22}$$

The structure of this calculation can be summarized as:

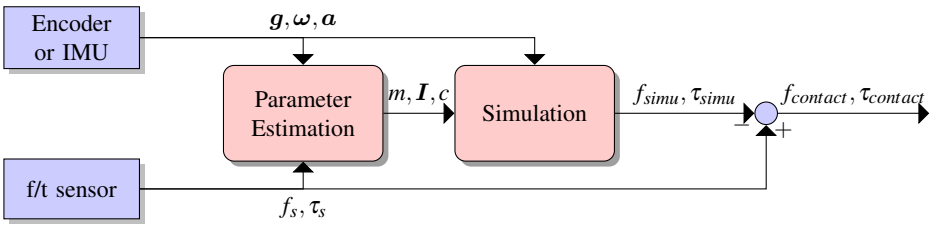


Figure 3.6: Contact Forces/Torques Computation. The gravitational vector $\boldsymbol{g}$, acceleration $\boldsymbol{a}$ and angular velocity $\boldsymbol{\omega}$ can be computed from the robot joint encoder and the kinematics of the robot, or by an inertial measurement unit (IMU). f_s and τ_s are measured forces/torques vectors.

The main challenge here is that the system should detect if the estimation really converges. One approach is to suppose that the excitation trajectories are predefined to guarantee the convergence. It is natural to think to give a criterion to detect the convergence, for example, when the estimated parameters are stabilized. The problem of this second approach is that the real parameters are unknown.

3.3 Force-Vision Fusion, a Multi-modal Tracking

Visual servoing is one of the most important functions for a robot to achieve sensor-based control. With visual servoing, the robot can track the object, human hand, or a planned virtual point in space. By the same principle, the robot may need to keep the distance to obstacles or to the human hand during the manipulation. All those tasks demand to know precise position of the objects or human body parts. There are many reasons requiring for a model-based tracking or obstacle avoidance.

- The sensors, mainly cameras and depth sensor, give a noisy estimation of the 6D position of objects. Constructing a model can reduce some noise.

- Some control policies may need to predict the future movement of objects. A model of object motion will make the prediction possible.

- One type of sensor may fail for some conditions. The cameras, for example, would have problem recognizing an object when it's held in the robot hand, while the forces/torques sensor can be used to recognize the object and localize it.

In this section, we will build a model for the motion of an object. To update the model, information from different sensors: vision and force sensor, can be used. The framework is based on Kalman filter, so for each source, an observation model is needed.

3.3.1 Process Model for Tracking

Visual tracking is a large and dynamic research area as reviewed in Chapter II. We do not try to solve the problem of recognition, instead, we use a simple marker based visual system. Nevertheless, the marker system gives quite noisy results in object localization, especially when lighting conditions vary, and when the marker is partially hidden from the robot cameras.

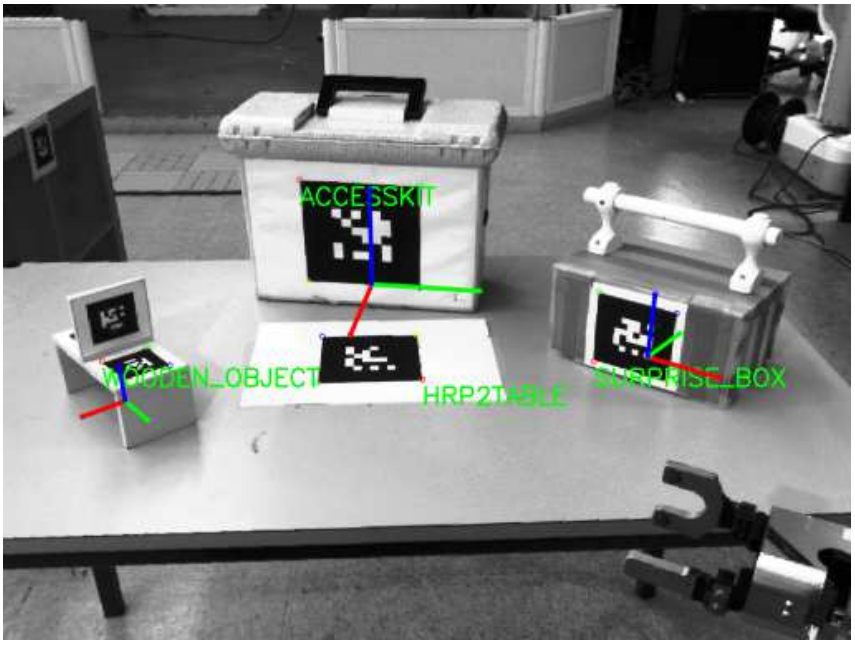


Figure 3.7: Vision based on tags.

The stereovision system computes the pose of the object relative to the 3D vision system, T_V^C, V means the 3D vision frame, and C the object. T_V^C is the transformation matrix:

$$T_V^C = \begin{bmatrix} & & & x \\ & R_{3\times3} & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

where R is the rotation matrix. The cameras are mounted on a platine which itself moves during the manipulations. Then the pose of the object in robot frame:

$$T_B^C = T_B^V T_V^O \tag{3.24}$$

In which, T_B^C is computed with the kinematics of the platine system. Although straightforward should the computation appear, a small error in this matrix could introduce a large error in object localization, which them can cause serious problem during manipulation. Calibration procedure shall be carried out. Hand-eye calibration techniques can be found in

literature, such as [Dornaika 98]. The coordinates are given in the robot base frame, which means, integrated all the frame transformations. The choice of robot base frame instead of world frame improves the level of precision in manipulation because it is often known with a higher precision than the robot base localization. To express the object pose in world frame, the computation should be:

$$T_W^C = T_W^B T_B^V T_V^C \tag{3.25}$$

In which, T_W^B is the robot base localization in the world frame. Errors in robot localization ere then introduced into object pose. On the robot *Jido*, manipulation and vision are all controlled in robot base frame. It should be considered in the world frame if manipulation should be carried out during navigation, which is not the case for the manipulations carried out during this thesis.

As we have chosen to employ a nonlinear Kalman filter for the sensor fusion, we need to build a model for the object movement with differential equations, or difference equations since the computation by computer programs will be discrete. We assume that the object held by a person or held by the robot is moving with a constant velocity with noises on position and velocity in the working space. This choice can be justified firstly by the fact that the application is on service robot. Secondly, try to build acceleration or even jerk into the model will not necessarily improve the efficiency since with the filtering, the velocity is constantly updated. Given that the object is often held by another agent, the constant velocity model is indeed efficient enough to track the object. The states of the model, $\boldsymbol{x}$ is of dimension 12:

$$\boldsymbol{x} = (\boldsymbol{x}_p^T \ \dot{\boldsymbol{x}}_p^T)^T \tag{3.26}$$

where $\boldsymbol{x}_p = (x\ y\ z\ \varphi\ \theta\ \phi)$. This vector tracks the translation and orientation.

Suppose that the tracking system runs with a time interval between updates of δt, at time t, the state equation of the dynamic system can be written as:

$$\boldsymbol{x}(t+\delta t) = A_{\delta t}\boldsymbol{x}(t) + \omega_t \tag{3.27}$$

In which, the matrix $A_{\delta t}$ can be written as:

$$A(\delta t) = \begin{bmatrix} I_6 & I_6\delta t \\ 0 & I_6 \end{bmatrix} \tag{3.28}$$

I_3 is the identity matrix of dimension 3. This transition matrix is based on the hypothesis that the target moves with a constant velocity on all dimensions during the time period. The process noise vector ω_t is supposed white Gaussian noises so that covariance matrix comes as:

$$E\{\omega_t \omega_{t+\varepsilon}^T\} = \begin{cases} Q(\delta t) & \varepsilon = 0, \\ 0 & \varepsilon \neq 0. \end{cases} \tag{3.29}$$

The term $Q(\delta t)$ should be calculated from the real noise. The values can be tuned to improve the dynamic of the filter, though in most of the cases, the filter tracks the process with a not so well tuned value for noise covariances. The special part for incremental orientation tracking lies in the observation model instead of the dynamic model. Between each time interval, the incremental orientations are simply the values of the state, since at the end of each update, they are reset to 0.

To avoid the problem of Euler angles that they are not unique representations, the angles are the incremental angles. The incremental angles means at the end of each step of tracking, the state of angles are reset to zero, and the observation of these angles are only the difference from the last update. At the end of Kalman filtering, this computation added:

$$(\varphi\ \theta\ \phi) = (0\ 0\ 0) \tag{3.30}$$

The absolute rotations of the object is stored outside the tracking filter, in the form of quaternions, which we write as:

$$\boldsymbol{q} = \{q_0\ q_1\ q_2\ q_3\} \tag{3.31}$$

3.3.2 Measurement Models

The application of Kalman filter framework needs a measurement model for each source of information. As we keep only incremental angles in the process model, the absolute rotation angles should be kept as value external of the filtering. The measurement models are the relationships between the incremental angles and the sensor information available on the robots. For our case, it is 3D vision, and force sensor with robot arm joint measurements.

Vision Measurement: For the compactness and clarity, we write the absolute rotation as quaternion. 3D vision gives position of the object in the robot frame. The computation from incremental angles to incremental quaternion is written as:

$$\Delta\boldsymbol{q} = h(\varphi\ \theta\ \phi) \tag{3.32}$$

In which, φ,θ and ϕ are the incremental Euler angles as in Equation 3.26.

$$\Delta\boldsymbol{q} = \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \\ \Delta q_3 \\ \Delta q_4 \end{bmatrix} = \begin{bmatrix} cos(\phi/2)cos(\theta/2)cos(\varphi/2) + sin(\phi/2)sin(\theta/2)sin(\varphi/2) \\ sin(\phi/2)cos(\theta/2)cos(\varphi/2) - cos(\phi/2)sin(\theta/2)sin(\varphi/2) \\ cos(\phi/2)sin(\theta/2)cos(\varphi/2) + sin(\phi/2)cos(\theta/2)sin(\varphi/2) \\ cos(\phi/2)cos(\theta/2)sin(\varphi/2) - sin(\phi/2)sin(\theta/2)cos(\varphi/2) \end{bmatrix} \tag{3.33}$$

which is a nonlinear function, and φ, θ, and ϕ are the states from the process model(3.27). This equation is the observation equation for the tracking process. And at each step, the

external quaternion for the absolute rotation bookkeeping is updated through the function:

$$\widehat{q} = \widehat{q} \bigotimes \Delta q \tag{3.34}$$

Which is the quaternion product.

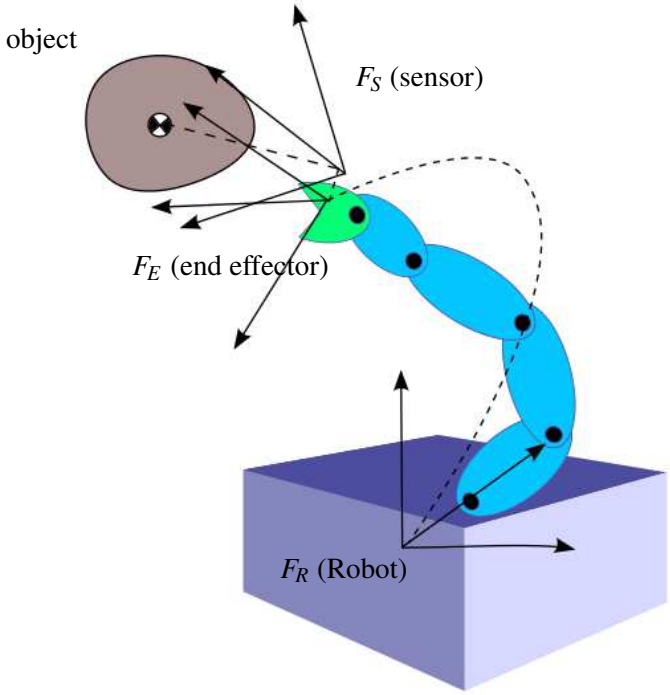


Figure 3.8: Coordination transforms are integrated into the measurement model.

Force-Torque Measurement: The force sensing process presented in this chapter provides the pose of object in the sensor frame, write it as P_S^C, and expanded by adding a 1 for the computation by transformation matrix. So $P_S^C = [c_x\ c_y\ c_z 1]$. The pose of the force sensor frame in the robot arm end effector frame is given when the sensor is installed, and written as the transformation matrix T_E^S, so that the pose of object in the robot frame is computed based on the kinematic model of the robot arm:

$$P_R^C = T_R^E T_E^S P_S^C \tag{3.35}$$

and the robot kinematic, T_R^E is updated with a rate higher than the fusion program. It should be mentioned though that the joint measurement is noisy and filters should be employed on the joint positions to filter the noise.

3.4 Nonlinear Kalman Filtering

In the preceding two sections, we build models for the two problems of force sensing and multi-modal tracking. These models are nonlinear. To use Kalman filtering for nonlinear systems, the classic Kalman filtering is no longer adequate. The Extended Kalman filter, approximating the nonlinear mapping by the first Taylor expansion, may give unsatisfactory results given that it is based on the assumption that the first order Taylor approximation produces small errors. (The Kalman filter and Extended Kalman filter are given in the Appendix).

The two problems in this chapter can all be considered as the estimation of the hidden state $\boldsymbol{x}_k$ of a discrete-time nonlinear dynamic system:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{w}, \boldsymbol{v}_k) \tag{3.36}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{w}, \boldsymbol{n}_k) \tag{3.37}$$

where $\boldsymbol{x}_k$ represents the system states and $\boldsymbol{y}_k$ the observed signal for the given system model. The process noise and observation noise are represented by $\boldsymbol{v}_k$ and $\boldsymbol{n}_k$, their covariance matrices are $\boldsymbol{R}_v$ and $\boldsymbol{R}_n$. In general, they do not necessarily to be additive. $\boldsymbol{w}$ are the fixed parameters. The system model $\boldsymbol{f}()$ and $\boldsymbol{h}()$ are assumed nonlinear and known. Kalman filter achieves a recursive minimum mean square error of $\boldsymbol{x}_k$. Given the observation $\boldsymbol{y}_k$, the recursive estimation for $\boldsymbol{x}_k$ is given as:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k) \tag{3.38}$$

$$\boldsymbol{P}_{x_k} = \boldsymbol{P}_{x_{k|k-1}} - \boldsymbol{K}_k \boldsymbol{P}_{\tilde{y}_k} \boldsymbol{K}_k^T \tag{3.39}$$

where $\hat{\boldsymbol{x}}_{k|k-1}$ is the optimal prediction at time k with all the observation at time $k-1$. $\hat{\boldsymbol{y}}_k$ is the optimal prediction of observation at time k, $\boldsymbol{P}_{x_{k|k-1}}$ the covariance of $\hat{\boldsymbol{x}}_{k|k-1}$, and $\boldsymbol{P}_{\tilde{y}_k}$ the covariance of innovation which is defined as $\tilde{\boldsymbol{y}}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$. The optimal prediction values are calculated by

$$\hat{\boldsymbol{x}}_{k|k-1} = E[\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{w}, \boldsymbol{v}_{k-1})] \tag{3.40}$$

$$\hat{\boldsymbol{y}}_k = E[\boldsymbol{h}(\hat{\boldsymbol{x}}_k, \boldsymbol{w}, \boldsymbol{n}_k)] \tag{3.41}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{x_k, y_k} \boldsymbol{P}_{\tilde{y}_k}^{-1} \tag{3.42}$$

The family of sigma-point Kalman filters (SPKF) are based on the idea that it is more reasonable to approximate a probability distribution than to approximate a nonlinear function. In equations (3.40) and (3.41), given the probabilistic distribution of $\hat{\boldsymbol{x}}_{k-1}$ (mean and covariance matrix), and nonlinear function $\boldsymbol{f}()$, SPKF uses a minimal set of deterministically chosen weighted sample points of the variable $\hat{\boldsymbol{x}}_{k-1}$ to capture the distribution. The set of values are then transformed through the nonlinear function $\boldsymbol{f}()$. The propagation needs no analytical derivatives but the known system nonlinear model. The posterior statistics are computed using the propagated sigma-points and associated weights.

The comparison of linearized transform with one type of sigma-point transformation,

called unscented transformation (UT) is given in Figure 3.9. In which, the green ellipse is the probability calculated by UT, while the purple one by linearized approach, and the black ellipse represents the real distribution. To be specific about the sigma-point transformation,

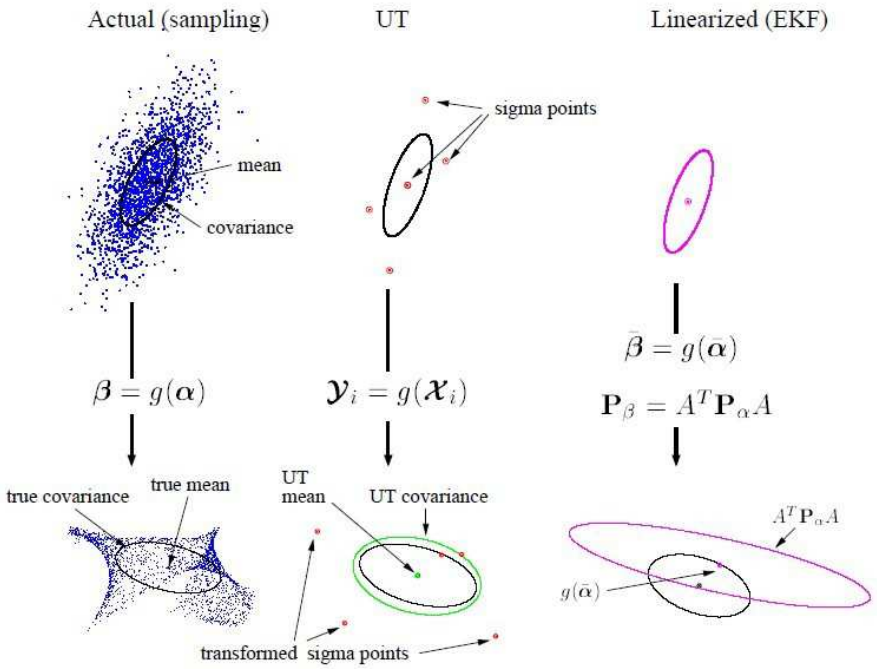


Figure 3.9: Compared to EKF, unscented transformation (UT) captures better the probability distribution through a nonlinear function[van der Merwe 04]

suppose we need to propagate a random variable $\boldsymbol{x} \in \mathbb{R}^L$ and $\boldsymbol{x} \sim (\bar{\boldsymbol{x}}, \boldsymbol{P}_x)$ through an arbitrary nonlinear function $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$. To calculate the statistics of $\boldsymbol{y}$, we find a set of $2L+1$ sigma-points $\mathcal{X}_i : i = 0, 1, ...2L$ where $\mathcal{X}_i \in \mathbb{R}^L$. The sigma-points are calculated using the following general method[Julier 96]:

$$\mathcal{X}_0 = \bar{\boldsymbol{x}} \tag{3.43}$$

$$\mathcal{X}_i = \bar{\boldsymbol{x}} + \zeta(\sqrt{\boldsymbol{P}_x})_i \; for \; (i = 1, 2, ...L) \tag{3.44}$$

$$\mathcal{X}_i = \bar{\boldsymbol{x}} - \zeta(\sqrt{\boldsymbol{P}_x})_i \; for \; (i = L+1, ...2L) \tag{3.45}$$

where ζ is a scalar scaling factor which determines the spread of the sigma-points around $\bar{\boldsymbol{x}}$, and $(\sqrt{\boldsymbol{P}_x})_i$ indicates the ith column of the square-root matrix of the covariance matrix. Once the sigma-points are calculated, they are propagated through the nonlinear function.

$$\mathcal{Y}_i = g(\mathcal{X}_i) \; for \; (i = 1, 2, ...2L) \tag{3.46}$$

$\bar{\boldsymbol{y}}$ and $\boldsymbol{P}_{yy}$ are computed by calculating the mean and covariance of the set of $\mathcal{Y}_i$.

The choice of Sigma-point set can be different. Based on different approaches, the SPKF is called differently, as discussed in [van der Merwe 04]. In our system, an algorithm called Spherical Simplex Unscented Transform is used. Spherical Simplex Unscented Transformation requires, for n dimensions, $n+2$ sigma points, $n+1$ of which lie on a hy-

persphere whose radius is proportional to $\sqrt{n}$. The weights on each point are proportional to $1/n$.

The algorithm is implemented in the frame of the Unscented Kalman filter (UKF) in general, which is an implementation of the SPKF. Algorithm used in this work is based on one type of Unscented Transform, called the Spherical Simplex Unscented Transform, and the details are given below.

Given a system model as in equations (A.13) and (A.14). The Spherical Simplex Unscented Kalman filter is implemented as below:

Function: Spherical Simplex Unscented Transformation (SSUT):
Calculate a set of sigma-points of a probability distribution.
1) Chose $0 \leq W_0 \leq 1$.
2) Chose weight sequence:

$$W_i = (1 - W_0)/(n+1)$$

3) Initialize vector sequence as:

$$\begin{aligned} \mathscr{X}_0^1 &= [0], \\ \mathscr{X}_1^1 &= [-1/\sqrt{2W_1}], \\ \mathscr{X}_2^1 &= [1/\sqrt{2W_1}] \end{aligned}$$

4) Expand vector sequence for $j = 2, \ldots L$, according to:

$$\mathscr{X}_i^j = \begin{cases} \begin{bmatrix} \mathscr{X}_0^{j-1} \\ O \end{bmatrix} & i = 0 \\ \begin{bmatrix} \mathscr{X}_i^{j-1} \\ -\frac{1}{\sqrt{j(j+1)W_1}} \end{bmatrix} & i = 1, \ldots, j \\ \begin{bmatrix} 0_{j-1} \\ -\frac{j}{\sqrt{j(j+1)W_1}} \end{bmatrix} & i = j+1 \end{cases}$$

and the general structure of the sigma-point Kalman filters [der Merwe 01] (see Appendix): The system states are augmented to be $\boldsymbol{x}_k^a = [\boldsymbol{x}_k^T \ \boldsymbol{v}_k^T \ \boldsymbol{n}_k^T]^T$. To simplify the notation, a is sometimes omitted.

Main program:

1. Initialization:

$$\begin{aligned}\hat{\boldsymbol{x}}_0^a &= E[\boldsymbol{x}_0^a],\\ \boldsymbol{P}_{x_0}^a &= Var(\boldsymbol{x}_0^a),\\ &= \begin{bmatrix}\boldsymbol{P}_{x_0} & 0 & 0\\ 0 & \boldsymbol{R}_v & 0\\ 0 & 0 & \boldsymbol{R}_n\end{bmatrix}\end{aligned}$$

For k = 1, ...:

2. Calculate sigma-points:

$\mathscr{X}_{k-1}^a$ is a set of sigma-points from $\hat{\boldsymbol{x}}_{k-1}^a$ and $\boldsymbol{P}_{k-1}^a$, computed by the function of *Spherical Simplex Unscented Transformation (SSUT).*

3. Prediction:

In the case of this thesis, this step is calculated as a linear Kalman filter since (3.15) is linear.

$$\begin{aligned}\mathscr{X}_{k|k-1}^x &= f(\mathscr{X}_{k-1}^x, \boldsymbol{\omega}, \mathscr{X}_{k-1}^v)\\ \hat{\boldsymbol{x}}_{k|k-1} &= \sum_{i=0}^{2L} W_i^m \mathscr{X}_{i,k|k-1}^x\\ \boldsymbol{P}_{x_{k|k-1}}^x &= \sum_{i=0}^{2L} W_i^c (\mathscr{X}_{i,k|k-1}^x - \hat{\boldsymbol{x}}_{k|k-1})(\mathscr{X}_{i,k|k-1}^x - \hat{\boldsymbol{x}}_{k|k-1})^T\end{aligned}$$

4. Measurement-update:

$$\begin{aligned}\mathscr{Y}_{k|k-1} &= h(\mathscr{X}_{k|k-1}^x, \boldsymbol{\omega}, \mathscr{X}_{k-1}^n)\\ \hat{\boldsymbol{y}}_{k|k-1} &= \sum_{i=0}^{2L} W_i^m \mathscr{Y}_{i,k|k-1}^x\\ \boldsymbol{P}_{\tilde{y}_{k|k-1}} &= \sum_{i=0}^{2L} W_i^c (\mathscr{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})(\mathscr{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})^T\\ \boldsymbol{P}_{x_k y_k} &= \sum_{i=0}^{2L} W_{i,j}^c (\mathscr{X}_{i,k|k-1}^x - \hat{\boldsymbol{x}}_{k|k-1})(\mathscr{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_{k|k-1})^T\\ \boldsymbol{K}_k &= \boldsymbol{P}_{x_k y_k} \boldsymbol{P}_{\tilde{y}_{k|k-1}}^{-1}\\ \hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1})\\ \boldsymbol{P}_{x_k}^x &= \boldsymbol{P}_{x_{k|k-1}}^x - \boldsymbol{K}_k \boldsymbol{P}_{\tilde{y}_{k|k-1}} \boldsymbol{K}_k^T\end{aligned}$$

where L is the dimension of the augmented state, $\boldsymbol{R}_v$, $\boldsymbol{R}_n$ are process noise covariance and observation noise covariance, respectively. W_i^m and W_i^c are the scalar weights that we choose in the SSUT function, so $W_i^m = W_i^c$.

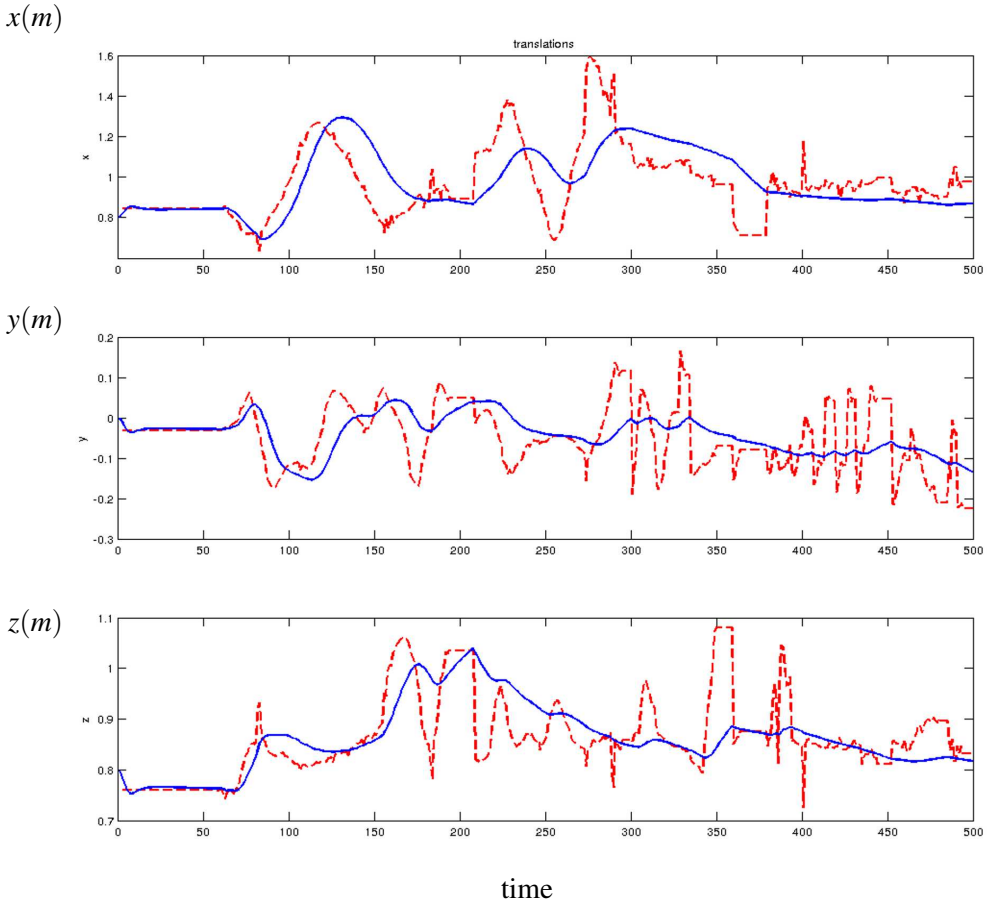


Figure 3.10: Visual Tracking during 10s on translation before and after filtering. The red dashed curves are results from 3D vision system before filtering, and the blue lines are filtered. The data is acquired in bad lighting conditions to show the dynamic of the filter. Time is expressed in sequence, with the period of 0.02s.

3.5 Simulation and Experimental Results

The force sensing procedure, including the estimation of inertial parameters, the simulation of rigid body dynamics, and the elimination of the inertial forces have been tested in simulation and with offline data. The complete procedure, together with object identification has not been fully tested online till the writing of this document because of difficulties with the sensor integration into the robotic system. However the tracking has been implemented and tested online with 3D vision and position estimation from the robot joint arm positions and end effector force sensing.

The 3D vision system computes instable tracking results when the lighting condition is poor, which is shown in Figure3.10 and rotations in Figure 3.11. KF to improve target tracking by reducing the noises have been discussed in various works in literature (and performs as a low-pass filter), and here we see that the filter can largely reduce the oscillation

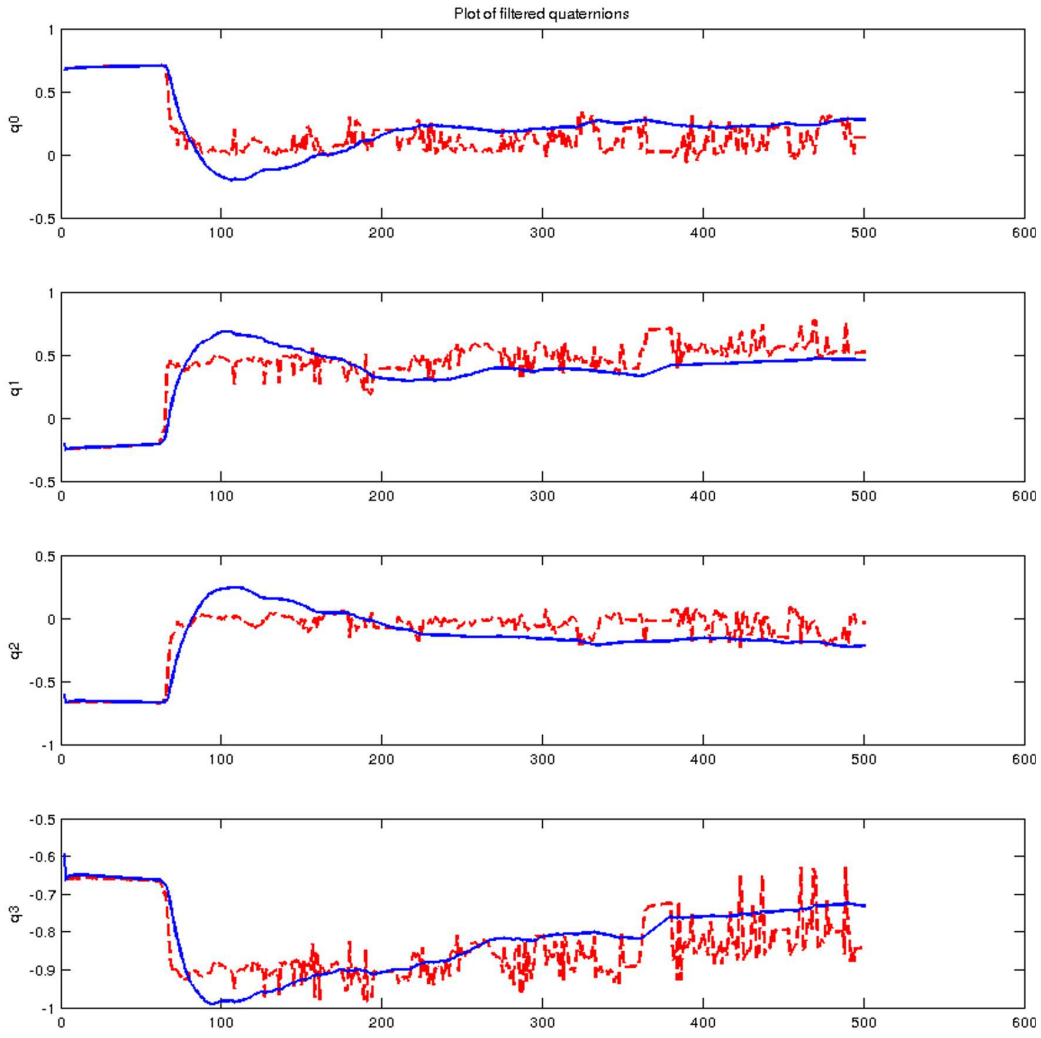


Figure 3.11: Rotation tracking by quaternions during 10s. The red dashed lines are quaternions before filtering, and the blue lines are filtered results. The data is acquired in bad lighting conditions to show the dynamic of the filter. Time is expressed in sequence, with the period of 0.02s.

when the 3D vision gives poor results. The oscillation can destabilize it if the control is based on visual tracking results.

3.6 Conclusion

In this chapter, we discussed two problems of sensor fusion. The first part is force sensing during manipulation, which allows estimating the dynamic parameters of the load or of the manipulated object using a nonlinear Kalman filter. Then these parameters are used to simulate the inertial forces/torques.

The inertial f/t are then substituted from the sensored f/t, giving the contact f/t between the object and the environment in which the robot works. The contact f/t will be used to detect events and collisions, and the estimated object model can be used for object recognition and to localize the object in the local frame. This localization is then fused in the second part with vision, by building a model for the motion of the object. Fusion of force sensor, vision and robot arm joint positions can produce robust tracking results. To achieve this tracking process, incremental angles are used in the model, while quaternions are used to keep track of the absolute angles of the object.

How the tracking process performs when switching between different sensory inputs for the tracking is still to test experimentally as we have only offline results till the moment this document is written.

4

Machine Learning for Manipulation

There is only one thing more painful than learning from experience and that is not learning from experience.
—- Archibald MacLeish

Abstract. In this chapter, we present the design of an object to teach the robot how to exchange an object with its human partner, and the techniques for learning. The object is designed to teach various aspects of object exchange, but this thesis is limited on the work of grasping-releasing synchronization. This problem is solved using classification algorithms for different types of collisions between the manipulated object and the environment. We propose to extract the information from time-frequency analysis using Wavelet Packet Transformation (WPT). Finally, the features are selected and passed to a classification tool, the Relevance Vector Machine (RVM).

4.1 Introduction

Mobile manipulators start to work outside of laboratories, and the cooperative manipulation between humans and robots in environment of households and workspace raises new challenges like the human partner may not have experience to manipulate a robot as the researchers do. Instead, they expect the robots to act more like human. To achieve this goal, a device was designed to teach robots the ability to exchange objects naturally with humans. With this device, the human-human object exchange is studied, and then the model

is implemented on the robot to improve human-robot object exchange.

4.1.1 Trajectory Control with Continuous Classification

The control of our robot is centered around a trajectory controller based on Online Trajectory Generation (OTG). During the handing-over movement, several situations concerning the contact forces will occur:

- When the robot is the giver, the robot approaches and tracks the exchange point, which is defined in the frame of its human partner. The robot should stop the movement and open the gripper when it detects that human has grasped the object. This enables a dynamic and reactive object exchange.
- The manipulated object could collide by accident with the human hand, or with the environment. There could also be unsuccessful grasps by human. For safety issue, the robot should react differently, depending on the type of collision or failed grasps.

Our first objective of learning is to use continuous classification on the force information history of the past second, for example, to identify what the situation is. We want the robot to be able to use classification to know the difference between the types of events, and more importantly, when to open the fingers if it is a grasp by human partner. We define firstly three classes: collision, grasp, and all the rest cases: pure noise, or grasp/collision but not yet the moment to react. More classes could be easily added, like collision with different environments. In the following, the chapter is organized as:

- The device to record human-human object exchange is presented.
- The wavelet packet transform and feature selection techniques are introduced.
- The Relevance Vector Machine is discussed as the classifier.
- Some results are then shown and followed by a discussion.

4.2 Bidule: a Device for Manipulation Learning

The model to learn aims to be used on a robot arm with a 6D forces/torques sensor mounted on the wrist, although the model can work with the estimated contact forces/torques too. The f/t sensor is common on service robots, and can be used for different purposes, as we discussed in the previous chapter. To detect events, we measure the interaction force during the exchange. As we wish the robot to execute the exchange as naturally as possible, an “intelligent” sensorized device named Bidule was built, which can be used to record information when the exchange is carried out between two people. For the grasp detection, we use wavelet analysis and classification techniques to study object exchange between

people. The classification model learned from human experience is to teach the robot the capability to know if the human grasps the exchanged object and to decide when to react. The same model can also be used for other manipulations, like putting objects on a table. The device is also designed to help to learn the trajectories and the force control policies, although the details of the last two parts will not be covered in this document. The main objective of this document is to present a set of techniques to extract efficiently patterns from transient signals for robotic manipulations, which can be from different sensory sources such as force sensor, tactile sensor or others.

The choice of f/t sensor is for that the learned model can be directly used on robots equipped with a wrist f/t sensor. The f/t sensor is common on mobile manipulators, and for robots without a f/t sensor, the wrist f/t is often estimated.

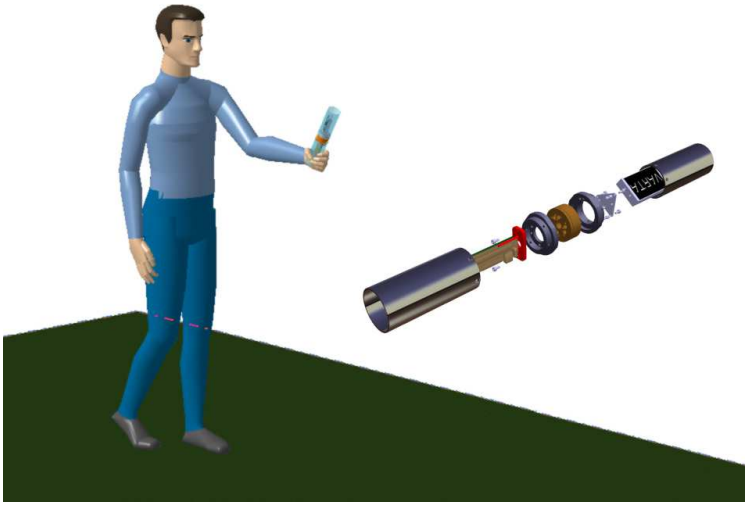


Figure 4.1: Bidule: a tool to record object exchange between people

Bidule is built around an embedded micro-computer, a WiFi link, and a 6D force/torque sensor. An Inertial Measurement Unit (IMU) is used to compensate inertial forces and to record the movement of Bidule. The shape and the size were chosen to be adapted to the human hand (Fig.4.2). The two tubes can be grasped by humans and transmit the external forces to the sensor.

The force and IMU analog signals are amplified, filtered and re-scaled before an Analog to Digital Converter (ADC) digitalizes these signals. The link between the computer board and the acquisition board is done via a SPI link running at 3 Mbits/s available on a 60 pins connector. Two additional bronze rings can be added around the object to increase the Bidule mass (Fig. 4.3). This enables us to study different exchange forces when the mass of exchanged objects varies.

During manipulation, the f/t sensor records not only the contact forces of Bidule with the environment, but also the inertial forces of Bidule. Dynamic compensation is needed

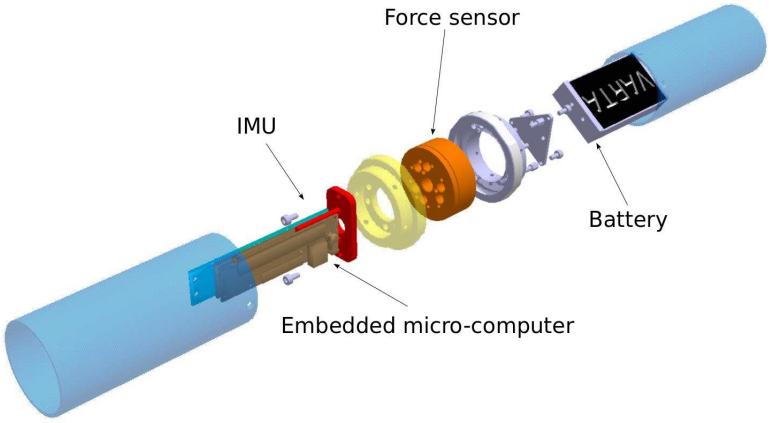


Figure 4.2: Bidule design with the 6D force/torque sensor (orange), the handle (blue) are fixed on rings screwed to the sensor, the computer is on the left side and the battery with the power supply board on the right side.

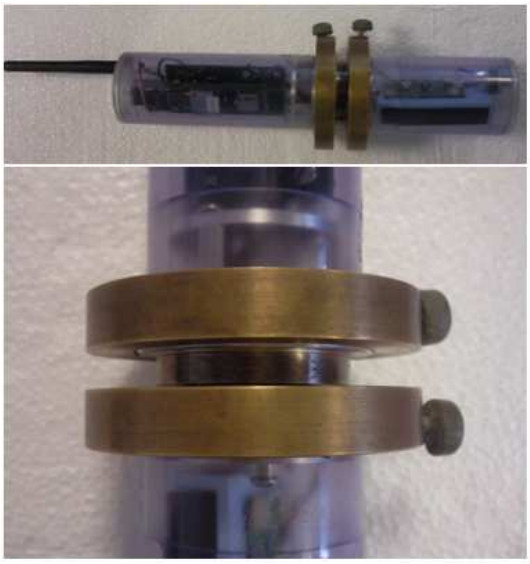


Figure 4.3: Bidule with additional mass

to estimate firstly the inertial parameters (the mass, center of mass, and inertia matrix) of the upper and bottom parts of Bidule. The reader can refer to the previous chapter to see how the dynamics of the load on a wrist f/t sensor is modeled. When the inertial parameters and sensor offset are estimated, the dynamics are compensated. In this chapter, we suppose that the recorded signals are directly the contact forces/torques between Bidule and the environment.

Bidule is designed to learn three aspects of object manipulation for a service robot:

- To extract the rich information from the f/t sensor during exchange or when the robot end effector is in contact with the environment, either for human grasp or collision

with different obstacles.

- The trajectory defining the exchange movement adapted to the mass and types of the manipulated object.
- The force control policies for robot during object exchange with human partner.

In this document, we focus on the first part. The trajectory learning and force control policies learning are work in progress.

4.3 Wavelet Analysis

4.3.1 Wavelet Packet Transformation (WPT)

The time series of forces are irregular, depending on the differences of the human partners and of the manipulated objects. For example, while the object is big, people would grasp it stronger and slower, which means the contact forces would be bigger in magnitude and last longer. Although some simple statistics on the f/t signals could also gain some useful results, we decide to learn the rich time-frequency information in the signals so we can have robust control policies for the robot.

In Fig.4.4, we see the forces produced when the object is grasped by a person and the forces during a typical collision between Bidule and a table. They show different forms of vibration. We see that grasp exhibits a double peak, and collision has a higher-frequencies pattern. For either of the two classes, neither the magnitude nor the duration of the vibration could be easily used to identify the signal. However some characteristics can be used to achieve the classification, which is what we try to extract by machine learning.

The choices of techniques are oriented to obtain a computationally efficient and hence fast classifier so it can be run at a high rate, together with other algorithms such as Online Trajectory Generation and online collision checking.

The contact forces are transient signals, for which wavelet based methods have been established to be a reasonable choice to extract time-frequency information. Wavelet Packet Transformation is an algorithm to decompose discrete signals into subbands with different time-frequency resolutions [Mallat 08]. Each step of decomposition is implemented as being filtered by a two-channel filter bank, together with downsampling operations. Fig. 4.5 shows an example of a signal of size sixteen decomposed by a Haar filter bank[A.Jensen 01]. The original signal is represented at level zero of the tree. At level $j = 1$, the blue part is obtained by lowpass-downsampling operations of the signal, while the pink part is obtained by highpass-downsampling operations. The same computation is then executed on these two signals, decomposing the signal into four subbands (level $j = 2$). The signal is decomposed until $j = 4$, which is the maximum levels J for a signal of size 16. If the size of original signal is D, we have $J = log_2 D$.

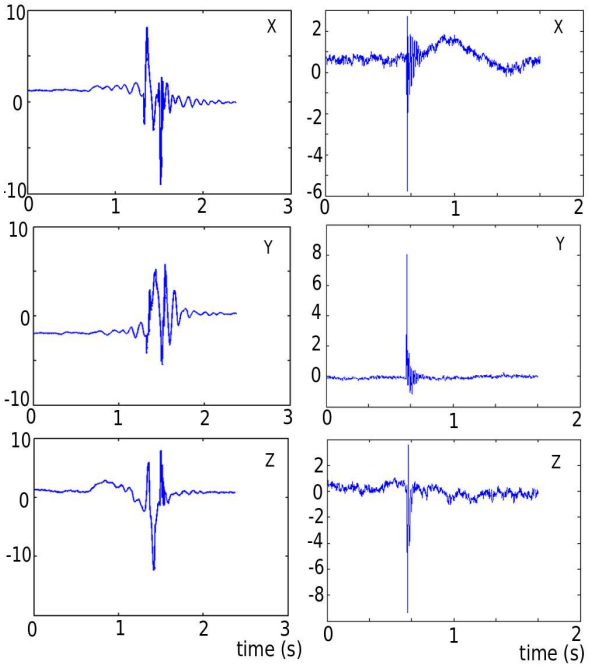


Figure 4.4: Typical patterns of measured contact forces (in N). Left: measured forces when Bidule is grasped by a person, while hold by another one. Right: measured forces during a collision with a table.

Notice that in Discrete Wavelet Transformation (DWT), only high frequency part is recursively decomposed. Then it can be seen as one subset of the WPT. The problem with DWT is it may lose the possibility to highlight some features. As reported in [Coifman 92], only some signals have the information focused in this fixed representation. And [Learned 95] shows a clear example of a transient signal where some patterns are captured by a WPT, but they will not appear in the results of a DWT.

In our system, as the transformation should be computed in real-time, we chose to implement each step of the wavelet transformation in Lifting scheme. The Lifting scheme reduces the computation complexity, requires less memory, and maintains always perfect reconstruction. The next section gives the definition for the Lifting scheme.

To illustrate the time-frequency pattern of a signal, we computed WPT of a chirp signal $y(t) = sin(k\pi t)$ by Daubechies wavelets ($db4$). Then the fifth level is chosen from the tree and ordered by frequency. The results are presented in Fig.4.6, in which the coordinates of time and frequency is without units because they lost the physical meaning, and magnitude plots the function $log(1+x^2(i))$, in which $x(i)$ are the wavelet coefficients in level 5. We can see clearly the pattern of an increasing frequency.

For this signal with a clear increasing frequency, it is easy to illustrate its pattern by constructing a time-frequency plot, but to classify complex and variant contact forces, we need techniques to find the best pattern representation from the WPT.

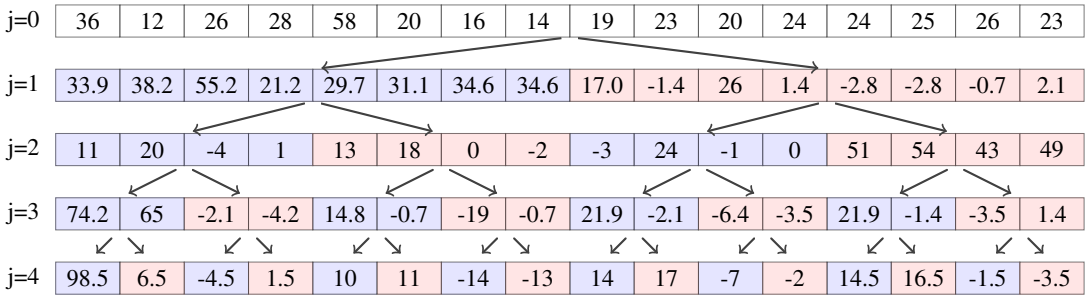


Figure 4.5: Wavelet Packet Transformation decomposes the original signal recursively into a tree structure, here through the Haar wavelet filter bank. Each subband is decomposed into a high-frequency part (pink) and a low-frequency part (blue).

4.3.2 Lifting Scheme

The WPT has been implemented by Lifting scheme, which is a technique for designing and performing the discrete wavelet transform. It has been firstly introduced by Daubechies *et al.* ([Daubechies 98]). Compared to the convolution computation, the technique keeps perfect reconstruction, and speeds up the computation. It is used in the JPEG2000 for DWT. For the basics of the so called first generation wavelet, readers can refer to the Appendix, which shows wavelet functions by scaling and translating a function called mother wavelet. The transform is computed by convolutions. As a recursive algorithm, each step of lifting scheme consists in three stages of computation: split, predict, and update (although split does not contain any calculation, so only two steps of computation are performed.) For an implementation of one step lifting (for example, the Haar wavelet transform), the computation can be represented as in Figure 4.7. The basic idea behind the complicated mathematic development ([Mallat 08]) can be summarized in three steps: split, prediction, and update.

split: The entries are split into even and odd items. This step does not require computation in real implementation.

prediction: If the signal contains some structure or pattern, then some correlation can be expected from a sample and its nearest neighbors. We can use then one sample in a set (the even ones) to predict the one in the other set. This prediction depends, of course, on the type of wavelet we try to implement. We would expect the neighbor has the same value. We then replace the value at $2n+1$ (the odd set) with the correlation to the prediction, which is actually the difference between two nearest neighbors for a Haar wavelet:

$$\gamma_j[n] = \lambda_{j+1}[2n+1] - \lambda_{j+1}[2n] \tag{4.1}$$

in which j is the step in the recursive algorithm. In general, the prediction can be expressed as:

$$\gamma_j = odd_j - P(even_j) \tag{4.2}$$

Which means that in signal γ, each entry is one odd sample minus some prediction based on

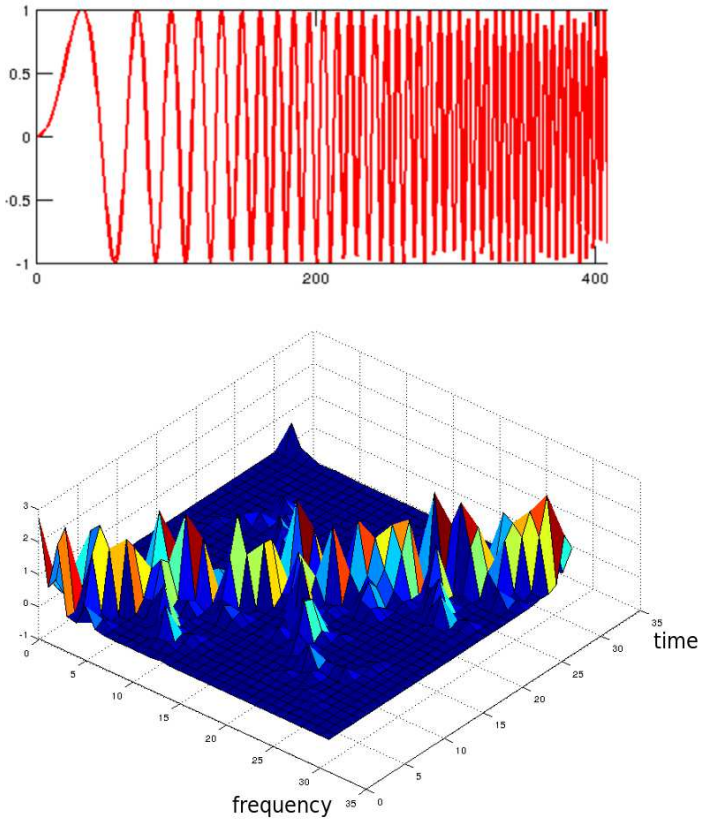


Figure 4.6: Linear chirp (top) and it's time-frequency pattern (bottom), constructed from level 5 of a WPT tree structure.

a number of even samples. The prediction (and number of samples) is defined by the type of wavelet we implement.

update: In the Haar implementation, given an entry, we predicted that the next odd entry has the same value, and stored the difference in the prediction step. We then update the even entry to reflect the knowledge of the signal. This can be given as:

$$\lambda_j[n] = \lambda_{j+1}[2n] - \gamma_j[n]/2 \tag{4.3}$$

or in general:

$$\lambda_j[n] = even_j + U(\gamma_j) \tag{4.4}$$

Again, U is defined by the type of wavelet we implement. Readers may refer to [A.Jensen 01] for the relations between lifting and filters, and the MATLAB code of some implementations of lifting scheme for different wavelets.

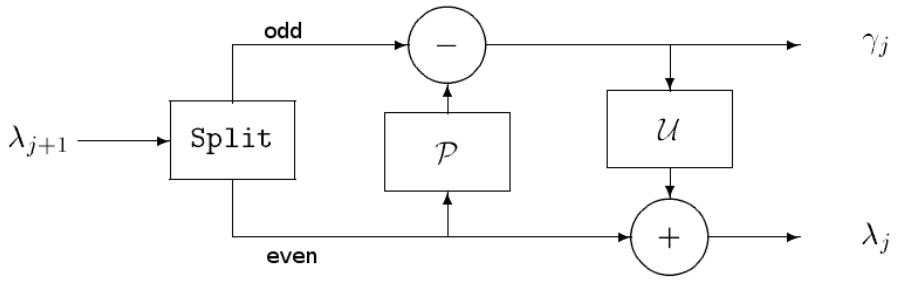


Figure 4.7: Lifting scheme computation for a one step transform

4.3.3 Feature Formation

Because of the redundancy of the WPT tree, several feature representations exist: part of the decomposition coefficients, statistical information on the coefficients, or the energy map[Fatourechi 04][Birbaumer 00][Xue 03]. The whole set of coefficients captures all the in-class variations and noise. And it would require a large data set to achieve convergence for the optimization process of learning. Two reasons demand us to select the features properly and reduce the size of the feature set. Firstly, building a large data set from manipulations with different candidates and different situations is a tedious task. Secondly, the classifier must run online at the same frequency as the controller. We propose to differentiate in time and in frequency separately by combining a part of the coefficients and the energy map.

The Relative Energy Map is the energy distribution map w.r.t. time and frequency for a signal. The energy map is used to capture the frequency information. Each level of the WPT decomposes the signal into a time-frequency representation with different time-frequency precision. For each subband in the tree, as shown in Fig. 4.8, an energy value is computed from all the coefficients in the subband. If we name one subband b and the coefficients $x(k)$ in this subband, then its energy e_b is computed as:

$$e_b = \sum_k (x(k))^2 \tag{4.5}$$

The energy is calculated through the whole tree, until the level that gives enough frequency precision, which should be defined by prior knowledge. Then they are all normalized to the total energy (e_0) of the signal in the time window, which means:

$$E_b = \frac{e_b}{e_0} \tag{4.6}$$

in which, e_0 is the total energy of the signal. For a sample to learn, we write this feature set as $\boldsymbol{E} = \{E_b\}$, which includes the relative energy in all the selected subbands. We notice that a large part of the feature produced by this method are zero, which is no surprise since

wavelets produce a sparse representation of the original signal (see [Mallat 08] for more details about the sparsity of WPT). We should then chose the most discriminative features from $\boldsymbol{E}$ to further reduce the size of feature vector.

The energy map is independent of the magnitude and the duration of vibration in the original force signal. But with the relative energy map, the time information is lost. It can be noted that the relative energy map is already scaled between 0 and 1.

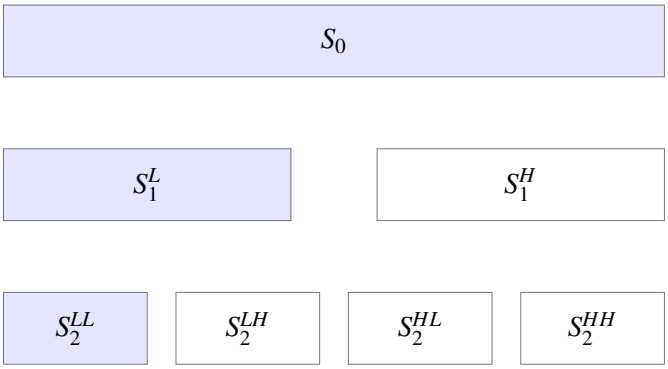


Figure 4.8: Subbands for the WPT tree of total level 3. L means that the subband has been filtered by a low-pass filter and then downsampled, H means filtered by a high-pass filter and downsampled. The subbands in blue are the low-frequency part at each level.

Coefficients of One Subband are selected to capture the global form of the signal. In our case, the objective is twofold: to keep the information of the changing in weight when the receiver starts to hold the exchanged object, and to deduce the right time to open fingers or to react for the robot. The robot should differentiate two signals as shown in Fig 4.9 where the same signal is shifted in time. We chose the coefficients from the first subband of a level in the WPT tree. It contains information of low frequency part of the signal and with a reduced size. We write this feature set as $\boldsymbol{S}$. In Fig. 4.8, the candidates are in the blue color. Since each level contains such a low frequency subband, the choice of the level is decided by compromising the classification error and the feature size produced. The higher level we chose for this subband, the less it is precise in time. The coefficients are then normalized to between 0 and 1 as the relative energy map.

The choice of the first subband of a level is based on prior knowledge on signals in all classes of the problem at hand, this subband does indeed capture well the peaks of the original force signals. For some signals, which contains only high frequency components, the choice of subband needs to be made by feature selection algorithms.

4.4 Feature Selection

Our objective is now to reduce the feature set. We propose to use the Fisher Linear Discriminant Analysis is then used to select features and to reduce further the dimension of the classification model.

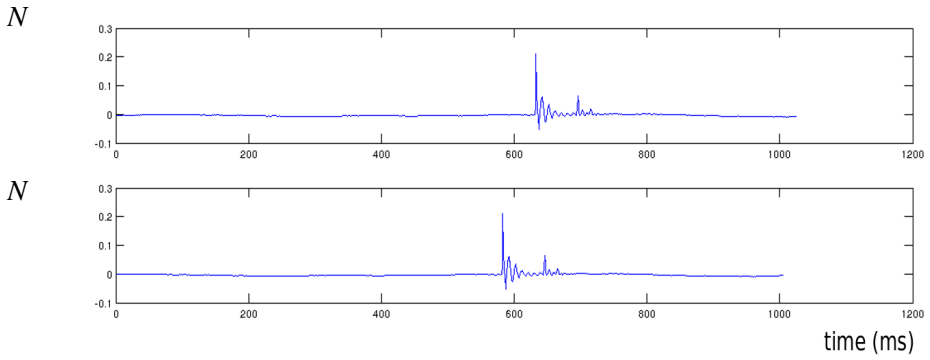


Figure 4.9: The classifier should be able to make a difference between the two signals above in order to find the right time to react to an event detected by the f/t sensor.

4.4.1 Fisher Linear Discriminant Analysis

Fisher-LDA is a tool used for classification and dimension reduction. One of the most popular tool for dimension reduction is the principle components analysis (PCA). The two will be compared and we will discuss why LDA is preferable for our problem. PCA is an orthogonal linear transformation that transforms the data to new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. Suppose that we have a data set $\boldsymbol{X} \in \mathbb{R}^{N\times P}$, in which $x_i \in \mathbb{R}^P, 0 \leq i \leq N$ denotes one data point, with zero mean (for data set with nonzero means, they can be normalized). So the N rows represent N data points, and a data point includes P features. The mapping is defined as finding a matrix of projection W, by which:

$$\boldsymbol{Y} = W\boldsymbol{X} \tag{4.7}$$

such that the individual variables of $\boldsymbol{Y}$ considered over the data set successively inherit the maximum possible variance from $\boldsymbol{X}$, with each loading vector w_i in W constrained to be a unit vector. PCA is an unsupervised technique and, as such, does not include label information of the data when performing dimension reduction. LDA, instead, does not try to maximize the variance of the whole data set, but try to project the data set, so that the classes of data are more separated. As PCA and LDA can both be seen as a rotation of the coordinates in which data are presented, the difference can be summarized as in Figure 4.11.

Fisher-LDA considers finding the best weight $\boldsymbol{w}$ by maximizing the objective:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} \tag{4.8}$$

where S_B is the “between classes scatter matrix”, and S_W is the “with classes scatter matrix”.

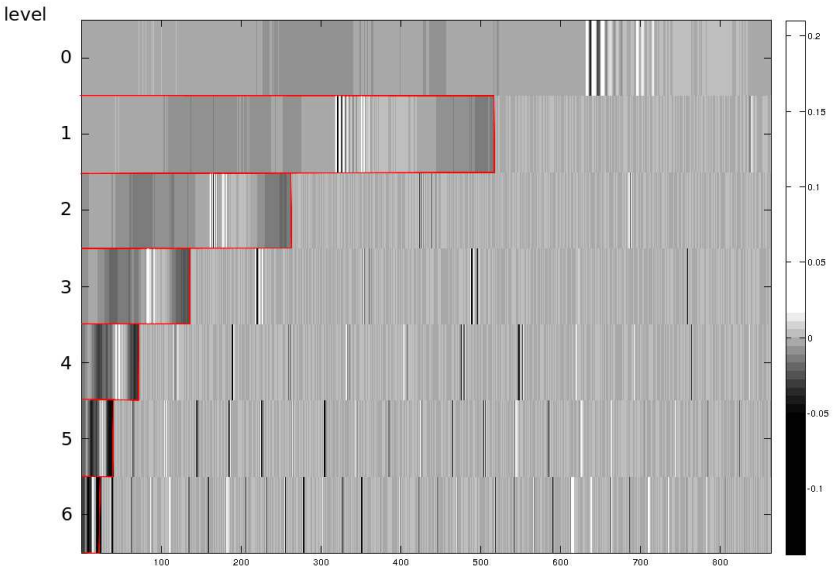


Figure 4.10: Part of the WPT tree plotted for the signals in Fig.4.9. The first subbands (highlighted by red rectangle) capture the peak in signal (the peaks are in white color). The WPT is computed with Symlets 5 wavelets (*sym5*). The zeros in the tree show also the sparsity of the WPT.

The value of J means the separability of classes for a set or a subset of features. The two values are computed as:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{4.9}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{4.10}$$

where $\bar{x}$ is the overall mean of the data points, c denotes the class label, and μ_c the mean value of the in class data. An important property about the projection J is that it is invariant w.r.t. rescaling of the vector $\boldsymbol{w}$. This is useful because we can always chose $\boldsymbol{w}$ such that $\boldsymbol{w}^T S_W \boldsymbol{w} = 1$. The problem of the maximization of J is then transformed into an optimization problem defined as:

$$min_w \quad -\frac{1}{2}\boldsymbol{w}^T S_B \boldsymbol{w} \tag{4.11}$$

$$s.t. \quad \boldsymbol{w}^T S_w \boldsymbol{w} = 1 \tag{4.12}$$

4.4.2 Fisher LDA for Feature Selection

Here, we define the criterion to select the best features in our feature set of the relative energy map $\boldsymbol{E}$. The criterion is reformulated into equation 4.13. It is calculated for every feature in E to evaluate the separability measure [Gu 11] of that feature. As WPT produces

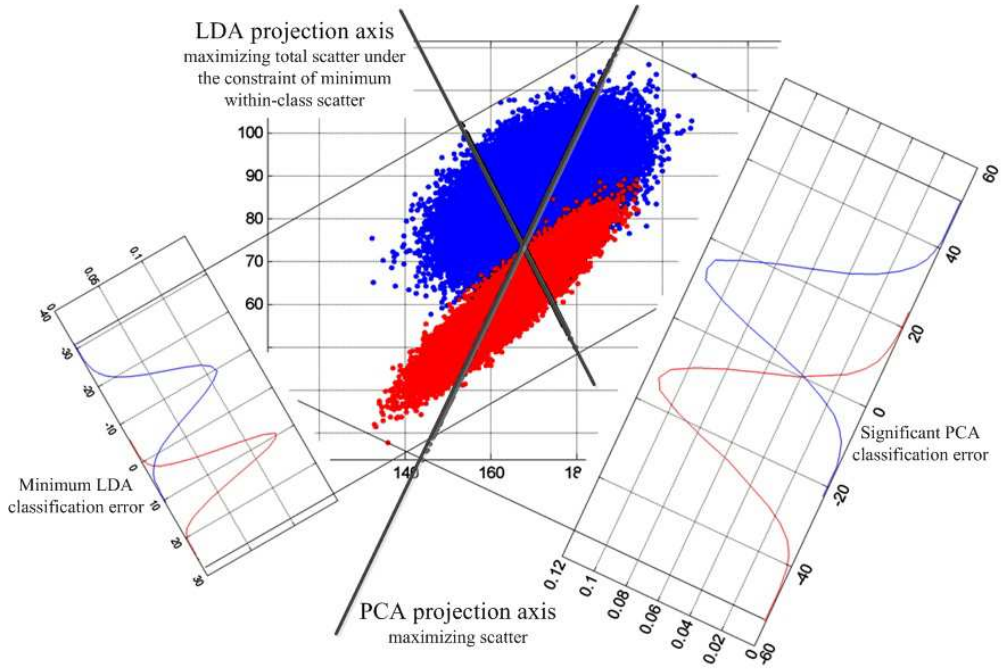


Figure 4.11: Difference between PCA and LDA: LDA maximize the separability between classes, while PCA projects data while maximizing the scatter of all data points.

a sparse representation of the signal, which means many terms in the tree are nearly zero, it is reasonable not to include those features, which are invariant between classes.

$$J = tr(S_W^{-1} S_B) \tag{4.13}$$

Compared to the definition 4.8, we will see that this defines the separability, and omits the projection vector. The fact that we have not simply used LDA as classifier is because our problem is not linearly separable, and also because of a lack of data. To find a classifier for an overlapping data set, of which few prior information is lacking, we choose a kernel based classifier. Then why does this criterion make sense while we use another tool for classification? As we want to reduce the size of the feature set, we observed that most features are zeros, which is reasonable because WPT produces a sparse representation of the original signal, and so the energy map. However for this sparse representation, the zero terms are not really zeros, because of the noise in the signal. In the results, we will see that the significant terms are very evidently distinguished from the near-zero terms, if measured by the LDA criterion.

During the object exchange manipulation, the orientations of f/t changes in different situations: the posture of the robot and of people, the exchange positions, and more. Not including orientations for the event detection simplifies the problem. In fact, the f/t are expressed in the local frame of the sensor, which makes the orientations more difficult to use for the classifier. Learning a model with orientations will also be much more complex

and produce a model of much higher dimension, resulting in a slower classifier. While the orientations of f/t are excluded from the feature set, the feature set building process can be summarized as:

- The WPT tree representation is built from force F and from torque T, F is computed as $F = ||Fx + Fy + Fz||$, and T is computed as $T = ||Tx + Ty + Tz||$.
- The relative energy map is computed for the WPT.
- It is then combined with a subband from the tree. The subband is also scaled to range from 0 and 1, matching the same as relative energy map.
- Fisher LDA criterion is used to reduce the size of the feature set of energy map $\boldsymbol{E}$ and the level from the subband $\boldsymbol{S}$ selected as a compromise between the classification accuracy and feature size.

4.5 Relevance Vector Machine (RVM) and Classification

Once the features are well defined, the problem is formulated as classification of multiple-classes. In the previous section, the f/t signals are transformed into the feature space of the energy map and a subband from the WPT tree. If we write the whole feature vector as $\boldsymbol{x}$, then this problem is: given $(\boldsymbol{x}_i, d_i)_{i=1,2,...N}$, a set of N training data where d_i is the class label, how to determine a classifier $y(\boldsymbol{x})$ that correctly classifies the force signals? The Support Vector Machine (SVM) is established as the state-of-the-art algorithm with strong theoretical foundations for classification and regression [Vapnik 99] [Burges 98]. However, the new Relevance Vector Machines (RVM) approach is really promising, and we choose it for this work. Compared to SVM, RVM has several advantages [Tipping 01]. The most important for our application is that although it takes longer to train (training is done off-fline), it gives a much sparser model, which means simpler and faster model for the online prediction.

We show how a sparse model is achieved by RVM. For clarity, the simple case of a binary classification problem is discussed, which means $d_i \in \{0, 1\}$. Given a new input vector $\boldsymbol{x}^*$, the probability of its class label is given as:

$$p(d|\boldsymbol{x}^*) = \frac{1}{1 + exp(-y(\boldsymbol{x}^*))} \tag{4.14}$$

The RVM classifier function y is given by:

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} w_i K(\boldsymbol{x}^*, \boldsymbol{x}_i) \tag{4.15}$$

in which, $K(\cdot, \cdot)$ is a kernel function, and $\boldsymbol{x}_i (i = 1, 2, 3..., N)$ are the N training data. As discussed in [Tipping 01], the parameters w_i in 4.15 are determined by Bayesian estimation,

and a sparse prior is introduced on these parameters, forcing them to be highly concentrated around 0. Then they are computed by maximizing the posterior distribution of the class labels given the inputs. A very few nonzero terms of w_i means that a very few samples in the training data are used in the classifier function given by 4.15, achieving a sparse model. Like for SVM, the kernel trick is used to expand the basis functions for $y(\boldsymbol{x})$. In this work, we use the RBF kernel function:

$$K(\boldsymbol{x}^*, \boldsymbol{x}) = exp(-\frac{||\boldsymbol{x}^* - \boldsymbol{x}||^2}{2\sigma^2}) \tag{4.16}$$

in which, the optimized $\sigma > 0$ defines the kernel width. Readers can refer to [Bishop 06] and [Tipping 01] to see how RVM is generalized for multi-class problems. RVM has been successfully applied for fault diagnosis [Widodo 09], supervised hyper-spectral classification [Mianji 11], and for recovering 3D human body pose by regression[Agarwal 04]. The development of RVM and its comparison to SVM is given in Appendix.

4.6 Data Acquisition and Results

4.6.1 The Experimental Protocol

The experiments were carried out with several pairs of volunteers. Three qualitative velocities were chosen: slow, normal and fast and explained to each participant. The normal velocities correspond to a natural exchange. For the slow velocities the experimenters were supposed to exchange with care a flimsy and precious object. Then the experimenters were asked to realize a fast exchange.

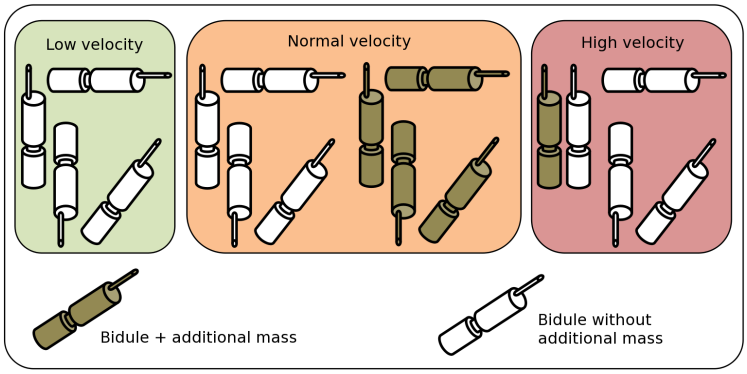


Figure 4.12: The configurations of Bidule used for the experiments

We defined four different orientations for Bidule, the two vertical ones (antenna up and upside down), one horizontal and one tilted. We realized a part of these experiments with an additional mass. Fig. 4.12 presents the seventeen different cases of exchange selected. Each couple of volunteers performed several times each case of exchange, one as giver and one as receiver. Each record lasts several seconds.

As we need to model the correct time to release the object for the giver, the samples for class "noise" are simply cut from the data, which could be pure noise, or just signal at different instant from the react time. For the third class "collision", we recorded data of class collision between Bidule and different environment: human hand, table, and others.

Figure 4.13: Two persons equipped with markers and exchanging Bidule

4.6.2 Results and Discussion

Firstly the time window is fixed to approximately 1 second, which gives 2048 data points as the acquisition rate is set to 1kHz (1024 for force data points and 1024 data points for torque). 213 experiments for each type of events are selected. Then the class noise is extracted from the same data set by two methods: the time period when nothings happens, or time-shifting on the signal of collision or grasping. The total data set for training is of size 639. Then 240 data of the three classes are also selected in the same way to test the model.

Table 4.1 shows the result of classification with different feature set: the full energy map ($\boldsymbol{E}$), the subband from level 1 $\boldsymbol{S}$, and the two combined. The choice to combine two different feature sets is justified by the results in the classification errors.

Table 4.1: Error rate for different feature set. RVM with a RBF kernel is used as the classifier

Features	$\boldsymbol{E}$	$\boldsymbol{S}$	$\{\boldsymbol{E}\ \boldsymbol{S}\}$
Error(%)	15.42	21.67	2.5

To reduce the feature size of the relative energy map $\boldsymbol{E}$ and the subband $\boldsymbol{S}$, we did two studies: firstly, we evaluated $\boldsymbol{E}$ by the Fisher LDA criterion, then we fixed $\boldsymbol{E}$ to evaluate the

different choices on $\boldsymbol{S}$ by directly comparing the classification error. Firstly, we calculated the energy map from the WPT until level 6, which produces a total of 126 features for force and 126 for torque. The relative energy of level 0 (original signal) is omitted since it is 1 for all instances. We examine the separability J of the relative energy map feature set $\boldsymbol{E}$. The results can be seen in Figure 4.14, here for the forces F. Several terms in the feature set dominate the discriminative measure, which enables us to reduce largely the size of the feature set without losing too much the discriminative information of the feature set. The final set of features is chosen from the most discriminating terms from the combination the WPT results of force and of torque.

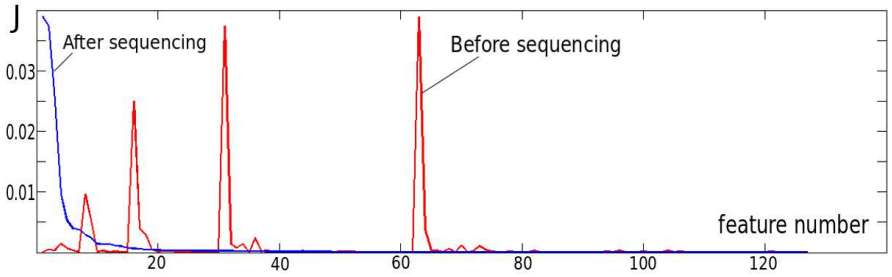


Figure 4.14: The seperability measurement of features in the relative energy map $\boldsymbol{E}$ of forces. J is calculated by (4.13) through all features in $\boldsymbol{E}$. The zeros in this measurement are largely due to the sparsity of the WPT tree. Similar results are obtained for torques too.

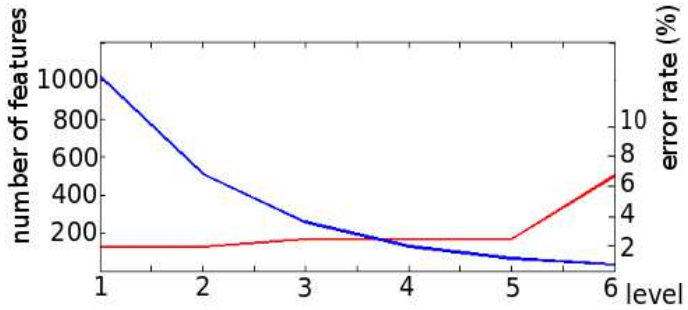


Figure 4.15: The compromise of feature size (blue curve) and classification error (red curve). The small feature size means a faster model for online prediction.

For the choice of the level from which to extract the low frequency subband, we use RVM with RBF kernel and compare directly the classification error rate. During this study the energy map is reduced to the first 13 most discriminative features. Firstly we choose $\boldsymbol{S}$ at the first level, then evaluate the classification error. Then we evaluate at level 2 and until that the error rate of classification increased significantly at level 6. It can be explained by the losing of time precision in the higher level of the WPT tree.

By this study, we finally chose the first 13 features out of 252 of the relative energy map $\boldsymbol{E}$, and the subband is fixed to level 5 with 64 features. After the dimension reduction, a classification accuracy rate of 97.5% is obtained, with a compact model having dimension of

Table 4.2: For this application, RVM produces a much sparser model than SVM, and a comparable classification accuracy.

	Dimension	Accuracy(%)
SVM	151	97.92
RVM	9	97.5

9 (the nonzero weights in RVM), while feature size is 77. When the whole WPT tree is used directly to train a RVM, the learning does not converge because of the in-class variations and due to the small size of the training data set, and the whole WPT tree decomposed until level 6 is of size 12288 (each level has 1024 elements for forces and 1024 elements for torques).

When the same feature set is used for a SVM, it gives a similar classification accuracy. As shown in Table 4.2, RVM produces a much sparser model (hence faster classification). In the table, the dimensions are the number of Support Vectors or of Relevance Vectors. Since we want to run the classifier online at a comparable speed of the controller with other complex computations, reduced complexity in model is important. Notice that only the used features but not the whole WPT tree need to be computed for online classification. The results for SVM is obtained by running the *libsvm* [Chang 11] (RBF kernel).

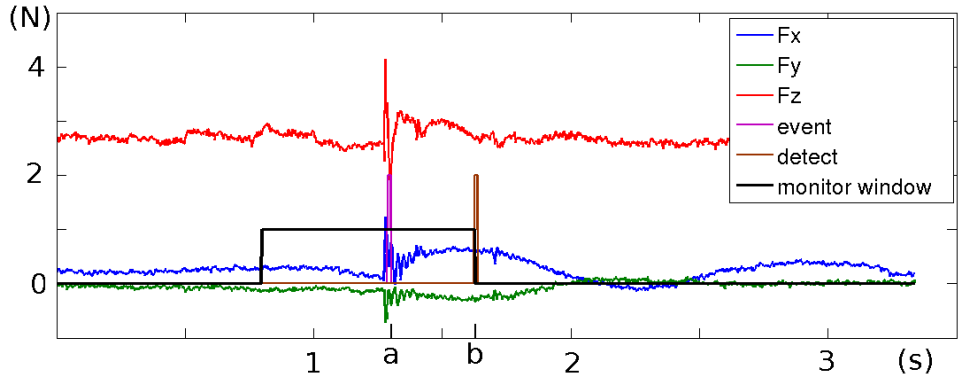
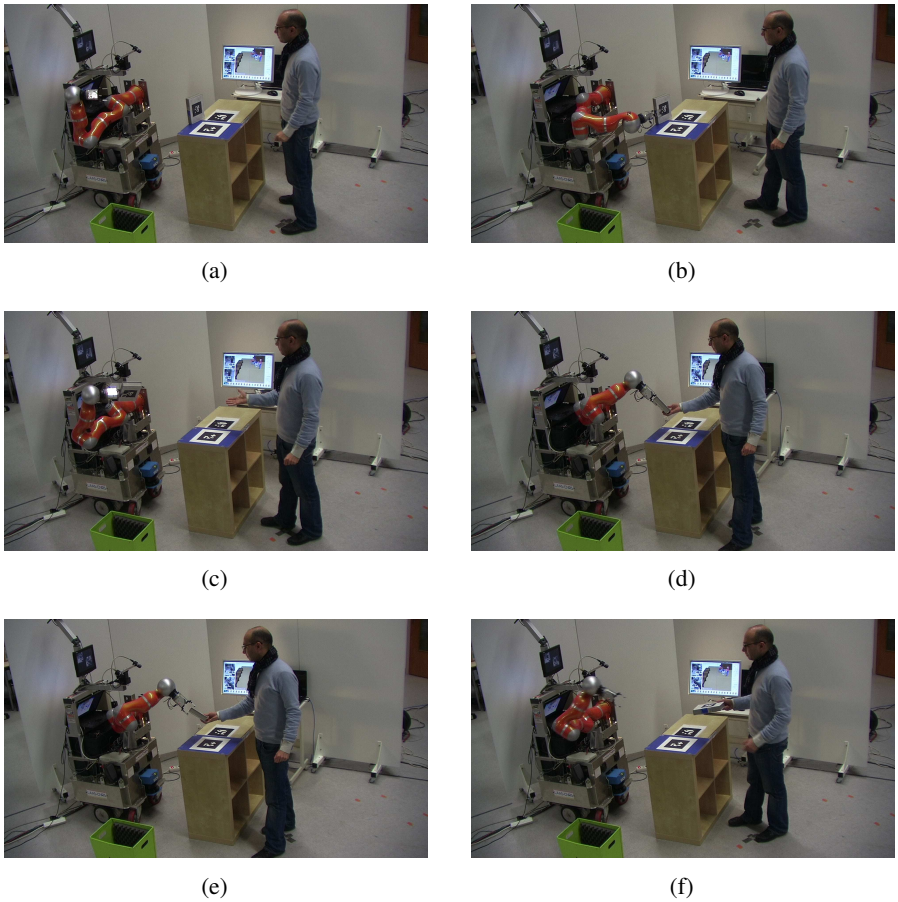


Figure 4.16: Time delay between the moment when the object is grasped (or touched here, shown by "event") and the moment when the robot reacts (shown as "detect"). The event happens at time a, and the robot reacts at time b. At time b, the system monitors the past 1 second of f/t. This time window that the system monitors is illustrated by the black line.

The classifier is implemented on the robot for the wrist f/t sensor, which is capable of using the learned model from human experience. The classifier runs at 50Hz, together with other software of perception and planning on the robot. Figure 4.16 shows how the classifier works. For this experiment, the object is held by the robot, and a person grasped the object, but not firmly (so, just touched). The figure shows the time delay between the grasp event and the time for the robot to react (here, is not to open the gripper), with detection by force signals. The time delay is normally less than half a second between the event happening and

the reaction. One important aspect is the result does not show the fastest speed the classifier can detect the event, but the right moment to react. This delay is learned by the model, which is decided by the instances acquired by human to human manipulations. Typically, the reaction to collision is faster than the reaction to grasp, and the difference is shaped by the instances to learn.

First manipulations between robot and people have been carried out with promising results. Figure 4.17 shows how the software, from signal processing to classification, are used for manipulations. During this manipulation, the robot decides to open the gripper. A user study of this approach to see how it can improve the quality of object exchange for human users in interactive manipulation is still to finish. The user study will include many aspects, such as human preference for the speed of reaction of the robot. Another example would be to compare the false negative and false positive for grasp detection and see which one has the greater influence on user's experience.



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

Figure 4.17: Software tested during manipulation. (a): Robot Jido, the human partner and objects to manipulate. (b): The robot decides to take the object on the table. (c): The person asks for the object. (d): Robot gives the object to him, and he takes the object. (e): Detecting that the object is taken by the person, the robot opens the gripper. (f): Object exchange finished with success.

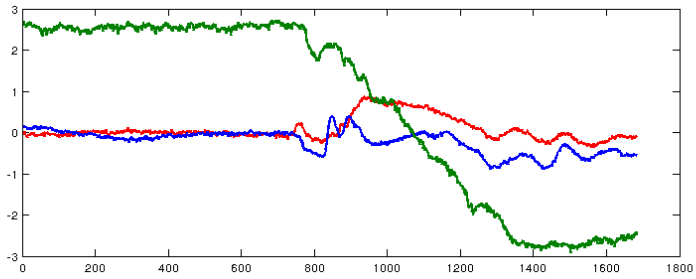


Figure 4.18: Force evolution during an exchange on Bidule. The grasping moment is visible on axis y (in blue), and the transfer of the holding of the object is evident on axis z (in green).

4.7 Conclusion

The device proposed to teach the robot how to exchange object like humans is very promising as rich information can be extracted. Synchronization of grasp and release is studied in this chapter. The Wavelet Packet Transformation is chosen, from which the energy map and a subband are used to extract the features, and Fisher Linear Discriminant Criterion is selected to reduce the dimension of feature set. Furthermore, the Relevance Vector Machine is employed as classifier because it produces a sparse model. Some results for feature selection and for classification are shown. It can be noted that by constructing a classifier, robust control policies can be achieved with no thresholds to be chosen by trial and error, which becomes especially difficult when more classes should be added.

Bidule is designed to learn more than what we discussed in this chapter, like force control policies during the exchange and the movements, as Figure 4.18 illustrates a normal exchange of the Bidule between two people. How the giver graduallys release the object (the green curve), for example, is interesting for the robot to learn. Those topics deserve further research. For the classification problem, more classes can be added to make a difference between the collisions (soft surface against hard surface), or separate more on the different grasp types. At the end, we want to mention that the collision detection by f/t sensor is also to be used for the robot to put properly an object on the table. For example, touching of an object with a table by its bottom surface of by an edge would surely produce different patterns on the force signals. And in this case, it would be reasonable to include the torques into the feature set too. With these new methods discussed in this chapter to build compact and efficient models, we can say that online classification can improve many aspects of the robotic manipulations.

5

Reactive Trajectory Controller

Rational behavior requires theory. Reactive behavior requires only reflex action.

—- W Edwards Deming

Abstract. This chapter presents the reactive controller based on sensor fusion and trajectory generation. We firstly introduce the trajectory generation methods, results from previous work at *LAAS-CNRS*. The Control Primitives are defined, based on which the main algorithms for reactive trajectory control are presented. Some results and discussion are presented at the end.

5.1 Introduction

In the context of Human Robot Interaction (HRI), intuitive and natural object exchange between human and robot is one of the basic necessary tasks for object manipulation. This chapter focuses on the trajectory controller, which enables the robot to realize a complete task of object exchange while respecting human's safety and other HRI specifications. This elementary manipulation task demands integration of different elements like geometrical and human-aware reasoning, position and external force monitoring, 3D vision and human perception information. The control system presented in this paper proposes to define a plan as a series of control primitives, each associated with a trajectory segment and a control mode. The process of trajectory generation and trajectory smoothing from path are also

introduced to help the reading of this document although it is not part of the contribution of this thesis.

5.2 From Path to Trajectory

The previous work proposed to generate a trajectory from a path using the soft motion trajectory planner designed by Broquère [Broquere 08c, Broquère 10, Broquère 11]. The path is firstly generated by a RRT path planner or its variants, presented in chapter II. This section is the results of previous work at *LAAS*, and has been reported in [Sidobre 12]. The author of this document has participated in some development and the test of the software. Research in robotics is often a cooperative work, and the content of this section, although not part of the scientific contribution of the author, is included because it is a key to understand this thesis.

5.2.1 Basic Concepts of the Trajectory Generation

5.2.1.1 Motion Condition

For the discussion of the next sections, we define a Motion Condition $M(t)$ as the position, velocity and acceleration at time t along the trajectory: $M(t) = (X(t), V(t), A(t))$. Once the trajectory is calculated, the function $M(t) = getMotion(t, \mathscr{T})$ returns the Motion Condition on trajectory $\mathscr{T}$ at time t.

5.2.1.2 Trajectory Model

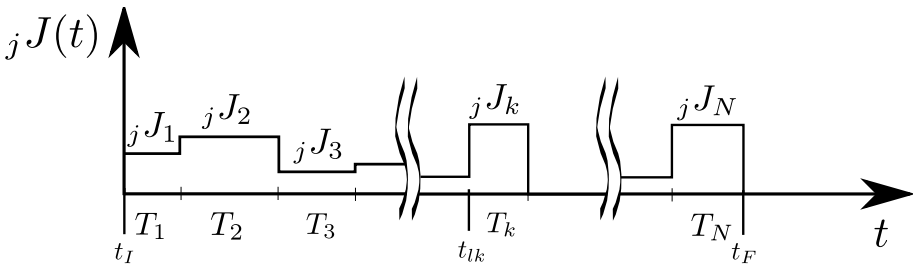


Figure 5.1: The jerk evolution for the j axis of the TR(t) trajectory.

A trajectory TR(t) is represented by a combination of n series of cubic polynomial curves. The use of polynomial cubic defined by the Soft Motion Trajectory Planner provides a solution in the context of HRI where the task introduces numerous constraints. From the trajectory generation point of view, the safety constraint is ensured by bounding the velocity and the comfort constraint by bounding the jerk and the acceleration.

The trajectory $_j$TR(t) corresponds to the evolution of the j axis and is composed of N cubic polynomial segments (curves) (Fig. 5.1). We consider that each axis has the same number of segments since they can be divided.

Functions ${}_jJ_k(t)$, ${}_jA_k(t)$, ${}_jV_k(t)$, ${}_jX_k(t)$ respectively represent the jerk, acceleration, velocity and position evolution over the segment k for the axis j. t_I is the initial time of the trajectory and t_F the final one.

A segment is defined by the Eq. (5.1) and depends on its duration T_k and on five parameters:

- the initial time t_{lk} with $t_{lk} = t_I + \sum_{i=1}^{k-1} T_i$,
- the initial conditions (3 parameters: ${}_jA_k(t_{lk})$, ${}_jV_k(t_{lk})$, ${}_jX_k(t_{lk})$),
- the jerk value ${}_jJ_k$

$\forall t \in [t_{lk}, t_{lk} + T_k]$:

$$ {}_jX_k(t) = \frac{{}_jJ_k}{6}(t - t_{lk})^3 + \frac{{}_jA_k(t_{lk})}{2}(t - t_{lk})^2 + {}_jV_k(t_{lk})(t - t_{lk}) + {}_jX_k(t_{lk}) \tag{5.1} $$

where ${}_jJ_k$, ${}_jA_k(t_{lk})$, ${}_jV_k(t_{lk})$, ${}_jX_k(t_{lk})$ and t_{lk} are constant $\in \mathbb{R}$.

The initial Motion Conditions of the trajectory ${}_j\mathrm{TR}(t)$ are ${}_jM_I = ({}_jA_I, {}_jV_I, {}_jX_I)$:

$$ \begin{aligned} {}_jX_1(t_I) &= {}_jX_I \\ {}_jV_1(t_I) &= {}_jV_I \\ {}_jA_1(t_I) &= {}_jA_I \end{aligned} \tag{5.2} $$

and the final conditions ${}_jM_F = ({}_jA_F, {}_jV_F, {}_jX_F)$:

$$ \begin{aligned} {}_jX_N(t_F) &= {}_jX_F \\ {}_jV_N(t_F) &= {}_jV_F \\ {}_jA_N(t_F) &= {}_jA_F \end{aligned} \tag{5.3} $$

where $t_F - t_I = \sum_{i=1}^{N} T_i$.

The multidimensional trajectory is then a composition of trajectories as:

$$ \mathrm{TR}(t) = [{}_1TR(t)\ {}_2TR(t)\ \ldots\ {}_nTR(t)]^T \tag{5.4} $$

where n is the number of axis.

From the N couples $({}_jJ_k, T_k)$ and the initial conditions (5.2) of the trajectory ${}_j\mathrm{TR}(t)$ we can compute the Motion Condition along the j axis at a given time with (5.5), (5.6) and (5.7). In order to simplify the notation, the j index representing the axis will be omitted.

$\forall t \in [t_{lk}, t_{lk} + T_k]$, with $t_I = 0$:

$$A_k(t) = J_k(t - \sum_{i=1}^{k-1} T_i) + \sum_{i=1}^{k-1} J_i T_i + A_I \tag{5.5}$$

$$V_k(t) = \frac{J_k}{2}\left(t - \sum_{i=1}^{k-1} T_i\right)^2 + \sum_{i=1}^{k-1} J_i T_i \left(t - \sum_{j=1}^{i} T_j\right) + \sum_{i=1}^{k-1} \frac{J_i T_i^2}{2} + A_I t + V_I \tag{5.6}$$

$$X_k(t) = \frac{J_k}{6}\left(t - \sum_{i=1}^{k-1} T_i\right)^3 + \sum_{i=1}^{k-1} \frac{J_i T_i}{2}\left(t - \sum_{j=1}^{i} T_j\right)^2 + \sum_{i=1}^{k-1} \frac{J_i T_i^2}{2}\left(t - \sum_{j=1}^{i} T_j\right) + \sum_{i=1}^{k-1} \frac{J_i T_i^3}{6} + \frac{A_I}{2} t^2 + V_I t + X_I \tag{5.7}$$

5.2.1.3 The Kinematic Constraints

The trajectory generation method is based on constraints satisfaction (velocity, acceleration and jerk). Each constraint is supposed constant along the planned motion. In the multidimensional case, each axis can have different constraints. We also suppose that the constraints are symmetrical:

$$\begin{aligned} {}_jJ_{min} &= -{}_jJ_{max} \\ {}_jA_{min} &= -{}_jA_{max} \\ {}_jV_{min} &= -{}_jV_{max}. \end{aligned} \tag{5.8}$$

Hence, the jerk, acceleration and velocity must respect:

$$\begin{aligned} |{}_jJ(t)| &\leq {}_jJ_{max} \\ |{}_jA(t)| &\leq {}_jA_{max} \\ |{}_jV(t)| &\leq {}_jV_{max}. \end{aligned} \tag{5.9}$$

5.2.1.4 The Canonical Case: the Kinematically Constrained Point-to-Point Motion

In the basic case a motion between two points where initial and final kinematic conditions are null, the Figure 5.2 represents the optimal point-to-point motion (according to the imposed kinematic constraints). This point-to-point motion is composed of seven segments of cubic polynomial functions at most [Broquère 08a].

In the multidimensional case each axis has also seven cubic polynomial segments at most. Computation details can be found in [Broquère 11].

5.2.1.5 The Minimum-Time Motion Between Two Non-null Kinematic Conditions

From the canonical point-to-point case we extend the monodimensional algorithm to compute minimum-time motion between two non-null kinematic states (non-null acceleration

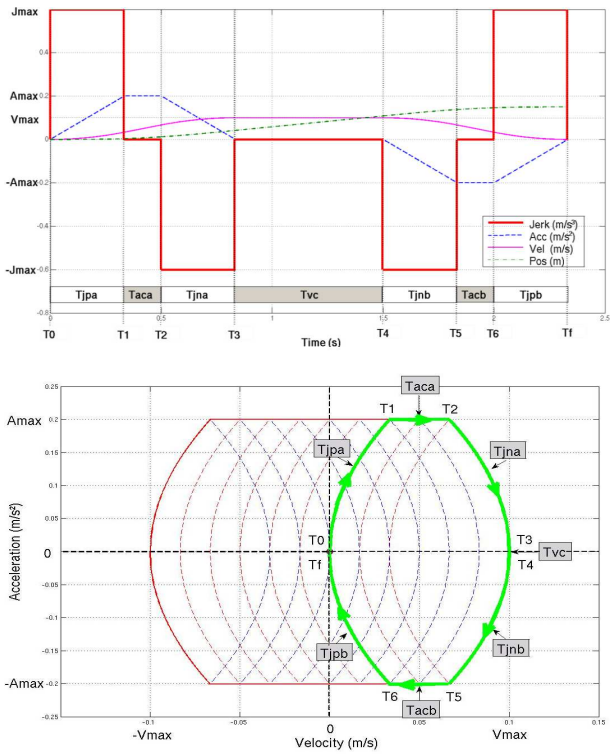


Figure 5.2: Jerk, acceleration, speed and position curves and motion in the acceleration-velocity frame for a single axis.

and velocity). An overview of this algorithm is presented in [Broquère 08a] and the details in [Broquère 11]. This kind of motion is composed of a set of elementary motions saturated in jerk, acceleration or velocity. The number of elementary motions is also seven at most. For the multidimensional case, [Broquère 11] proposes a solution to synchronize the axis motions.

5.2.1.6 The Time Imposed Motion Between Two Non-null Kinematic Conditions: the *3-Segment Method*

The method for computing a motion with an imposed duration was previously presented in [Broquère 10]. This method does not bound the jerk, acceleration nor velocity. It uses three cubic polynomial curves to define such a motion. This simple definition provides a solution to compute analytically the motion.

5.2.1.7 Smoothing an Input Function

We use the method proposed in [Broquère 08a] to compute online a smooth movement from an input defined by acceleration and velocity. At each update of the set function, a motion is computed from the current state of the system. This move is bounded by the kinematic con-

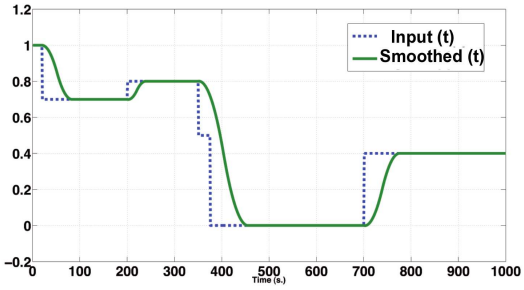


Figure 5.3: Example of the smoothing of a set function.

straints (J_{max}, A_{max} and V_{max}). Under these kinematic constraints, the minimum-time motion is defined by the critical movement associated to the critical length dc [Broquère 08a].

Thus, in order to allow a mono-dimensional system to reach its set value in minimum-time, the critical movement is computed at each iteration. An example of a smoothed signal is plotted in the Fig. 5.3. The blue dotted curve is the input and the green curve is the smoothed velocity. The method acts like a filter for the acceleration.

5.2.2 Trajectory Generation From a Given Path

The trajectory generation is based on the three main methods introduced in the previous section. The input is the path $\mathscr{P}$ computed by the path planner.

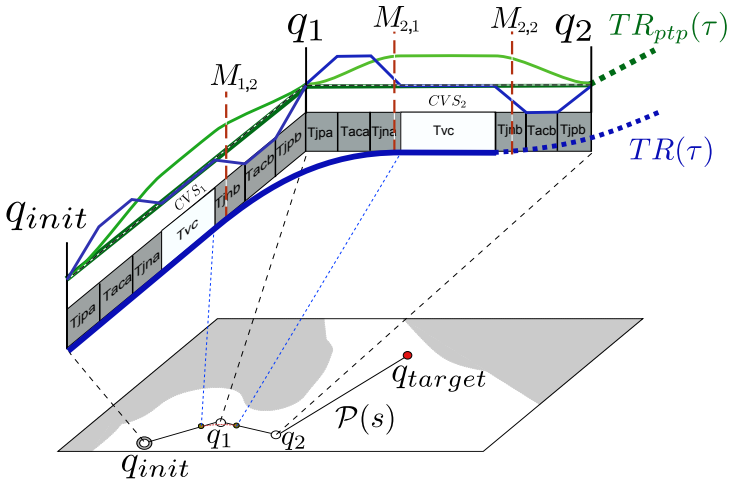


Figure 5.4: From the path $\mathscr{P}$ to the smoothed trajectory TR.

The first step is to calculate a trajectory passing through all the nodes of the path $\mathscr{P}$. This trajectory, which we call TR_{ptp} consists of point-to-point movement (Sect. 5.2.1.4) and therefore includes stop motions at each configuration defining a node.

The second step consists in smoothing these stop motions to obtain a shorter trajectory in time TR. Smoothing uses the same 3D model than the research phase of the path. Thus, collisions are tested during the computation of the transition moves at each node. If a

collision appears during the smoothing of the stop move at node q_i, then the movement will not be smoothed for this node and the stopping move will be kept.

In the following, we detail a method for smoothing stop motions based on the computation of a fixed time movement using the *3-segment* method presented in previous work.

5.2.2.1 Smoothing of the Stop Motions

We propose a method based on the minimum time algorithm for trajectory generation (Sect. 5.2.1.5) [Broquère 08a] and on the *3-segment* method (Section 5.2.1.6) to smooth the stopping motions [Broquère 10].

The trajectory TR_{ptp} (Fig. 5.4) between the first two nodes q_{init} and q_1 is a point-to-point motion in a straight line of duration $T_{(q_{init}q_1)}$. Similarly the motion between q_1 and q_2 is a point-to-point motion of duration $T_{(q_1q_2)}$. The stop motion is smoothed between the points $M_{1,2}$ et $M_{2,1}$.

Notation: We note the points that limit the smoothing $M_{i,j}$, the index i is the index of the point-to-point motion (the first of the trajectory has an index of 1). The index $j \in \{1,2\}$ is 1 if this point is the final extremity of the transition motion with the previous point-to-point motion and conversely for $j = 2$.

Choice of the Points $M_{i,j}$

Let us consider the transition motion in the neighborhood of q_1 located at time t_{q1}:

$$t_{q1} = t_I + T_{(q_{init}q_1)} \tag{5.10}$$

To simplify, we choose $t_I = 0$ as the time origin of the trajectory.

The time positions $t_{M_{1,2}}$ and $t_{M_{2,1}}$ of the points $M_{1,2}$ and $M_{2,1}$ are determined from a given parameter τ ($\tau \geq 0$)such that:

$$M_{1,2} = TR_{ptp}(t_{q1} - max(\tau, \frac{T_{(q_{init}q_1)}}{2})) \tag{5.11}$$

$$M_{2,1} = TR_{ptp}(t_{q1} + max(\tau, \frac{T_{(q_1q_2)}}{2})). \tag{5.12}$$

So when τ is null, the movement stops at the point q_1. When τ satisfies (5.13), the transition motion connects the midpoints of the line segments (q_{init}, q_1) and (q_1, q_2) because of the symmetry of the velocity profile about this point.

$$\tau \geq max\left(\frac{T_{(q_{init}q_1)}}{2}, \frac{T_{(q_1q_2)}}{2}\right) \tag{5.13}$$

In practice, unless otherwise specified, by default we choose the points $M_{i,j}$ such that the transition movement begins at the end of the constant velocity segment of the first point-

to-point movement (P_1, P_2); the transition movement ends at the beginning of the constant velocity segment of the second point-to-point movement (P_2, P_3).

Notice that, for a given value of the parameter τ, the Euclidean distance between the points $M_{i,j}$ and the corresponding point q_i varies according to kinematic parameters of the point-to-point movement.

5.2.2.2 Computation of the Transition Movement

Let us consider a trajectory of dimension n. The instants $t_{M_{i-1,2}}$ and $t_{M_{i,1}}$, start and end of the transition movement at the configuration q_i, are identical for all n dimensions. The computation method is described by Algorithm 1. The first step consists in computing, for each axis, the optimal time motion to determine the duration T_{imp} of the transition movement. The method *3-segment* to compute the movement in fixed time is then applied to each axis.

Algorithm 1: Computation of a transition movement near of a node q_i

begin
  Determine the switching points $M_{i-1,2}$ and $M_{i,1}$ (eq. 5.12 and 5.11)
  for *each dimension* n_i **do**
    Compute the one-dimensional movement in minimum time (Section 5.2.1.5)
    Compute the duration of the one-dimensional movement in minimum time $T_{opt}[i]$
  end
  Determine the duration of the transition movement
  $T_{imp} = max(\forall\, i \in [1,n] \mid T_{opt}[i])$
  Compute the Motion Condition at switching points, at time $t_{i-1,1}$ and $t_{i,1}$
  for *each dimension* n_i **do**
    Compute triplets of cubic curve segments from the method *3-segments* (Section 5.2.1.6)
  end
end

Figure 5.5 illustrates an application of the method for the case of a movement defined by three points P_1, P_2, P_3 and by the kinematic constraints $V_{max} = 0.1m/s$, $A_{max} = 0.3m/s$ et $J_{max} = 0.9m/s$. The transition movements are computed for different values of the parameter τ.

The proposed method ensures the continuity in velocity and acceleration for each dimension. The initial and final velocities of the transition movements can be different and acceleration not zero. The duration of the transition movement is computed by taking into account the kinematic constraints of each dimension using the minimum time algorithm (Sect. 5.2.1.5). Therefore this method guarantees that changes in velocity, acceleration and jerk are limited. However, in some cases, constraints can be exceeded by the *3-segment*

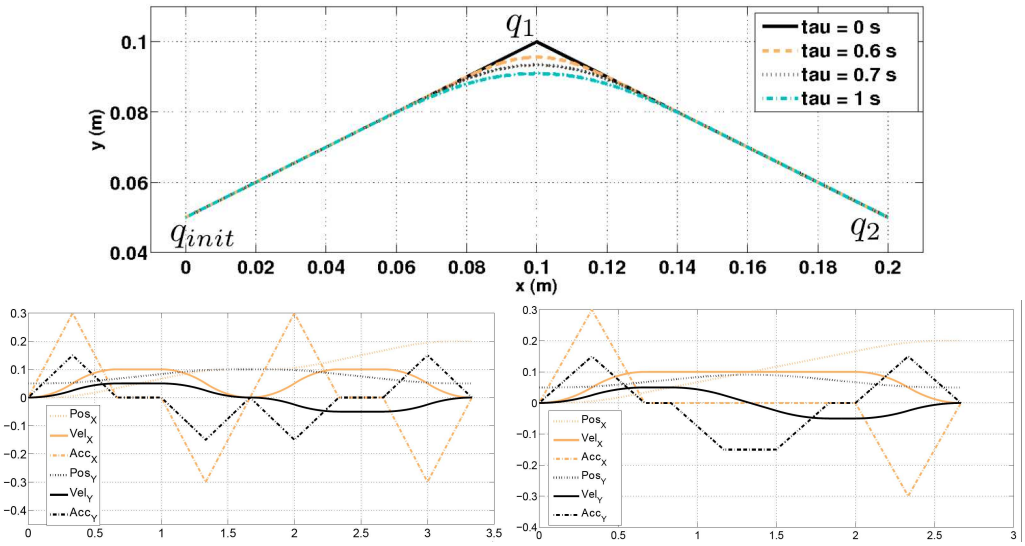


Figure 5.5: Transition movement for two lines that form an angle of about 127 ° (top), graph of position, velocity and acceleration as function of time for a point-to-point motion for $\tau = 0$ (bottom-left) and $\tau = 1s$ (bottom-right)

method. In practice, we introduce a percentage (10%) of exceeding for each constraint. If, for a transition movement, the exceeding of kinematic constraints is too large, this movement is not smoothed to comply with the constraints of human comfort.

5.2.3 Application to Robot Manipulators

To better explain the method, we apply it to an example of task of grasping an object, the grey tape cassette of the Fig. 5.6. The path of the center of the end effector of the robot (hand) is described by the green line segments in Fig. 5.7. On this path, the spheres represent the initial, final and intermediate configurations (nodes). The path of the point-to-point trajectory TR_{ptp} is identical to the path planned. This trajectory stops at each intermediate node. The smoothed path TR is represented by the black curve. We note that the trajectory, before smoothing, stops at the first node as a smoothing in its neighborhood would have introduced to collision[1] between the hand of the robot and the environment. The planning of the path of the trajectory was performed in the Cartesian space of the robot by considering the platform was fixed. The following section presents the methodology to take into account the redundancy of the robot.

[1]Note: Another solution would be to compute a path that goes further from the obstacle but it is not the purpose here.

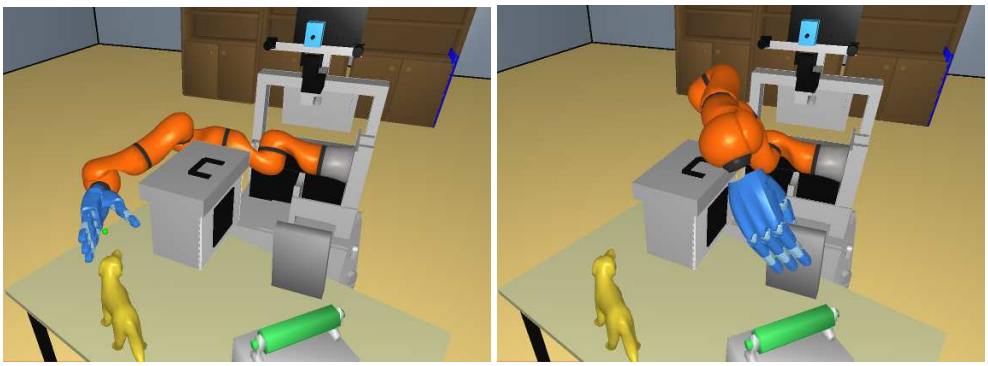


Figure 5.6: Initial configuration and grasp configuration of the robot Jido.

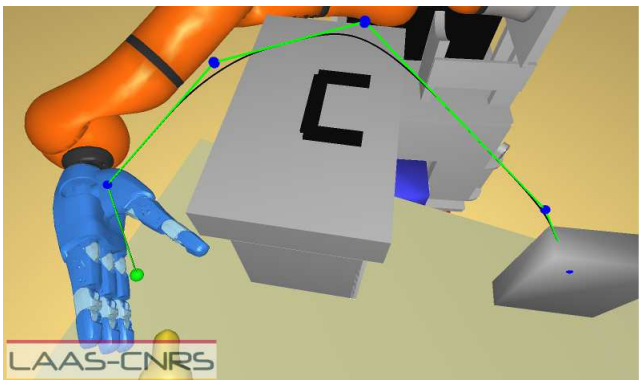


Figure 5.7: Trajectories TR_{ptp} et TR in the Cartesian space to grasp the cassette

5.2.4 Planning in the Cartesian Space

5.2.4.1 Generation of the smoothed trajectory TR in Cartesian space

To represent the complete configuration of the robot in Cartesian space, we propose to use a vector X_i with:

- the position of the robot base,
- the pose of the end effector(s),
- the configuration of the redundant axis of the arms if they have more than six degrees of freedom (DOFs),
- the configuration of the hand(s),
- the configuration of the head.

In the following, we consider that the platform is fixed. For a system operating in 3D space, six independent parameters are used to define the position of the end effector. For the planning, the system is decomposed into *passives* and *actives* parts corresponding respectively to dependent and independent variables [Cortés 04, Han 01]. Thus a robot manipulator with six DOFs, is decomposed as follows: the independent variables (active) are the six DOFs (position and orientation) of the end effector and the joint variables are the dependent variables (passive) .

In the case of our Jido[2] robot, as the robot arm is composed of seven DOFs, it is therefore redundant. In addition to the pose of the end effector, a joint of the arm is chosen and becomes an *active* variable. Notice that, if the motion of a holonomic platform was considered, then these DOFs would be *active* variables.

During the planning of the path in the Cartesian space, only the *active* variables are sampled using, according to the circumstances, the RRT or the T-RRT algorithm. The *passive* variables are computed in a second step by solving the inverse kinematics of the arm prior to test the validity of the sampled configuration of the robot (bounds and collision). During the test of the validity of a local path between two configurations, the inverse kinematic function is also called.

To perform the interpolation between two configurations, we represent the position of the end effector by a displacement: three parameters for the position and three parameters for the orientation (vector and angle representation with the norm of the vector equal to the angle [Broquère 08a]). We have implemented a local method of interpolation between two configurations. This method takes as parameters two local configurations (with their kinematic conditions) and the imposed kinematic constraints (J_{max}, A_{max} et V_{max}) for each active axis. After applying the local method between each intermediate configuration, the

[2] Jido is an MP-L655 platform from Neobotix, equipped with a KUKA LWR arm.

obtained trajectory TR_{ptp} is composed of point-to-point movement of dimension n (n is the number of active axes), that is for Jido $n = 22$ parameters (6 for the end effector, 1 for the axis of the redundant manipulator, 13 for hands and 2 for the head).

The smoothed trajectory TR in Cartesian space is then obtained by the method described in the previous section applied to the active axes (Sect. 5.2.2).

5.2.4.2 Conversion of the Trajectory in the Joint Space of the Robot

As most of the robot controllers operate in the joint space, it is important to provide a solution to convert Cartesian trajectories into joint ones. To perform this transformation, the trajectories of passive axes are obtained by discretizing the trajectory TR defined in Cartesian space and performing inverse kinematics for each sample. The trajectory TR is discretized at the period of operation of the robot controller. This allows obtaining the position, and by derivation, the velocity and the acceleration of all the DOFs of the robot.

However, this discretization removes the notion of time and requires a large amount of data to represent the trajectory.

We can use the approximation method of trajectory presented in [Broquère 10] and [Broquère 11] to approximate this discretized trajectory and thus obtain a compact description of the trajectory. Unlike the approximation in the Cartesian space, the trajectory error taken into account by the approximation algorithm is the maximum error of the trajectory of each DOF.

The obtained approximated trajectory TR_{app} is a function of time, it is composed of series of segments of cubic curves for each joint variable of the robot.

However, movements of the passive axes are not planned and many situations can lead to exceed the kinematic limits of the robot. In this case, the trajectory cannot be directly performed. To adapt the trajectory when the task allows it, we replace the time parameter t of the trajectory by applying a function α, $\mathbb{R} \longrightarrow \mathbb{R}$. The function α will make it possible to change the time increment during the execution of the trajectory and therefore allow slowing down the execution.

The period of the trajectory controller is denoted ΔT. In the case of a classical execution, the application α is defined by:

$$\alpha(t) = t \tag{5.14}$$

and, in discrete notation:

$$\alpha(k\Delta T) = \alpha((k-1)\Delta T) + \Delta T \tag{5.15}$$

The trajectory carried out is $TR_{app}(\alpha(t))$.

The introduction of the function α makes it possible to modify the motion law of the trajectory TR_{app} and thus to adapt the evolution of each joint of the robot in a synchronized way.

To determine the function α in the case where one wishes to adapt the motion law, we

first determine for each instant of the trajectory TR_{app} exceeding β the velocity of each axis relatively to the corresponding maximum velocity (maximum values used here are the default limits accepted by the system). We obtain:

$\forall k\Delta T \in [t_I, t_F]$,

$$\beta(k\Delta T) = \begin{cases} 1 & if\ \forall j \in [1,n],\ {}_jV(k\Delta T) \leq_j V_{max}^{mot} \\ min\left(\forall j \in [1,n] \mid \frac{{}_jV_{max}^{mot}}{{}_jV(k\Delta T)}\right) & \text{else} \end{cases} \tag{5.16}$$

where n is the number of controlled DOFs, ${}_jV(t)$ the evolution of the velocity of the joint j and ${}_jV_{max}^{mot}$, the maximum velocity of the joint j.

Thus we obtain:

$$\alpha(k\Delta T) = \alpha((k-1)\Delta T) + \beta(k\Delta T)\Delta T \tag{5.17}$$

with $\alpha(0) = 0$.

However, the trajectory $TR_{app}(\alpha(t))$ cannot be executed directly because it would introduce discontinuities in velocity due to the discontinuity of β. To smooth the evolution of β, we apply a variant of the method described in Sect. 5.2.1.7 that anticipates the change in β. The smoothed function β is denoted by β_{smooth}. The details of the smoothing algorithm is omitted here and readers may refer to [(Ed.) 12] for further reading.

The method presented above allows modifying the velocity of each joint of the robot to satisfy the velocity bounds. We have supposed that the resulting path respects the constraints of acceleration. Otherwise, it is possible to identify a function β^{acc} equivalent to β to take into account overtaking accelerations. In practice, for HRI, the kinematic constraints of the trajectory are small in comparison to the capabilities of the system and it is not necessary to check for overtaking of acceleration.

5.3 Control Primitives

In HRI, the robot does various tasks like picking up an object, giving an object to human, taking an object from the human. For each task, a path is planned to realize it, and then the path is transformed into a trajectory. The controller designed here takes directly the trajectory as input and segments it based on some cost maps.

Figure 5.8 shows the basic frames needed to define a task. The trajectory $\mathscr{T}_m$ defines the move that allows the robot to do the task of grasping an object handed by the human.

Based on the cost values associated to each point of the trajectory, the trajectory is divided into segments associated to a control strategy. The 3D cost maps used are of different types: collision risk map calculated based on the minimum distance between the trajectory and the obstacles; visibility and reachability map of a human [Sisbot 11] and safety and comfort 3D map of a human. Chapter II presented two examples of cost maps. For example, when the risk of collision with the robot base is high, the trajectory can be controlled

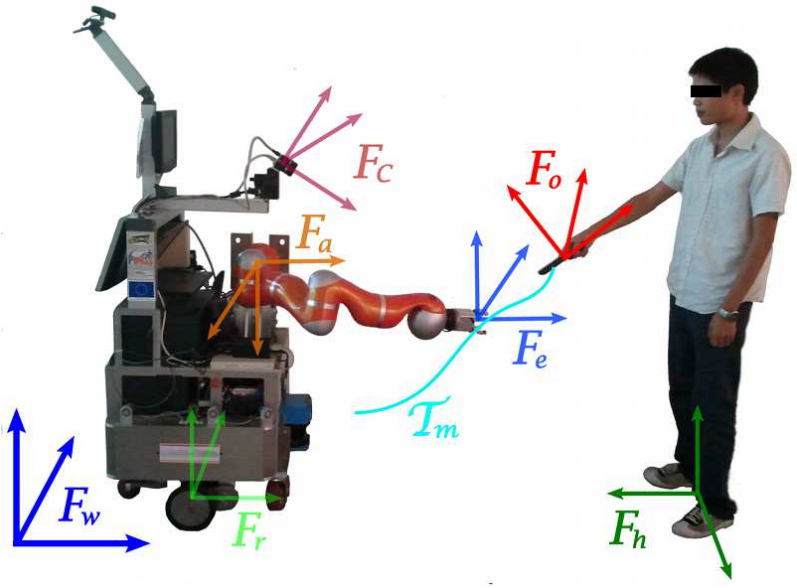


Figure 5.8: Frames for object exchange manipulation: F_w: world frame; F_r: robot frame; F_c: camera frame; F_e: end effector frame; F_o: object frame; F_h: human frame. The trajectory realizing a manipulation should be controlled in different task frames.

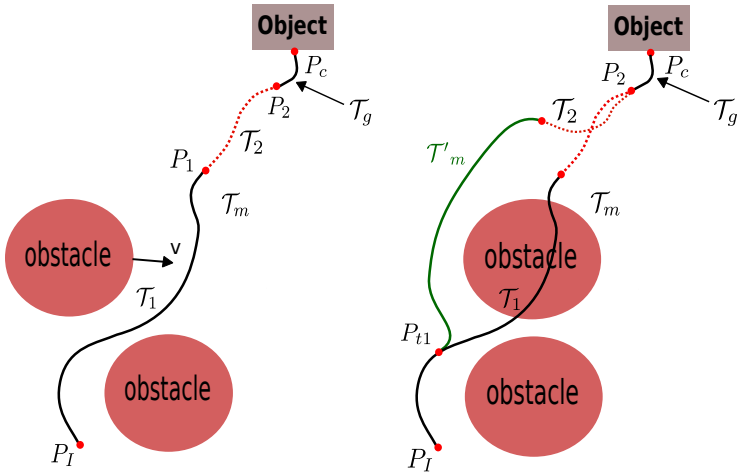


Figure 5.9: Left: trajectories of the control primitives. Right: trajectory switching for the controller due to the movement of an obstacle.

in the robot frame. Similarly, in the case where the human is handing an object to the robot, the grasping must be controlled in the object frame. [Sidobre 12] details other aspects of the use of cost maps to plan manipulation tasks.

To simplify the presentation, in the reminder of the document we focus on the manipulation tasks where a human hands over an object to the robot. During the manipulations, the human moves and the different frames defining the task move accordingly. Based on the change of cost values, we divide the trajectory $\mathcal{T}_m$ in Figure 5.8 into three segments, as illustrated in the configuration space in the left part of Figure 5.9. In the figure, the points connecting the trajectory segments are depicted by red dots. The first segment $\mathcal{T}_1$, which is

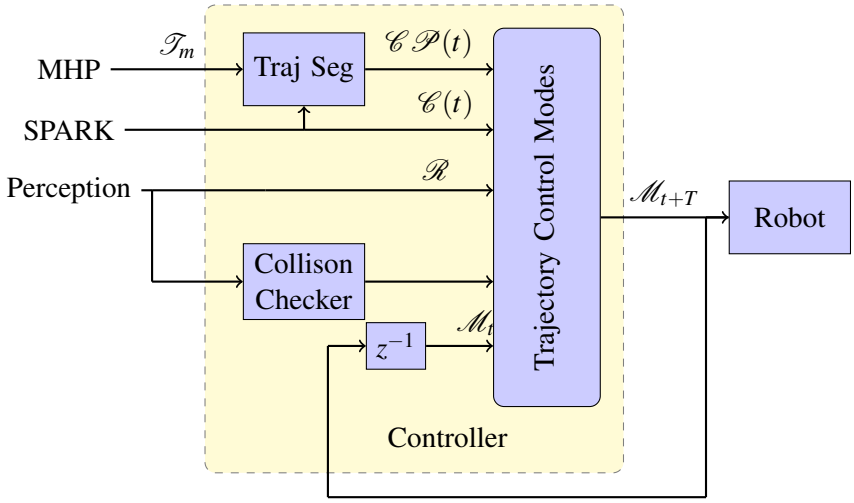


Figure 5.10: Concept of input and output of the controller. $\mathscr{T}_m$ is the trajectory computed by the human aware planner MHP, it is then segmented into control primitives ($\mathscr{C}\mathscr{P}(t)$). Traj Seg represents *trajectory segmentation*. $\mathscr{C}(t)$ are the cost values. $\mathscr{R}$ represents the transformation matrices, giving the position of the target and of the robot. $\mathscr{M}_t$ is the current state of the robot, $\mathscr{M}_{t+T}$ is the desired motion condition for the next control cycle. z^{-1} represents delay of a control cycle.

defined in the robot frame, has a high risk of auto-collision. When human or object moves, the cost value of collision risk stays the same. Segment $\mathscr{T}_2$ has a lower collision cost value, so modifying the trajectory inside this zone does not introduce high collision risk. The end part, segment for grasping movement $\mathscr{T}_g$, has a high collision cost value. To ensure the grasping succeeds without collision this segment of trajectory should be controlled in the moving object frame.

We name *task frame* the frame in which the trajectory must be controlled. We define a *control primitive* $\mathscr{C}\mathscr{P}$ by the combination of five elements: a segment of trajectory, a cost function, a task frame, a control mode, and a stop condition.

$$\mathscr{C}\mathscr{P}(t) = (\mathscr{T}_{seg}(t),\ \mathscr{C}(t),\ \mathscr{F},\ \mathscr{O},\ \mathscr{S})^T \tag{5.18}$$

In which, $\mathscr{T}_{seg}(t)$ is the trajectory segment, $\mathscr{C}(t)$ is the cost value, provided by SPARK and associated to the trajectory which is monitored during the execution of a control primitive, $\mathscr{F}$ is the task frame, $\mathscr{O}$ is the control mode which we will define in next section, and $\mathscr{S}$ is the stop condition of the control primitive. For example, the grasping movement includes five elements: the trajectory segment $\mathscr{T}_g$, the high collision risk cost value $\mathscr{C}(t)$, the task frame $\mathscr{F}_o$, the control mode as trajectory tracking, and the stop condition $\mathscr{S}$ as a predefined threshold for the distance between the robot end effector and the end point of $\mathscr{T}_g$. In the literature, Manipulation Primitives or Skill Primitives are often the concept for the interme-

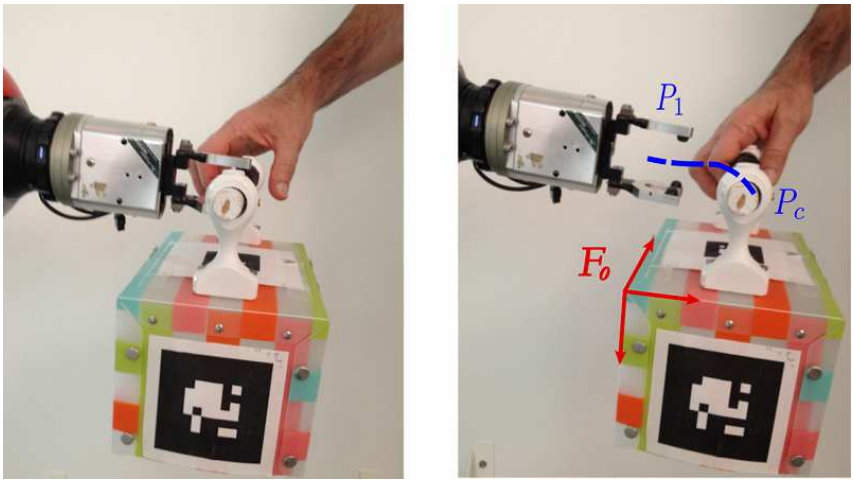


Figure 5.11: A simple case of grasp. Left: a planned grasp defines contact points between the end effector and the object. Right: To finish the grasping, the manipulator must follow the blue trajectory P_1 - P_c, and then close the gripper. This movement must be controlled in the object frame F_o.

diate level between planning and control and have been discussed in numerous works, as in [Kröger 11].

Using the definition of control primitives ($\mathscr{C}\mathscr{P}(t)$) and Motion Condition: $M(t) = (X(t), V(t), A(t))$, or written as M_t, the different components of the trajectory controller and the input and output are presented in Figure 5.10. The initial trajectory $\mathscr{T}_m$ is segmented into a series of $\mathscr{C}\mathscr{P}(t)$. The cost values $\mathscr{C}(t)$ are used during the segmentation, they are also monitored by the controller during execution of a control primitive. The collision checker integrates data from vision, human perception and encoder of the robot. It prevents collision risk by slowing down or suspending the task execution. With all the data and the current Motion Condition M_t of the robot, different control modes can compute Motion Condition for the next control cycle, which are the input for the robot servo system.

Figure 5.11 shows the last control primitive of *grasping an object*. It is similar to the end part, $\mathscr{T}_g$, of the trajectory in Figure 5.8. The grasp position, the contact points and the final trajectory are planned by the grasp planner. More details on the grasp planner are given in [Bounab 08] and [Saut 12]. When the object moves, the object frame F_o and the path of the trajectory moves also. So to avoid collision, the trajectory of these control primitives must be controlled in the object frame F_o.

5.4 Reactive Trajectory Controller

At the control level, a task is defined by a series of control primitives, each defined by a quintuplet. The first level of the proposed trajectory controller is a state machine which monitors the execution, controls the succession of the control modes, and manages the

collision risk and other special situations. Target tracking and trajectory tracking are parts of the control modes presented after the state machine.

5.4.1 Execution Monitoring

A state machine controls the switching between the different control modes associated to each control primitive and monitors the execution. Due to human presence, the robot environment is moving and the control task must be adapted accordingly. The state machine can also suspend or stop the control of a control primitive like depicted in Figure 5.12.

Suspend Events: When the visual system fails or the target becomes unavailable, or because of some specific human activities based on the monitoring of cost value $\mathscr{C}(t)$, the trajectory controller should suspend the task.

Stop Events: Whatever the control mode chosen, unpredictable collisions can occur and they must stop the robot. Our controller uses two modules to detect these situations.

The first is a geometric collision checker based on results from Larsen et al.[Larsen 99]. It updates a 3D model of the workspace of the robot, and runs at the same frequency as the trajectory controller. This checker is geometric based, and can stop the robot when the collision between the robot and the environment is predicted.

The second one is based on [De Luca 08] and monitors the external torques. The method was designed to detect unexpected physical collision between the robot and the obstacles. The fast detection of collision is realized using the momentum-based method reported in the paper, which does not require any external sensing. This monitor provides a security guarantee for Human Robot Interaction context. With the implementation of the torque monitor on the robot, the robot automatically stops when collision occurs.

Slow Down On Trajectory: Based on the input cost function, the controller can slow down on the main trajectory by changing the time function $s(t)$. Imagine that a fast movement could cause some people anxiety when the robot is close to them, for example. We propose to use the geometric models of the robot and of the human, updated at each iteration during the execution to ensure the safety and comfort of humans. We choose to take into account the weighted average *cost* of the security and visibility constraints introduced in chapter II. The method to adapt the motion law is the same as the one presented in the previous section. The costs are high when the distance human-robot is short or when the robot is outside the field of view of the human, the cost taken into account is $cost_{inv} \in [0,1]$ such that:

$$cost_{inv}(k\Delta T) = 1 - cost(k\Delta T) \tag{5.19}$$

The cost $cost_{inv}$ is then smoothed on-line, using methods presented in section 5.2.1.7.

Each elementary controller based on online trajectory controller is implemented with a simple state machine inside.

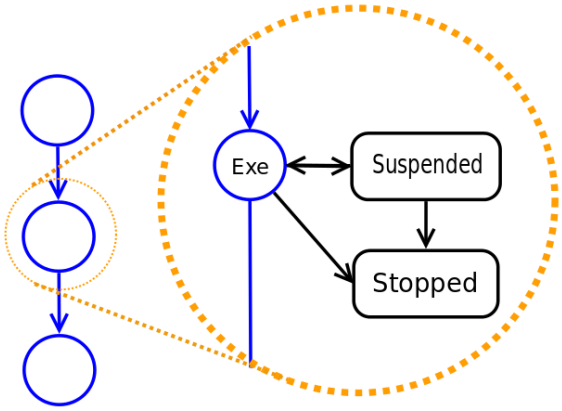


Figure 5.12: In the left, each circle represents the controller of a control primitive. The system can suspend or stop the execution of a control primitive.

5.4.2 Trajectory Control Modes

Depending on the context defined by the control primitives, different control strategies must be developed. Online trajectory generator gives a flexible solution to build these controllers, which can easily react to unforeseen sensor events and adapt the kinematic parameters, mainly velocity, to the environment context. Switching to a new trajectory or a new frame in which the trajectory is controlled is also possible.

The main idea of the controller is to compute and follow a trajectory while joining up the target trajectory or a target point from the current state. Several control modes are defined to solve the reactive HRI manipulation problem.

Control Mode 1: Target tracking. If we suppose the robot is in an area without risk of collision, the system can track the end point of the trajectory. In this case, the controller generates iteratively a trajectory to reach the end point and send the first step of this trajectory to a low-level controller. In the special case where the controller does target tracking with visual system, it does visual servoing.

Figure 5.13 shows the details of the trajectory control mode for *Target Tracking*. The object is at position O at current time t, and moves following the curve $\mathscr{T}_{obj}$.

This curve is obtained by a simple Kalman filter, building a movement model from the results of 3D vision system. $\mathscr{F}_r$ is the robot base frame, $\mathscr{F}_c$ and $\mathscr{F}_o$ are camera frame and object frame, respectively. Also, R_r^c is the 4×4 transformation matrix from $\mathscr{F}_r$ to $\mathscr{F}_c$ and R_c^o the transformation matrix from $\mathscr{F}_c$ to $\mathscr{F}_o$. They are all in dashed line and they change with time when the human or the object moves. The camera direction is adjusted to center the object in the image. Initially, the robot is at point P_e, since there is no risk of collision, the controller can simply track point P_2, which is the end point of the segment. It is also

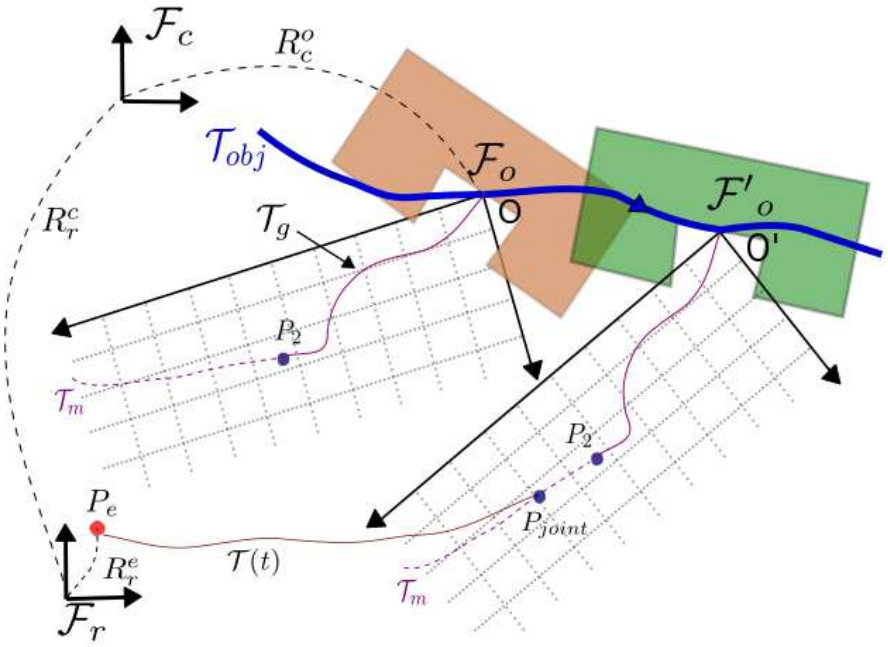


Figure 5.13: *Control Mode 1.* The robot tracks the point P_2. The object moves to the right, it is drawn at two times: firstly in brown for time t_1 and then in green at time t_2. In both cases, the entry point P_2 of the trajectory $\mathcal{T}_g$ is drawn relatively to the object frame $\mathcal{F}_o$.

possible for the robot to join up the trajectory at another point P_{joint} defined in the object frame which is the task frame. The details of the algorithm is given in Algorithm 2 where:

T**:** duration of one control cycle.
$\mathbf{M_r}$**:** current motion condition of the robot, so $M_r = (X_r, V_r, A_r)$.
δ**:** distance threshold to stop the tracking process.
$\mathbf{M_g(t)}$**:** motion conditions at time t on trajectory $\mathcal{T}_g$.
$\mathbf{M_{P_2}}$**:** motion conditions of the target P_2 on the main trajectory, which is calculated by the planner.
$\mathbf{T_{Max}}$**:** the maximum time the controller to track the target or the trajectory. Once the time exceeds the value, the trajectory controller is suspended and a signal is sent to the supervisor, requiring the replanning of new task or a new path to realize the task.
$\mathbf{X}, \mathbf{Q}$**:** input signal in Cartesian space and in joint space for low-level controller.

Control Mode 2: Trajectory tracking in task frame. Once robot reaches point P_2, it starts the grasping movement, which corresponds to trajectory $\mathcal{T}_g$ in Figure 5.9 and Figure 5.15. The object is still moving, but as the robot is in the high cost zone, it should track the main trajectory in the task frame. The details of the control mode is given in Algorithm 3.

Figure 5.15 shows the details of the control mode, all the frames and object movements are the same as in Figure 5.13, but the robot is at point P'_e. The robot tracks $\mathcal{T}_g$ in the object frame $\mathcal{F}_o$, and will end up executing $\mathcal{T}'(t)$ in the robot frame. This control mode

can be applied in numerous situations. For example, when PR2 needs to grasp an object on a moving conveyor belt, the grasping movement is achieved by tracking a trajectory in the object frame, as shown by Figure 5.14.

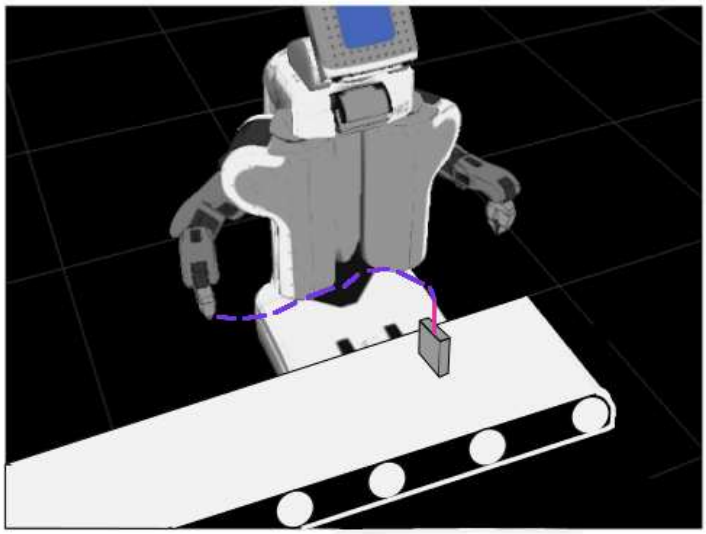


Figure 5.14: One application for *Control Mode 2*. The robot needs to grasp an object on a moving conveyor belt. The object is moving w.r.t. the robot and the grasping movement is planned as the red part of the trajectory. This part of the trajectory should be tracked in the object frame.

Algorithm 2: Control for target tracking (*Control Mode 1*)

input : Target point P_2;
while $(distance(P_2, M_r) > \delta) \wedge (t < T_{Max})$ **do**
  system time $t = t + T$, $Loop = Loop + 1$;
  Update perception data;
  if *Collision Detected* **then** Emergency stop;
  if *Suspend Events Occur* **then** Suspend task;
  Coordinates transformations;
  Generate Type V control trajectory $\mathscr{T}(t)$, for which: $IC = M_r$, $FC = M_{P_2}$;
  $X = getMotion(t + T, \mathscr{T}(t))$;
  Inverse kinematics: $X \rightarrow Q$;
  Q to the position servo system;
end

Control Mode 3: Path re-planning and trajectory switching: during the execution, a path can be re-planned, for example when an obstacle moves (see Fig. 5.9). A new trajectory is computed by the planner MHP and given to the controller that switches to the new trajectory. While the controller is following the trajectory $\mathscr{T}_m$, an obstacle moves and invalidates the initial trajectory. In this case, the controller can suspend the execution, and decide to switch the new trajectory provided by MHP, written as $\mathscr{T}'_m$, beginning at time t_1

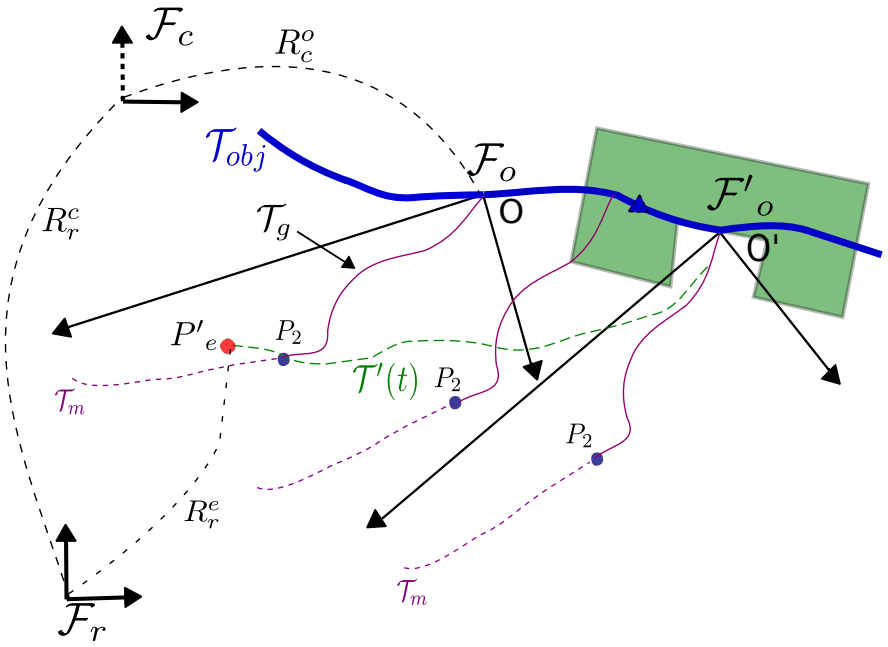


Figure 5.15: *Control Mode 2.* Object at time t_1 is colored in light brown, and green at time t_2. It follows a movement model given as the blue trajectory $\mathcal{T}_{obj}$. The purple trajectory for grasping $\mathcal{T}_g$ stays unchanged in the object frame. The robot tracks the trajectory $\mathcal{T}_g$ as it does the grasping movement.

Algorithm 3: Control for trajectory tracking in a moving work frame (*Control Mode 2*)

input : Trajectory segment $\mathcal{T}_g$;
while $(distance(P_c, M_r) > \delta) \wedge (t < T_{Max}))$ **do**
  system time $t = t + T$, $Loop = Loop + 1$;
  Update perception data and object movement model;
  if *Collision Detected* **then** Emergency stop;
  if *Suspend Events Occur* **then** Suspend task;
  Coordinates transformations;
  $M_{\mathcal{T}_g} = getMotion(t + T, \mathcal{T}_g)$;
  $M_{object} = getMotion((t + T, \mathcal{T}_{obj})$;
  $X = M_{\mathcal{T}_g} \oplus^* M_{object}$;
  Inverse kinematics: $X \rightarrow Q$;
  Q to the position servo system;
  $^*\oplus$ denotes the vector addition of two Motion Conditions.
end

in the future. The controller anticipates the switch, and when the robot reaches P_{t_1} at time t_1, the robot switches to the new trajectory $\mathcal{T}'_m$. Because the new trajectory $\mathcal{T}'_m$ is calculated using the state of the robot at time t_1 as its initial condition, the trajectory is switched without problem.

In this section, we essentially solved the problem of the task of grasping a moving object

held by the human counterpart. For other tasks, like picking an object, giving an object to human or putting an object on the table, the same functionalities can be used. For example, putting an object on a moving platform would require the end segment of the main trajectory to be controlled in the frame of the platform, which moves in the robot frame. Likewise, giving an object to a moving human hand will require the manipulator to track the exchange point, normally planned by a human-aware motion planner till the detection that human grasps the object successfully. Although the algorithm to decompose the tasks into control primitives is still to improve, the basic HRI tasks can all be controlled by the control modes discussed above.

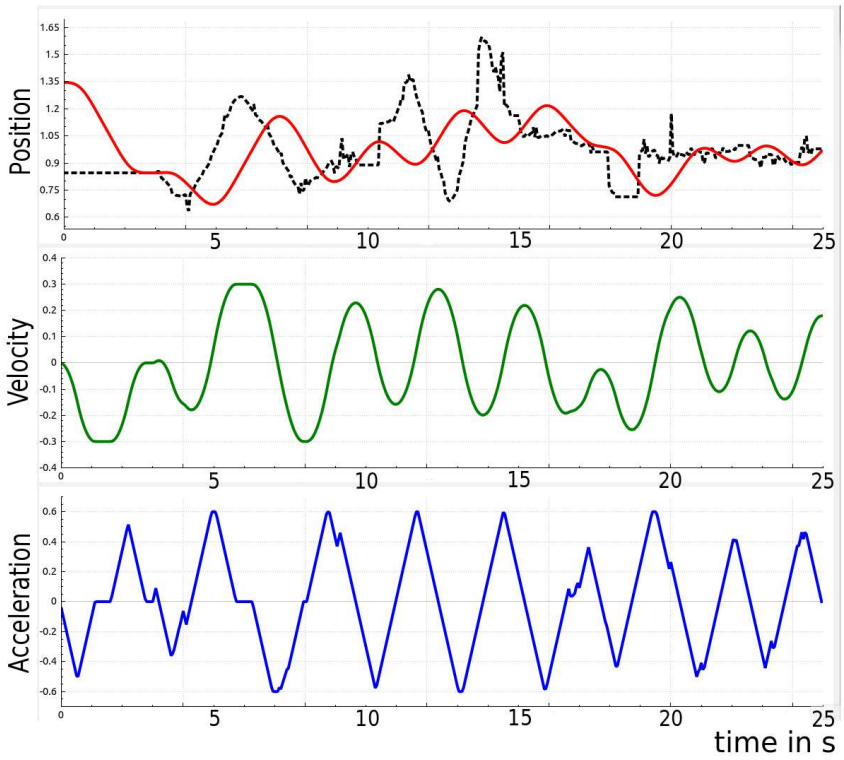


Figure 5.16: Results of robot tracking a target: position (in m), velocity (in m/s) and acceleration (in m/s^2) during the tracking for 25 seconds. The black dashed line is the target position, with noise of the 3D vision, and the red line is the position of the robot, which tracks the target with a delay. The positions of the robot are calculated from measured joint values and the kinematic model, while velocity and acceleration are estimated. The velocity, acceleration and jerk are always limited, maintaining a smooth tracking process.

5.5 Results and Comparison

We focus on some results on how the controller is integrated in a HRI manipulator. For the performance of the trajectory generator, readers may refer to [Broquere 08c].

Figure 5.16 shows the results of the target tracking by the trajectory controller, as in *Case 2*, over 25 seconds. For simplicity, only axis X is shown. The black dashed line is the

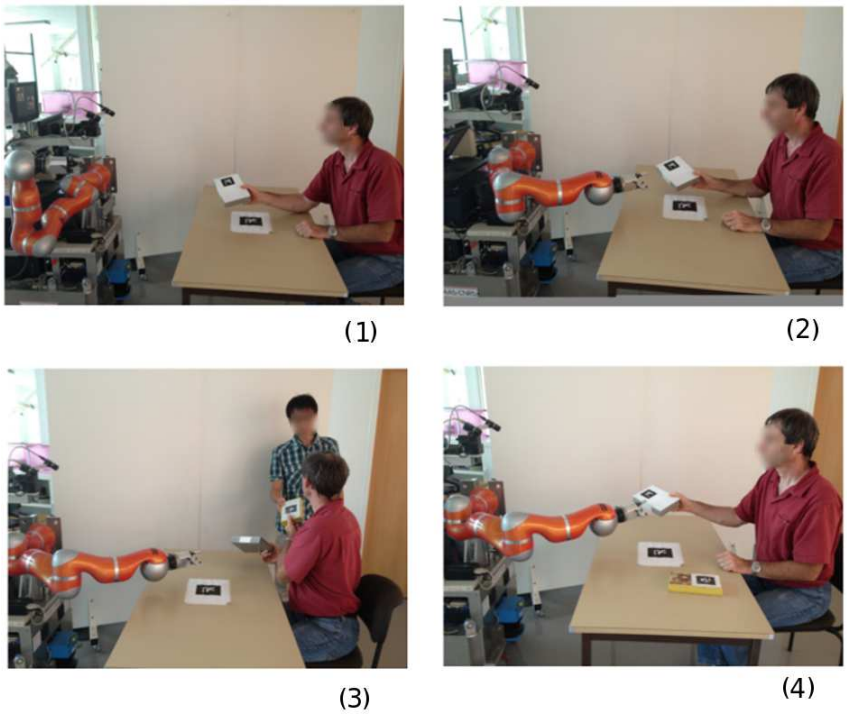


Figure 5.17: (1): The controller is given the task of receiving the object from human. It tracks the first segment in the robot frame. (2): The object is moving and the robot tracks a target. (3): Human is distracted by another human and the task is suspended. (4): Human returns to the task, and the robot resumes the task and grasps the object.

position of the target, generated by the 3D vision system. The red line is the position of the robot. The two bottom diagrams show the velocity and acceleration of the robot in the same period. Firstly, we can see that the controller produces robust behavior to the noise in the visual system. Secondly, the velocity and acceleration of the robot are saturated as type V trajectories and computed.

Finally, we show the behavior of the controller for a complete manipulation task. Figure 5.17 shows the scenario of the manipulation and Figure 5.18 shows the real position of the robot end effector in the robot frame (see figure 5.8 for the axes assignment of the robot base). The high-level task planner plans a task to receive the object. When the robot sees the object held by the human, the grasp planner calculates a valid grasp and the path planner with the trajectory generator plans the main trajectory for the robot to take the object.

The trajectory is divided into three segments by the controller, and different control modes are chosen. As we have seen above, each control primitive is associated to a trajectory segment. In this case, we obtain three segments, the first one is controlled in the robot frame, the second is defined as the tracking of the entry point of the third segment and the third segment is a trajectory defined in the object frame.

During the target tracking, human is distracted because a second human arrives and gives an object to him. High-level software detects this event by monitoring the visibility

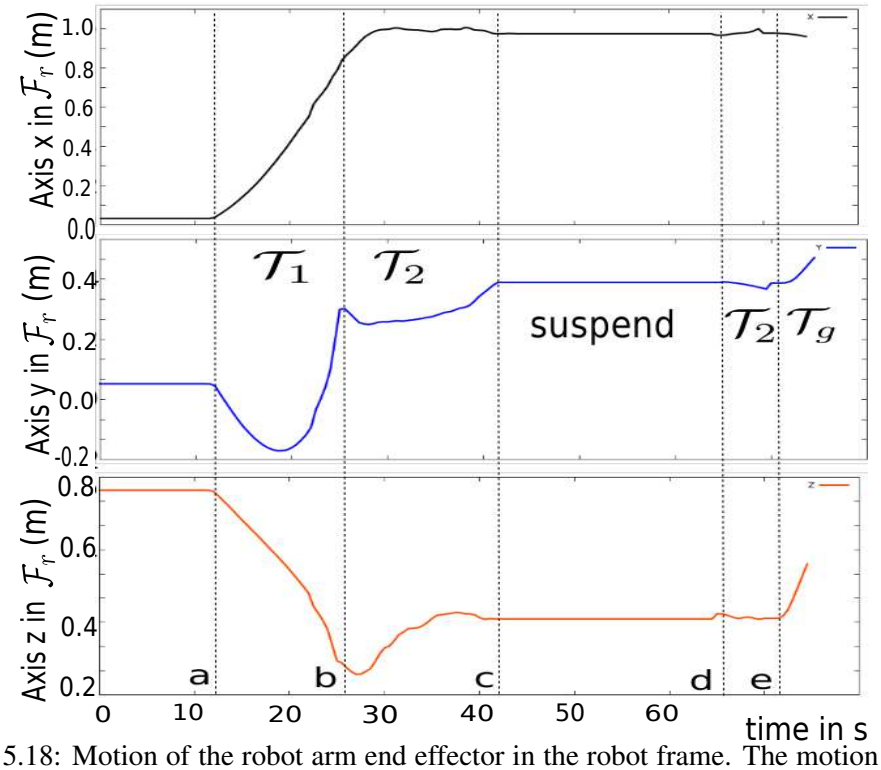


Figure 5.18: Motion of the robot arm end effector in the robot frame. The motion starts at time a; Between a and b: the controller tracks the first trajectory segment $\mathcal{T}_1$ in $\mathcal{F}_r$; From b to c and d to e: target tracking; From c to d: the task is suspended; From e to end: the grasping movement controlled in $\mathcal{F}_o$.

cost map of the human. Because of the event, the controller suspends the task. It resumes the tracking when the human look again at the robot and the object to exchange comes back in the reachable zone. Then, the grasping movement is finished. Note the performance of the target tracking process in the time intervals: between b to c, and between d to e. The controller finished the task reactively without the need of task or path replanning. The results shows that a reactive controller can be built based on Online Trajectory Generation, and as it is more responsive for the human, the robot is easier to interact with. Before the implementation of the reactive controller, the human needs to hold still the object for the robot to grasp successfully, now the robot can succeed when the human moves the hand during the exchange.

5.6 Conclusion

A reactive trajectory controller has been presented with some results relative to a robot grasping an object held by a human. The first results presented illustrate the versatility of the controllers based on online trajectory generation. In the example shown here, the controller switch between frames and suspend the control task during the time the human is distracted.

The trajectory controller proposed uses an online trajectory generator to build a trajectory to join up the trajectory to follow. It is very simple to use and implement and gives an efficient solution to follow trajectories and track moving objects in the HRI context. More precisely, it can adapt kinematic limits to the changing state of the scene and switch between trajectories and control modes.

The challenge is now to extend this type of trajectory controller and the concept of control primitives to manage forces, to handle events based on force sensing and to control dual arm manipulators.

6

Conclusion and perspectives

Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience.

—- Roger Bacon

6.1 Conclusion

This work is a part of the development of a service robot capable of interacting with humans in an unstructured environment. The high-level softwares on the robot plan the interaction tasks and the motion to accomplish the tasks. The main objective of this thesis is to implement the methodologies to provide the robot with the ability to react to the sensory information and events: mainly the visual tracking and force events. Trajectory control is proposed as the center of this function, and different techniques are used for the sensor fusion: nonlinear Kalman filter for the estimation and multi-modal tracking, and Relevance Vector Machine to detect the force event. Some simulation and experimental results show how sensor fusion and sensor based trajectory control can improve HRI manipulations.

6.1.1 Visual Servoing and Trajectory Based Control

One of the functions of the proposed system can be compared to classic visual servoing. For the trajectory control, only position of the objects and human body parts are needed for

the controller to achieve complex tasks. The author of this document argues that the trajectory generation based control is easier to implement with different sensor systems, such as different vision systems, or when the perception is obtained through the fusion of different sensors. Compared to visual servoing, another advantage is that trajectory based system can be easier to integrate with a path planner. With different Human-Robot Interaction specifications, stopping, slowing down, and accelerating on a trajectory can be also achieved while the robot stays on the path, guaranteeing collision free motion.

6.1.2 Force Sensing and Force Events

One of the challenges for service robots (compared to industrial robots) is that the dynamical model of the manipulated object is not known in advance. The need to explore the environment also demands the robot to discover new objects. While picking up a new object, playing in hands to feel what is the object made of and what it should contain is natural for human, the robot is also able to accomplish the same. To achieve this, an on-line method is proposed to estimate the inertial parameters, including the center of mass (COM) of the object. The inertial parameters can be used to identify the object, while the position of COM can be used further for the multi-modal tracking. This part is tested offline with data acquired on the robot.

One important result of the on-line estimation is that when the dynamics of object is simulated, the contact forces can be precisely computed. The contact forces between the manipulated object and the environment contains key information in particular for human robot object exchange. A classifier is proposed to solve the problem of synchronizing the grasping and the releasing of the object. Together with the estimation, the system proposed in this work is able to distinguish inertial forces, collision, and when a human grasps an object. The detected event is then used by the trajectory controller.

A special device has been designed to acquire data between people for object exchanges to train the classifier.

6.2 Perspectives

6.2.1 Sensor Fusion and Learning

Learning from demonstration is a promising area for service robots. The Bidule designed at *LAAS-CNRS* is also to teach motion and force control for object exchange. To continue the work presented in this thesis, building trajectories based on learned motion is a future work direction.

For the events classification based on force, more classes can be added, and the same philosophy can be implemented for other manipulations, for example, putting an object on the table based on events detection. For the same task, the estimated inertial model of the

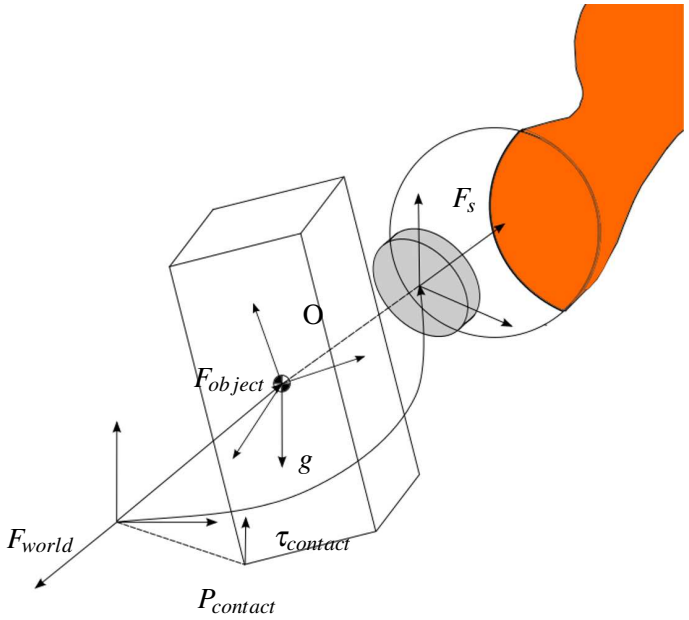


Figure 6.1: Contact Modeling by Force Sensing can benefit from the estimation of position of COM of the object. By reasoning on the external torques and the gravity, the robot can deduce which way to turn to adjust the angles so that object is well put on the surface.

object can help to model the contact, as shown by Figure 6.1. By modeling the contact forces when the robot puts an object on a table, force control laws can be designed to adjust the rotation of the object. One objective would be that the robot releases the object only when the object is stable on the table, which requires that one surface of the object is aligned with the surface of the table.

Although trajectory learning is not presented here, some recorded data are shown in this document. Figure 6.2 presents an exchange realized in normal conditions. In the top part of the figure, the positions of the Bidule object (blue), of the wrist of the giver (green) and of the wrist of the receiver (red) are drawn in the world frame. Only the movement of approaching is drawn in the figure. Two vertical black lines determine the exchange phase. This phase is characterized by the period when the hand of the giver, the object and the hand of the receiver are kinetically linked. During the first part of the record, the Bidule object is placed on a table between the two volunteers and it does not move. The giver grasps the object and executes a backward move before hands over the object to the receiver. After giving the object, the giver moves back his hand. At the end of the exchange phase, the giver brings back his hand. The giver must release the grasp at the end of the exchange phase. The approaching movement and the synchronization movement between the giver and receiver can be learned to achieve more natural movements. Learning from Demonstration is a dynamic research area, methods and strategies of which can be found in numerous documents, among which we have [Vakanski 12b], [Khansari-Zadeh 11] and surveys such as [Argall 09]. How to integrate learning techniques into a system based on

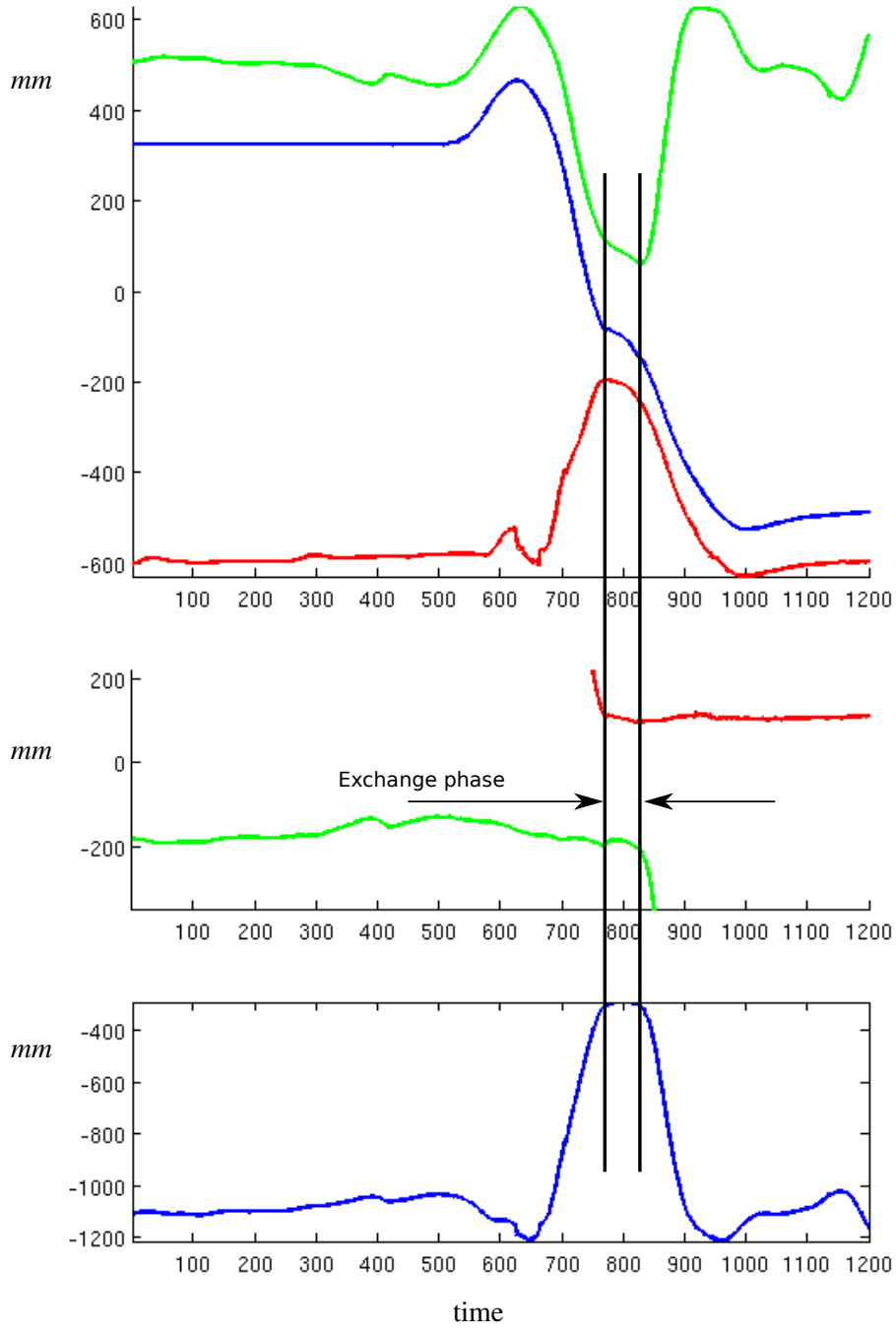


Figure 6.2: Hands and Bidule object trajectories during an exchange (over 12s). Top: the absolute position of Bidule (blue) the hand of the giver (green) and of the receiver (red). Middle: position of the hands relatively to Bidule. Bottom: The distance between the giver and the receiver. The two vertical black lines delimit the exchange phase. Time is expressed in motion capture periods (1kHz).

path planning is an interesting topic. For example, the robot can remember the successfully executed trajectories that it planned and use them later for similar situations.

6.2.2 Trajectory Control

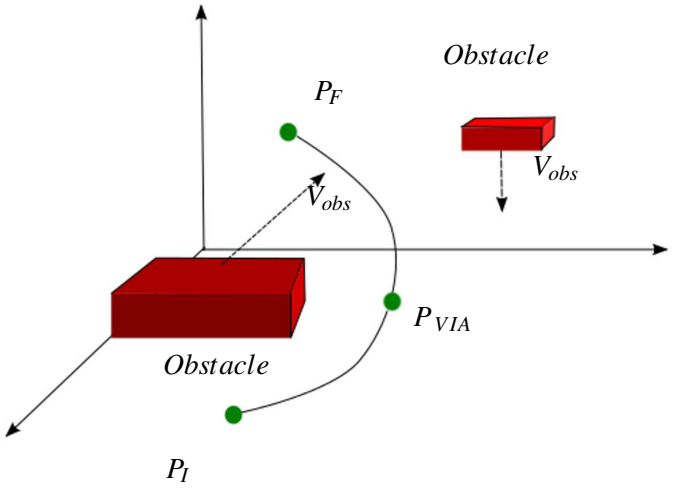


Figure 6.3: Via-points for obstacle avoidance. By adding a point P_{VIA} in the middle, and compute it based on the motion of obstacles, the trajectory passing through P_I, P_{VIA}, and P_F can avoid the obstacle and track the target.

One of the advantages of trajectory generation based control is the ability to react to different sensory events and information. However, on-line obstacle avoidance is still a challenge. Instead of tracking a point or simply tracking a trajectory, introducing a point between the starting point and the target and then generate a trajectory passing through this point would help to solve the problem, as shown in 6.3. In this case, the via-points trajectory is generated without a planned path (which is the case for the approach in Section 5.2, and the trajectory should be generated within a control sequence, all raising more challenges for the trajectory generation algorithm. New methods shall be investigated.

The whole body control of a mobile manipulator is another issue on which we are studying. The Human-Aware Motion Planner plans path for a robot to follow, including the base and the arms. While synchronized, the robot finishes complex tasks, such as navigation and manipulation in the same time, while avoiding obstacles. The problem requires more study because the navigation of the robot and the motion of the arms should slow down or stop to avoid moving obstacles, and the two should be synchronized. But the dynamic and precision of trajectory following of the robot base and arms are different, hence new strategies shall be proposed to achieve manipulation during navigation.

In Chapter V we see some results for the trajectory based manipulator control. And we believe that this middle-level Control Primitives, include trajectory control and impedance control policies, can be very flexible and open a wide range of solutions to build reactive controller for HRI context.

A

Nonlinear Kalman Filters

Kalman filter is a well-established technique which has application in various areas. We give the basic formulation of KF in this appendix, as a necessary background to understand the Unscented Kalman Filter employed to solve two problems in this thesis.

A.1 Discrete Kalman Filtering

Although equations which define the evolution of a physic system are normally continuous in time, the Kalman filter is almost always implemented in discrete time. The basic linear Kalman filter is based on linear dynamic system models, discretized in the time domain. The formulation is a Markov chain built on a linear operator, pertubated by Gaussian noises. To simplify the notations, we define: $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$. When the dynamic system is linear and can be modelled as:

$$\boldsymbol{x}_k = \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{G}_k \boldsymbol{v}_k \tag{A.1}$$

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{D}_k \boldsymbol{n}_k \tag{A.2}$$

In which:
$\boldsymbol{y}$: the observation;
$\boldsymbol{x}$: the state vector to estimate;
$\boldsymbol{F}_k$: the state transition model which is applied to the previous state;
$\boldsymbol{B}_k$: the control-input model which is applied to the control vector $\boldsymbol{u}_k$.
$\boldsymbol{H}_k$: the observation model.
$\boldsymbol{G}_k$ and $\boldsymbol{D}_k$: transition matrix for noises. A basic assumption in the development of the

Kalman filter is that the noises, $\boldsymbol{v}_k$, and $\boldsymbol{n}_k$, on the process and observation, are all Gaussian, uncorrelated, and zero-mean:

$$E(\boldsymbol{v}_k) = \mathbf{0} \tag{A.3}$$
$$E(\boldsymbol{n}_k) = \mathbf{0} \tag{A.4}$$

with known covariance.

$$E(\boldsymbol{v}_i \boldsymbol{v}_j^T) = \delta_{ij} R_v \tag{A.5}$$
$$E(\boldsymbol{n}_i \boldsymbol{n}_j^T) = \delta_{ij} R_n \tag{A.6}$$

And the process and observation noises are uncorrelated. The Kalman filter proceeds recursively in two stages: prediction and update.

Prediction: A prediction of the state vector to estimate, written as $\hat{\boldsymbol{x}}_{k|k-1}$, and its covariance matrix $\boldsymbol{P}_{k|k-1}$ at time k is computed according to:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}(k)\boldsymbol{x}_{k-1\hat{|}k-1} + \boldsymbol{B}_k \boldsymbol{u}_k \tag{A.7}$$
$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{G}_k R_v \boldsymbol{G}_k^T \tag{A.8}$$

Update When new observation is available, the new estimation is updated by:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{W}_k [y_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1}] \tag{A.9}$$
$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{W}_k \boldsymbol{S}_k \boldsymbol{W}_k^T \tag{A.10}$$

where the gain matrix $\boldsymbol{W}_k$ is computed as:

$$\boldsymbol{W}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k \boldsymbol{S}_k^T \tag{A.11}$$

in which:

$$\boldsymbol{S}_k = \boldsymbol{R}_u + \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k \tag{A.12}$$

is called the innovation covariance, which is often important in data association.

A.2 Extended Kalman Filter

We consider the problem of estimation the hidden state $\boldsymbol{x}_k$ of a discrete-time nonlinear dynamic system:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{w}, \boldsymbol{v}_k) \tag{A.13}$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{w}, \boldsymbol{n}_k) \tag{A.14}$$

where $\boldsymbol{x}_k$ represents the system states and $\boldsymbol{y}_k$ the observed signal for the given system model.The process noise and observation noise are presented by $\boldsymbol{v}_k$ and $\boldsymbol{n}_k$, their covariance matrices are $\boldsymbol{R}_v$ and $\boldsymbol{R}_n$. In general, they do not necessarily be additive. $\boldsymbol{w}$ are the fixed parameters. The system model $\boldsymbol{f}()$ and $\boldsymbol{h}()$ are assumed nonlinear and known. Kalman

filter achieves a recursive maximum-likelihood estimation of $\boldsymbol{x}_k$. Given the observation $\boldsymbol{y}_k$, the recursive estimation for $\boldsymbol{x}_k$ is given as:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k) \tag{A.15}$$

$$\boldsymbol{P}_{x_k} = \boldsymbol{P}_{x_{k|k-1}} - \boldsymbol{K}_k \boldsymbol{P}_{\tilde{y}_k} \boldsymbol{K}_k^T \tag{A.16}$$

Where $\boldsymbol{K}_k$ is a gain computed by equation A.19. $\hat{\boldsymbol{x}}_{k|k-1}$ is the optimal prediction at time k with all the observation at time $k-1$. $\hat{\boldsymbol{y}}_k$ is the optimal prediction of observation at time k, $\boldsymbol{P}_{x_{k|k-1}}$ the covariance of $\hat{\boldsymbol{x}}_{k|k-1}$, and $\boldsymbol{P}_{\tilde{y}_k}$ the covariance of innovation which is difined as $\tilde{\boldsymbol{y}}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_k$. The optimal prediction values are calculated by

$$\hat{\boldsymbol{x}}_{k|k-1} = E[\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{w}, \boldsymbol{v}_{k-1})] \tag{A.17}$$

$$\hat{\boldsymbol{y}}_k = E[\boldsymbol{h}(\hat{\boldsymbol{x}}_k, \boldsymbol{w}, \boldsymbol{n}_k)] \tag{A.18}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{x_k, y_k} \boldsymbol{P}_{\tilde{y}_k}^{-1} \tag{A.19}$$

Kalman filter calculate these quantities in the linear case. For a nonlinear system, extended Kalman Filter (EKF) [Chui 98] approximate these by linear Taylor approximation, as:

$$\hat{\boldsymbol{x}}_{k|k-1} \approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{w}, \boldsymbol{v}_{k-1}) \tag{A.20}$$

$$\hat{\boldsymbol{y}}_k \approx \boldsymbol{h}(\hat{\boldsymbol{x}}_k, \boldsymbol{w}, \boldsymbol{n}_k) \tag{A.21}$$

$$\boldsymbol{K}_k \approx \boldsymbol{P}_{x_k, y_k} \boldsymbol{P}_{\tilde{y}_k}^{-1} \tag{A.22}$$

In which $\boldsymbol{P}_{x_k, y_k}$, $\boldsymbol{P}_{\tilde{y}_k}$ are the covaraince of the approximate calculated values, which means, for example, $\boldsymbol{x}$ and $\boldsymbol{P}_x$ are calculated as:

$$\boldsymbol{F}_{k-1} = \frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{x}_{k-1}}|_{\hat{\boldsymbol{x}}_{k-1}} \tag{A.23}$$

$$\boldsymbol{B}_{k-1} = \frac{\partial \boldsymbol{f}_{k-1}}{\partial v_{k-1}}|_{\hat{\boldsymbol{v}}_{k-1}} \tag{A.24}$$

$$\hat{\boldsymbol{x}}_{k|k-1} \approx \boldsymbol{F}_{k-1}(\hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{B}_{k-1}(\hat{\boldsymbol{v}}_{k-1}) \tag{A.25}$$

$$\boldsymbol{P}_{x_{k|k-1}} \approx \boldsymbol{F}_{k-1} \boldsymbol{P}_{x_{k-1}} \boldsymbol{F}_{k-1}^T + \boldsymbol{B}_{k-1} \boldsymbol{R}_v \boldsymbol{B}_{k-1}^T \tag{A.26}$$

As such, the EKF can be viewed as a first-order approximation to the optimal term. But as argued in [Julier 96], it has flaws, these approximations can result in large errors and even divergence of the filter.

B

Quaternions and Rotations

Quaternions gives a compact and effective representation for three dimentional rotations. This annexe gives the basics of quaternion, its relations with several other common representations and stops at the perturbations and time-derivatives of quaternions, which are used in this thesis for 6D tracking of object.

B.1 Axis-Angle Representation

The axis-angle representation parametizes the rotation of a rigid body in a three dimentional space by two values: a unit vector $\boldsymbol{u}$ which defines the direction of rotation, and a rotation angle ϕ the magnitude. Axis-angle is useful to interpolate rotations of rigid body and easy to convert from and to quaternions.

B.2 Definition of Quaternion

Quaternions can be seen as the extension of the complex numbers. A quaternion q is written as:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{B.1}$$

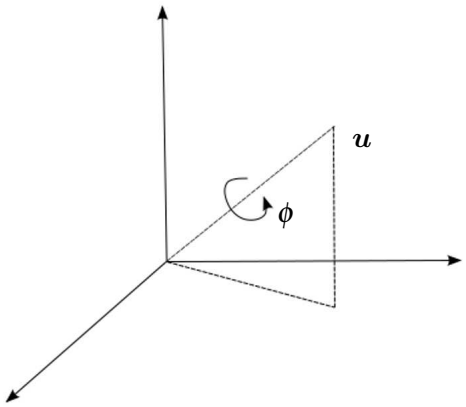


Figure B.1: Axis-Angle Representation

where $q_0, q_0, q_0, q_0 \in \mathbb{R}$, and i, j, k are defined so that:

$$\begin{aligned} i^2 &= j^2 = k^2 = -1 \\ ij &= -ji = k \\ jk &= -kj = i \\ ki &= -ik = j \end{aligned} \tag{B.2}$$

Quaternions can be written in vector representation, simply as:

$$q = q_0 + \overrightarrow{q} = (q_0, \overrightarrow{q}) \tag{B.3}$$

In which $\overrightarrow{q}$ is the imaginary or vector part of quaternion. While complex numbers with unit lengh, written as $z = e^{i\theta}$ can encode rotations in the 2D plane, quaternions of unit length encode rotations in 3D space, although the computation is not as straightforward as for complex numbers. Given the rotation in vector-angle form, $\boldsymbol{v} = \phi \boldsymbol{u}$, a rotation of ϕ rad along the axis given by the unit vector $\boldsymbol{u} = (u_x, u_y, u_z)$, we have the unit quaternion to represent the rotation:

$$q = (cos(\phi/2), \boldsymbol{u} sin(\phi/2)) \tag{B.4}$$

And

$$\phi = arctan(\| q \|, a) \tag{B.5}$$

$$\boldsymbol{u} = q/\| q \| \tag{B.6}$$

The multiplication of two quaternions is defined as:

$$q = \tilde{q} \bigotimes \bar{q} = \tilde{Q}\bar{q} \tag{B.7}$$

In which $\tilde{q} = [\tilde{q_1}, \tilde{q_2}, \tilde{q_3}, \tilde{q_4}]$, $\bar{q} = [\bar{q_1}, \bar{q_2}, \bar{q_3}, \bar{q_4}]$, and $\tilde{Q}$ is the matrix:

$$\tilde{Q} = \begin{bmatrix} \tilde{q_1} & -\tilde{q_2} & -\tilde{q_3} & -\tilde{q_4} \\ \tilde{q_2} & \tilde{q_1} & -\tilde{q_4} & \tilde{q_3} \\ \tilde{q_3} & \tilde{q_4} & \tilde{q_1} & -\tilde{q_2} \\ \tilde{q_4} & -\tilde{q_3} & \tilde{q_2} & \tilde{q_1} \end{bmatrix} \tag{B.8}$$

B.3 Rotation matrix

Rotation matrix is used commonly in numerical computation libraries. Given a rotation vector $\boldsymbol{v}$,

$$\boldsymbol{R} = e^{[\boldsymbol{u}]_\times} \tag{B.9}$$

where the operator $[\bullet]_\times$ is operator defined by:

$$[\boldsymbol{u}]_x \triangleq \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{B.10}$$

Rotation matrix $\boldsymbol{R}$ is then given as:

$$\boldsymbol{R} = \begin{bmatrix} cos(\theta) + u_x^2(1-cos\theta) & u_x u_y(1-cos\theta) - u_z sin\theta & u_x u_z(1-cos\theta) + u_y sin\theta \\ u_y u_x(1-cos\theta) - u_z sin\theta & cos(\theta) + u_y^2(1-cos\theta) & u_y u_z(1-cos\theta) + u_x sin\theta \\ u_z u_x(1-cos\theta) - u_y sin\theta & u_z u_y(1-cos\theta) + u_x sin\theta & cos(\theta) + u_z^2(1-cos\theta) \end{bmatrix} \tag{B.11}$$

Which can be written as:

$$\boldsymbol{R} = \boldsymbol{I} cos\theta + sin\theta[\boldsymbol{u}]_\times + (1-cos\phi)\boldsymbol{u}\bigodot\boldsymbol{u} \tag{B.12}$$

where $\bigodot$ is the tensor product (which is often written as $\bigotimes$, which being used to represent quaternion product in this document), and defined as:

$$\boldsymbol{u}\bigodot\boldsymbol{u} = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix} \tag{B.13}$$

B.4 Rotations and Compositions

Write a rotation in quaternion as q, and in rotation matrix as R. The rotation applied to a vector $\boldsymbol{x}$ into a new vector $\boldsymbol{x}'$ is given by:

$$\bar{\boldsymbol{x}}' = q\bigotimes\bar{\boldsymbol{x}}\bigotimes q* \tag{B.14}$$

$$\boldsymbol{x}' = R\boldsymbol{x} \tag{B.15}$$

with:

$$\bar{\boldsymbol{x}} = [0 \;\; \boldsymbol{x}^T]^T \tag{B.16}$$

And $q* = (q_0, -q_1, -q_2, -q_3)$ is the conjugate quaternion. The composition of two rotations are straigtforward with the definiton of multiplication of quaternions introduced above:

$$q = \tilde{q} \bigotimes \bar{q} \tag{B.17}$$

$$R = \tilde{R}\bar{R} \tag{B.18}$$

Another useful representation is homogeneous transformation matrix, which is defined as:

$$T = \begin{bmatrix} & & & x \\ & R_{3\times 3} & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.19}$$

For two frames, F_2 is obtained by translation $trans = [x, y, z]$ and rotation R from F_1, then the transformation matrix is written as above. If we write the pose of a point in a frame $F1$ as $P_1 = [P_x, P_y, P_z, 1]$, then the pose of this point in frame $F2$ is given as:

$$P_2 = TP_1 \tag{B.20}$$

And transformation matrix has the same composition rule:

$$T = \tilde{T}\bar{T} \tag{B.21}$$

B.5 Perturbations and Derivatives

When using quaternions to represent rotations and build a dynamic model for motion, one important aspect is the computation of perturbations and time-derivatives. Given a quaternion q, and the perturbation written as Δq, expressed in the local body frame. Then the new quaternion can be written as:

$$\tilde{q} = q \bigotimes \Delta q \tag{B.22}$$

The same for rotation matrix:

$$\tilde{R} = R \bigotimes \Delta R \tag{B.23}$$

In the case the perturbation angle $\Delta\theta$ is small ($\Delta\theta$ represents the rotation around an axis $\boldsymbol{u}$), then the perturbation quaternion and rotation matrix can be approximated by the first terms

of the Taylor expansion. Which means:

$$\Delta q = \begin{bmatrix} 1 \\ \frac{1}{2}\Delta\theta \end{bmatrix} + \mathscr{O}(|\Delta\theta|^2) \tag{B.24}$$

$$\Delta R = \boldsymbol{I} + [\Delta\theta]_\times + \mathscr{O}(|\Delta\theta|^2) \tag{B.25}$$

where $\mathscr{O}(|\Delta\theta|^2)$ is the remainder of Taylor expansion. If at time $t = kT$, the rigid body rotation is written as $q = q(t)$, and $\tilde{q} = q(t+\Delta t)$, the derivative of $q(t)$ given as:

$$\frac{dq(t)}{dt} \triangleq \lim_{\Delta t \to 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} \tag{B.26}$$

And writing the angular velocity as $\omega(t)$, expressed in local body frame. The development of the derivative is given as:

$$\begin{aligned}
\dot{q} &\triangleq \lim_{\Delta t \to 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{q \bigotimes \Delta q - q}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{Q(\Delta q)q - q}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{(\boldsymbol{I} + \frac{1}{2}\begin{bmatrix} 0 & -\Delta\theta^T \\ \Delta\theta & -[\Delta\theta]_\times \end{bmatrix} q + \mathscr{O}(|\Delta\theta|^2))q - q}{\Delta t} \\
&= \lim_{\Delta t \to 0} \frac{\frac{1}{2}\begin{bmatrix} 0 & -\Delta\theta^T \\ \Delta\theta & -[\Delta\theta]_\times \end{bmatrix} q + \mathscr{O}(|\Delta\theta|^2)}{\Delta t} \\
&= \frac{1}{2}\begin{bmatrix} 0 & -\omega^T \\ \omega & -[\omega]_\times \end{bmatrix} q
\end{aligned} \tag{B.27}$$

$$\tag{B.28}$$

The details of this matrix is given as:

$$\Omega(\omega) \triangleq \begin{bmatrix} 0 & -\omega^T \\ \omega & -[\omega]_\times \end{bmatrix} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{B.29}$$

C

Wavelet Analysis

Wavelet Transforms can be considered as computing a time-frequency representation for a given signal, in 1D for a time signal, or 2D for image and more. Wavelet Analysis is often compared to Fourier Analysis, as for the two the transforming function are all orthogonal and the computation are based on convolutions. The main difference and often advantage of wavelet analysis is that it achieves not only frequency information but also the frequency information can be localized in time. The Windowed Fourier Transform can also compute local-frequency information, and a short introduction is given in the next section.

C.1 Windowed Fourier Transform

The classic Fourier transform decomposes a signal into its harmonic components. Employed in feature extraction, its major disadvantage is that while extracting the information in frequence space, the time information is lost. For a given time window of monitoring, Fourier transform of a signal in which an event arrives at the beginning is the same as the Fourier transform of a signal in which the same event arrives at the end of the time window. Only stationary signal is suitable to use Fourier transform for feature extraction. Windowed Fourier transform (WFT) is an analysis tool to extract time-frequency information from a signal. The Fourier transform is computed on a sliding window in time. The length of this time window can be chosen, and the length defines the precision of the information in time, for all the spectrum.

As discussed in various works, the WFT has a main problem of imposing an interval

into the analysis, the choice of which is often not evident and need a trial-and-error process. The aliasing of high-frequency and low-frequency components that do not fall into the time intervals would also cause inacurrancy. The fact that the same computation is carried out for all frequency band and at each time step rises the problem of inefficiency too. As we will show in the wavelet part of this documents, the fact that the transforming functions could also have an influence on the results, which is even more important if the results are used to extract features from signals. In the WFT, the base functions stay the same. The different time-frequency tiling between Fourier transfor, Windowed Fourier transform and wavelet transform can be shown as FigureC.1.

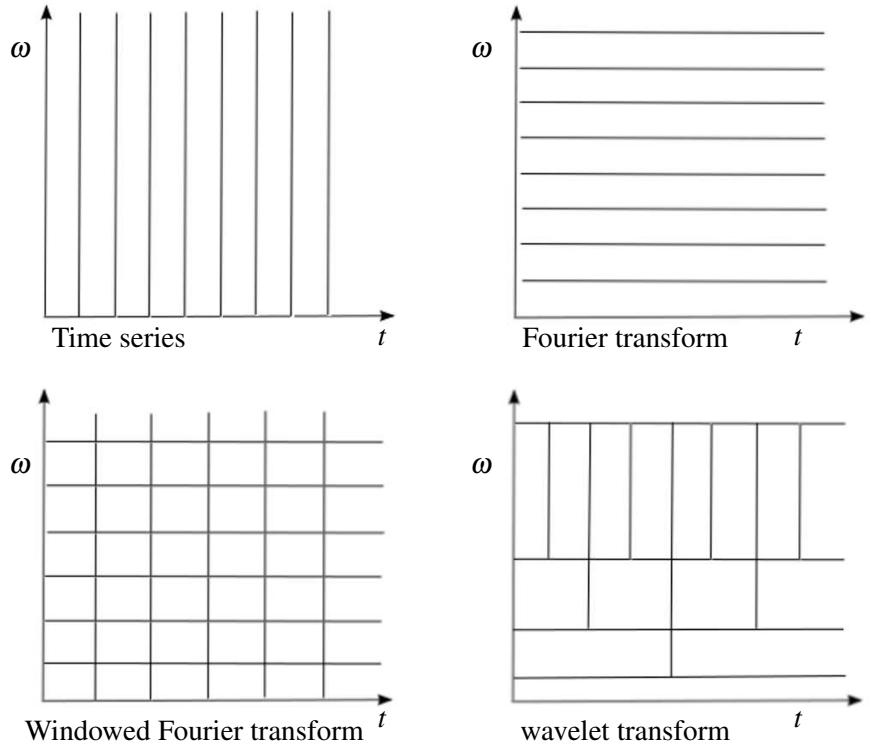


Figure C.1: Time-frequency tiling for Windowed Fourier Transform and wavelet transform.

C.2 Continuous Wavelet Transforms

For a signal in continuous time domain, with finite energy, the wavelet transform project the signal on a continuous family of frequency bands. The projection is computed through convolution between the original signal and a function called wavelet function, which is generated from the so-called mother wavelet. The mother wavelet is scaled and shifted (also called dilated and translated), which enable the computation to localize information in

time and in frequency, producing a time-frequency representation. The wavelets produced by scaling and shifting defines an orthonormal basis of $\mathbf{L}^(\mathbb{R})$.
Given a mother wavelet, $\psi(t)$, a wavelet is constructed by:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{C.1}$$

where $a \in \mathbb{R}^+$ defines the scale and $b \in \mathbb{R}$ defines the shift. The projection of a signal, $x(t)$, onto a subspace of scale a, is computed through:

$$x_a t = \int_{\mathbb{R}} WT_\psi\{x\}(a,b)\dot{\psi}_{a,b}(t)db \tag{C.2}$$

In which, $WT_\psi\{x\}(a,b)$ are wavelet coefficients, computed by:

$$WT_\psi\{x\}(a,b) = \langle x, \psi_{a,b}\rangle = \int_{\mathbb{R}}\{x\}(a,b)\psi_{a,b}(t)dt \tag{C.3}$$

The mother wavelet $\psi(t)$ can be chosen according to many different criterions. The wavelet coefficients can be analysised and displayed by a scaleogram, which is often compared to the spectrum diagram of the Fourier transform, but integrates also time information.

C.3 Discrete Wavelet Transforms

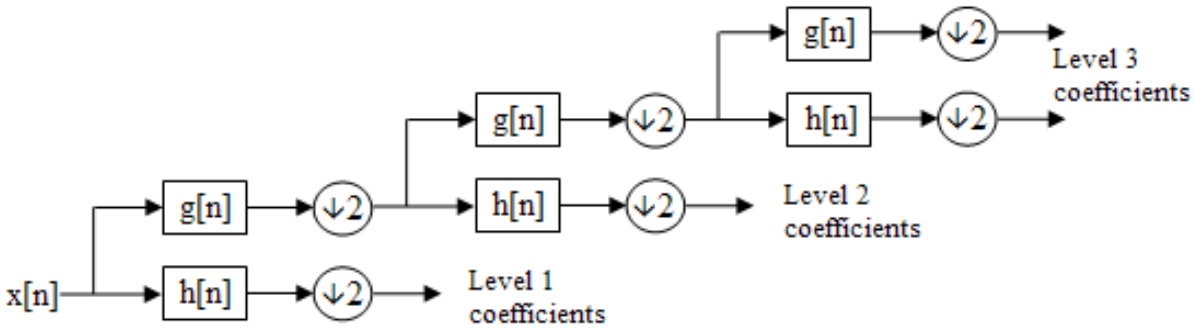


Figure C.2: Windowed Fourier Transform and wavelet transform

For a discrete signal, the wavelet transform can be carried out by passing through a series of filters. This set of filters is sometimes called fiter bank. The wavelet transform is also called decomposition since it decomposes the signal into different spectrum ranges. Firstly, the original signal is passed thourgh a high-pass filter, h, and at the same time low

pass filter g. The computation is convolution in discrete time. For a signal $x[n], n \in \mathbb{N}$

$$y[n] = (x*g)[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k] \tag{C.4}$$

$$d[n] = (x*h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{C.5}$$

producing signal $y[n]$ as the approximate of the original signal, and $d[n]$ as the detail of the signal. The two signals are then downsampled by a factor of 2. Then $d[n]$ is passed through the same computation as original signal $x[n]$. Which is called cascading. The computation can be given as in Figure C.2.

Because each transform devides the signal into a high frequency part and a low frequency part, the signals will be devided into 2^N parts. N is the level of decomposition, which requires that the original signal is a multiple of 2^N elements. The construction of filter banks is also through shifting and scaling from a discrete mother wavelet. In this work, we use lifting to implement the discrete wavelet transforms to optimize the computation.

D

Sparse Kernel Machines

Before introducing the Relevence Vector Machine (RVM), we will try to give a short explaination on Support Vector Machine (SVM), the two all being Sparse Kernel Machines. Without trying to give a better presentation or precise theoritical proves, this appendix is only trying to provide an easy reference for the reading of this document. The SVM is firstly introduced for linearly seperable data problems, to explain the concept of maximazation of margin. Then the Kernel trick, being vital to understand SVM and RVM is presented, followed by the formulation of RVM.

D.1 Support Vector Machine

Support Vector Machine(SVM) became a popular tool since some years for problems of classification and regression. Firstly introduced by Cortes and Vapnik ([Cortes 95]), it is applied in different fields due to its clear mathematical formulation and several powerful software implementation provided by researchers([Chang 11]).

We limit the discussion as a problem of classification of two classes. The training data set $\mathbf{Z} \in \mathbb{R}^{L \times D}$ containins L data pair $\{\boldsymbol{x}_i, y_i\}$. In which, vector $\boldsymbol{x} \in \mathbb{R}^D$ represents the features of dimension D, and $y_i \in \{+1, -1\}$ is the labels which indicate the classes of features. The basic classification problem would be to build a model can predict the class y^* given a new data x^*.

D.1.1 Linear SVM

We assume that the data are linearly seperable, which means that in the data space, we can draw a line (a hyperplane) which seperates data into two classes. The hyperplane can be written as:

$$w\dot{x} = b \tag{D.1}$$

where $\boldsymbol{w}$ is normal to the hyperplane, and $\frac{b}{\boldsymbol{w}}$ is the distance of hyperplane to the origin.

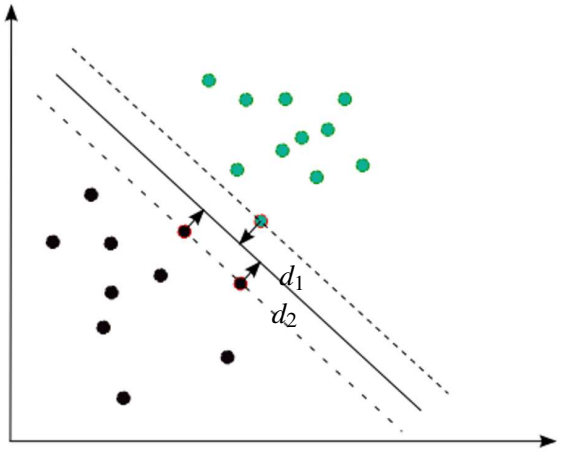


Figure D.1: Hyperplane through two classes. The points with arrows are the *support vectors*.

A linear SVM tries to solve this problem by finding a maximum-margin hyperplane that divides the input points by classes. Implementing a SVM is then to select $\boldsymbol{w}$ and b so that:

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 \text{ for } y_i = +1 \tag{D.2}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 \text{ for } y_i = -1 \tag{D.3}$$

Or written as:

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i \tag{D.4}$$

The support vectors are those points that lie closest to the hyperplane, which can be described by:

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b = +1 \text{ for } y_i = +1 \tag{D.5}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b = -1 \text{ for } y_i = -1 \tag{D.6}$$

As shown in Figure D.1, we define d_1 as the distance from support vectors from class I to

the hyperplane, and d_2 as the distance of class II, then we have the margin, the hyperplane's equidistance from the two classes, equal to $\frac{1}{\|\boldsymbol{w}\|}$. To maximize this margin with respect to the constraint of D.4 is equivalant as finding:

$$\min \|\boldsymbol{w}\| \quad \text{such that} \quad y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i \tag{D.7}$$

Minimizing $\|\boldsymbol{w}\|$ is equivalant to minimizing $\frac{1}{2}\|\boldsymbol{w}\|$ and we have:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{such that} \quad y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i \tag{D.8}$$

This problem is then solved by introduce the Lagrange multipliers $\boldsymbol{\alpha}$, where $\alpha_i \geq 0 \forall i$:

$$\begin{aligned} L_P &= \frac{1}{2}\|\boldsymbol{w}\| + \boldsymbol{\alpha}[y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i] && \text{(D.9)} \\ &= \frac{1}{2}\|\boldsymbol{w}\| + \sum_{i=1}^{L}[\alpha_i y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1] && \text{(D.10)} \\ &= \frac{1}{2}\|\boldsymbol{w}\| + \sum_{i=1}^{L}\alpha_i y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{L}\alpha_i && \text{(D.11)} \end{aligned}$$

Which is then solved as a normal optimization problem, by differentiating L_P w.r.t. $\boldsymbol{w}$ and b and setting the derivatives to zero:

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = 0 \quad \Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{L}\alpha_i y_i \boldsymbol{x}_i \tag{D.12}$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{L}\alpha_i y_i = 0 \tag{D.13}$$

Substituiting D.13 and **??** into D.11, gives to maximize:

$$\begin{aligned} L_D &= \sum_{i=1}^{L}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \ \text{ s.t. } \ \alpha_i \geq 0, \forall i, \sum_{i=1}^{L}\alpha_i y_i = 0 && \text{(D.14)} \\ &= \sum_{i=1}^{L}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i H_{ij}\alpha_j \ \text{ where } \ H_{i,j} = y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j && \text{(D.15)} \end{aligned}$$

In this new formulation, L_D is called Dual form of the primary L_P. It is worth noting that the Dual form requires only the dot product of each input vector x_i to be calculated, this is important for the Kernel Trick described in the fourth section. The problem is then given by:

$$\max_{\alpha}[\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha}] \ \text{ s.t. } \ \alpha_i \geq 0 \forall i \ \text{ and } \ \sum_{i=1}^{L}\alpha_i y_i = 0 \tag{D.16}$$

Which is solved as a convex quadratic optimization problem. For any data point which is a

support vector, writed as x_s:

$$y_s(\boldsymbol{x}_s \cdot \boldsymbol{w}_s + b) = 1 \tag{D.17}$$

Then:

$$y_s(\sum_{m \in \mathscr{S}} \alpha_m y_m \boldsymbol{x}_s \cdot \boldsymbol{w}_s + b) = 1 \tag{D.18}$$

Where $\mathscr{S}$ denotes the set of indices of the support vectors. Equation D.18 is used to calculate b. For a data set which is not fully seperatable linearly to be solved by a linear SVM, the concept of soft margin is then introduced. The formulation D.3 and D.3 is changed by introducing a positive slack variable $\xi_i, i = 1, 2...L$:

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i \text{ for } y_i = +1 \tag{D.19}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i \text{ for } y_i = -1 \tag{D.20}$$

which can be combined into:

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i \geq 0 \text{ where } \xi_i \geq 0 \forall i \tag{D.21}$$

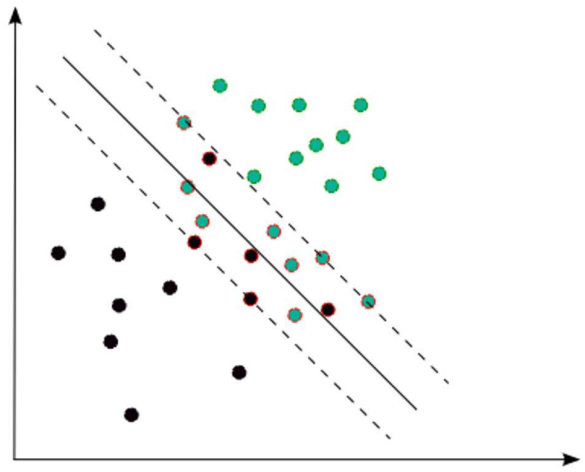


Figure D.2: Soft margin SVM will produce a classifier which minimizes the number of misclassifications.

This is called a soft margin SVM, for which data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. As we are trying to reduce the number of misclassifications, a sensible way to adapt our objective function is

to find:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L}\xi_i \quad \text{such that} \quad y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \forall i \tag{D.22}$$

where the parameter C controls the trade-off between the slack variable penalty and the size of the margin. Reformulating it as a Lagrangian for which, as before, we need to minimize with respect to $\boldsymbol{w}$, b and ξ_i and maximize the objective with respect to $\boldsymbol{\alpha}$:

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}[y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i] - \sum_{i=1}^{L}\mu_i\xi_i \tag{D.23}$$

in which, $\alpha_i \geq 0, \mu_i \geq 0 \; \forall i$. This is then solved by differentiating the term w.r.t $\boldsymbol{w}, b$ *and* ξ_i and setting the derivatives to zero.

D.1.2 The Kernel Trick and Nonlinear SVM

In the previous section, we apply SVM to linearly seperable data, by creating a matrix $\boldsymbol{H}$ from the dot product of the input variables:

$$\boldsymbol{H}_{ij} = y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j = \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{D.24}$$

in which, $\boldsymbol{H}$ is an example of a family of functions which are called kernel functions, and equation D.24 is called the Linear Kernel. The other kernel functions are also based on the calculation of the inner product of inputs. This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $\boldsymbol{x} \Rightarrow \boldsymbol{\phi}(\boldsymbol{x})$, only inner products of the mapped inputs in the feature space need to be determined without us needing to explicitly calculate $\boldsymbol{\phi}$.

The kernel trick is used because many classification problems (the same as regression) are not seperable in space $\boldsymbol{x}$ but might be seperable in a higher dimensional spcae which is projected by a suitable mapping $\boldsymbol{x} \Rightarrow \boldsymbol{\phi}(\boldsymbol{x})$.

D.2 Relevance Vector Machine

The Relevance Vector Machine (RVM) takes a Bayesian processing treatment to solve the regression and classification problem. We will introduce the method by regression and then generalize it to classification.

D.2.1 Evidence Approximation Theory

Given inputs $\boldsymbol{x_i}, 0 \leq i \leq N$, and output y_i, in which N is the number of data points. A problem of linear regression is to find parameter vector $\boldsymbol{w}$ and offset c based on the data, so

$$\Phi(\boldsymbol{x}) = \Phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) = (z_1, z_2, z_3) = \boldsymbol{z}$$

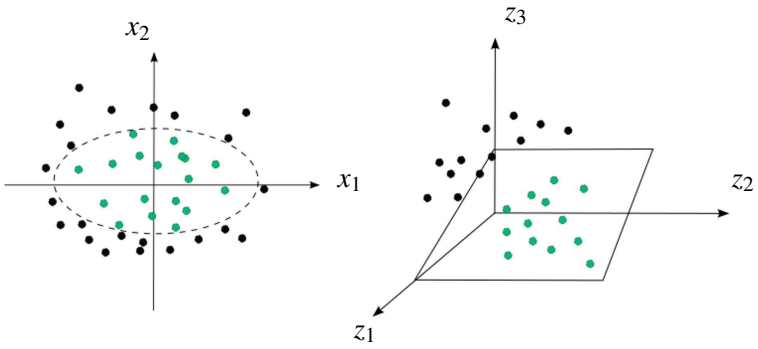


Figure D.3: The kernel trick transforms data to a higher dimensional space. In this example, the projection is defined by function $\boldsymbol{z} = \Phi\boldsymbol{x}$.

that we can predict y for a unknown input x by:

$$y = \boldsymbol{w}^T\boldsymbol{x} + c \tag{D.25}$$

In practice, we usually incorporate c into $\boldsymbol{w}$. For a nonlinear regression problem, a nonlinear mapping $\boldsymbol{x} \mapsto y$ is introduced:

$$y = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) \tag{D.26}$$

And the mapping $\boldsymbol{x} \mapsto y$ is called basis function. We then suppose that the training data is representative of the true output y_i and an addictive noise ε_i, and write t_i as the target

$$t_i = y_i + \varepsilon_i \tag{D.27}$$

where ε_i are independent samples of a Gaussian noise process with zero means and of variance σ^2. Which means then:

$$P(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \sigma^2) \sim N(y_i, \sigma^2) \tag{D.28}$$

$$= (2\pi\sigma^2)^{\frac{1}{2}} \exp\{-\frac{1}{2\sigma^2(t_i - y_i)^2}\} \tag{D.29}$$

$$= (2\pi\sigma^2)^{\frac{1}{2}} \exp\{-\frac{1}{2\sigma^2(t_i - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}))^2}\} \tag{D.30}$$

With all the training data, writing Φ is a matrix constructed such that the ith row represents the vector $\phi \boldsymbol{x}_i$, we then have:

$$P(t_i|\boldsymbol{x}_i, \boldsymbol{w}, \sigma^2) \sim \prod_{i=1}^{N} N(y_i, \sigma^2) \tag{D.31}$$

$$= \prod_{i=1}^{N} (2\pi\sigma^2)^{\frac{1}{2}} \exp\{-\frac{1}{2\sigma^2(t_i - \boldsymbol{w}^T\phi(\boldsymbol{x}))^2}\} \tag{D.32}$$

$$= (2\pi\sigma^2)^{\frac{N}{2}} \exp\{-\frac{1}{2\sigma^2}\|\boldsymbol{t} - \Phi\boldsymbol{w}\|^2\} \tag{D.33}$$

RVM imposes an explicit prior probability distribution on the parameters $\boldsymbol{w}$ to constrian the complexity of the learned model by using a zero-mean Guassian distribution:

$$P(\boldsymbol{w}|\alpha_i) \sim N(0, \alpha_i^{-1}) \tag{D.34}$$

in which α_i is the precision(inverse of the variance) of each w_i, so we have:

$$P(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} N(0, \alpha_i^{-1}) \tag{D.35}$$

which means that there is an individual hyperparameter α_i associated with each weight, modifying the strength of the prior thereon. The posterior probability over all the unknown parameters, given the training data, is given as $P(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t})$. The training is then to find parameters by maximizing this posterior probability. By the marginal probability rule, this can be decomposed into:

$$P(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}) = P(\boldsymbol{w}|\boldsymbol{\alpha}, \sigma^2, \boldsymbol{t}) P(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}) \tag{D.36}$$

If we write $\beta^{-1} = \sigma^2$, the first part:

$$P(\boldsymbol{w}|\boldsymbol{\alpha}, \sigma^2, \boldsymbol{t}) \sim N(\boldsymbol{m}, \boldsymbol{\Sigma}) \tag{D.37}$$

in which (with $A = diag(\alpha)$):

$$\boldsymbol{m} = \beta\Sigma\Phi^T t \tag{D.38}$$

$$\boldsymbol{\Sigma} = (A + \beta\Phi^T\Phi)^{-1} \tag{D.39}$$

The objective of training is now to find hyperparameters α and β which maximize the second part of D.36, which can be decomposed into:

$$P(\boldsymbol{\alpha}, \sigma^2|\boldsymbol{t}) \propto P(\boldsymbol{t}|\boldsymbol{\alpha}, \sigma^2) P(\boldsymbol{\alpha}) P(\sigma^2) \tag{D.40}$$

The first part of which is called *evidence*, and we assume uniform distribution for parameters α and β. Now the problem is to maximize the *evidence*:

$$P(\boldsymbol{t}|\boldsymbol{\alpha},\sigma^2) = P(\boldsymbol{t}|\boldsymbol{\alpha},\beta) = \int P(\boldsymbol{t}|\boldsymbol{w},\beta)P(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \tag{D.41}$$

The first component of the equation:

$$\begin{aligned} P(\boldsymbol{t}|\boldsymbol{w},\beta) &= \prod_{i=1}^{N} N(y,\beta^{-1}) \\ &= (\frac{2\pi}{\beta})^{-\frac{N}{2}} \exp\{-\frac{\beta}{2}\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\|^2\} \end{aligned} \tag{D.42}$$

And the second component (M denotes the dimension of $\boldsymbol{x}$):

$$\begin{aligned} P(\boldsymbol{w}|\boldsymbol{\alpha}) &= \prod_{i=1}^{M} N(0,\alpha_i^{-1}) \\ &= \prod_{i=1}^{M} (2\pi\alpha^{-1})^{-\frac{1}{2}} \exp\{-\frac{1}{2}\alpha_i w^2\} \\ &= (2\pi)^{-\frac{M}{2}} \prod_{i=1}^{M} \alpha_i^{\frac{1}{2}} \exp\{-\frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w}\} \end{aligned} \tag{D.43}$$

Substituting D.43 and D.44 into D.41, we have:

$$\begin{aligned} P(\boldsymbol{t}|\boldsymbol{\alpha},\beta) &= \int (\frac{2\pi}{\beta}^{\frac{N}{2}}) \exp\{-\frac{\beta}{2}\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\|^2\}(2\pi)^{-\frac{M}{2}} \prod_{i=1}^{M} \alpha_i^{\frac{1}{2}} \exp\{-\frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w}\}d\boldsymbol{w} \\ &= (\frac{\beta}{2\pi})^{\frac{N}{2}} (\frac{1}{2\pi})^{\frac{M}{2}} \prod_{i=1}^{M} \alpha_i^{\frac{1}{2}} \int \exp -\{\frac{\beta}{2}\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\|^2 + \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w}\}d\boldsymbol{w} \end{aligned} \tag{D.44}$$

In order to simplify the equation, we define:

$$E(\boldsymbol{w}) = \frac{\beta}{2}\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{w}\|^2 + \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w}$$

Such that it can be written as:

$$P(\boldsymbol{t}|\boldsymbol{\alpha},\beta) = (\frac{\beta}{2\pi})^{\frac{N}{2}} (\frac{1}{2\pi})^{\frac{M}{2}} \prod_{i=1}^{M} \alpha_i^{\frac{1}{2}} \int \exp\{-E(\boldsymbol{w})\}d\boldsymbol{w}$$

And we can expand $E(\boldsymbol{w})$ as:

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{\beta}{2}(\boldsymbol{t}^T\boldsymbol{t} - 2\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}) + \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} \\ &= \frac{1}{2}(\beta\boldsymbol{t}^T\boldsymbol{t} - 2\beta\boldsymbol{t}^T\boldsymbol{\Phi}\boldsymbol{w} + \beta\boldsymbol{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{w}^T A\boldsymbol{w} \end{aligned} \tag{D.45}$$

Substituting into D.39 and using $\Sigma^{-1}\Sigma = \boldsymbol{I}$:

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}(\beta \boldsymbol{t}^T \boldsymbol{t} - 2\beta \boldsymbol{t}^T \boldsymbol{\Phi} \boldsymbol{w} + \boldsymbol{w}^T \Sigma^{-1} \boldsymbol{w} \\ &= \frac{1}{2}(\beta \boldsymbol{t}^T \boldsymbol{t} - 2\beta \boldsymbol{t}^T \boldsymbol{\Phi} \Sigma^{-1} \Sigma \boldsymbol{w} + \boldsymbol{w}^T \Sigma^{-1} \boldsymbol{w} \end{aligned} \tag{D.46}$$

Substituting into D.39:

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}(\beta \boldsymbol{t}^T \boldsymbol{t}) - 2\boldsymbol{m}^T \Sigma^{-1} \boldsymbol{w} + \boldsymbol{w}^T \Sigma^{-1} \boldsymbol{w} + \boldsymbol{m}^T \Sigma^{-1} \boldsymbol{m} - \boldsymbol{m}^T \Sigma^{-1} \boldsymbol{m}) \\ &= E(\boldsymbol{t}) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{m})^T \Sigma^{-1} (\boldsymbol{w} - \boldsymbol{m}) \end{aligned} \tag{D.47}$$

with:

$$E(\boldsymbol{t}) = \frac{1}{2}(\beta \boldsymbol{t}^T \boldsymbol{t} - \boldsymbol{m}^T \Sigma^{-1} \boldsymbol{m}) \tag{D.48}$$

Now D.45 becomes:

$$P(\boldsymbol{t}|\boldsymbol{\alpha}, \beta) = (\frac{\beta}{2\pi})^{\frac{N}{2}} (\frac{1}{2\pi})^{\frac{M}{2}} \int_{i=1}^{M} \alpha_i^{\frac{1}{2}} \exp\{-E\{\boldsymbol{w}\}(2\pi)^{\frac{M}{2}} \|\boldsymbol{\Sigma}\|^{\frac{1}{2}}\}$$

This is the *marginal likelihood* and taking log, we obtain the *log marginal likelihood*:

$$\ln P(\boldsymbol{t}|\boldsymbol{\alpha}, \beta) = \frac{N}{2} \ln \beta - E(\boldsymbol{t}) - \frac{1}{2} \ln \|\Sigma\| - \frac{N}{2} \ln(2\pi) + \frac{1}{2} \sum_{i=1}^{M} \ln \alpha_i \tag{D.49}$$

We chose to maximize this equation with respect to α and β. The process is called *evidence approximation*.

D.2.2 Evidence Approximation

To maximize the log marginal likelihood, we start by taking derivative of D.49 w.r.t. α_i and setting it to zero:

$$\begin{aligned} \frac{d}{d\alpha_i} \ln P(\boldsymbol{t}|\boldsymbol{\alpha}, \beta) &= \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii} - \frac{1}{2} m_{ii}^2 = 0 \\ &\Rightarrow \quad \alpha_i = \frac{1 - \alpha_{ii}\Sigma_{ii}}{m_i^2} \end{aligned} \tag{D.50}$$

substituting in $\gamma_i = 1 - \alpha_i \Sigma_{ii}$ the recursive definition for the α_i which maximize D.49 can be expressed as:

$$\alpha_i = \frac{\gamma_i}{m_i^2}$$

And the derivative of D.49 w.r.t. β is given as:

$$\frac{d}{d\beta} \ln P(\boldsymbol{t}|\boldsymbol{\alpha}, \beta) = \frac{1}{2}(\frac{N}{\alpha} - \|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{m}\|^2 - Tr[\Sigma \Phi^T \Phi]) = 0 \tag{D.51}$$

In order to solve this, we shall firstly simplify the argument of the trace operator $Tr(\bullet)$:

$$\begin{aligned}\Sigma\Phi^T\Phi &= \Sigma\Phi^T\Phi+\beta^{-1}\Sigma A-\beta^{-1}\Sigma A \\ &= \Sigma(\Phi^T\Phi\beta+A)\beta^{-1}-\beta^{-1}\Sigma A \\ &= (A+\beta\Phi^T\Phi)^{-1}(\beta\Phi^T\Phi+A)\beta^{-1}-\beta^{-1}\Sigma A \\ &= (I-A\Sigma)\beta^{-1}\end{aligned} \tag{D.52}$$

substituting back to D.51:

$$\begin{aligned}&\frac{1}{2}(\frac{N}{2}-\|\boldsymbol{t}-\boldsymbol{\Phi m}\|^2)-Tr[\frac{\boldsymbol{I}-A\Sigma}{\beta}]=0 \\ \Rightarrow\ &\frac{N}{\beta}-Tr[\frac{\boldsymbol{I}-A\Sigma}{\beta}]=\|\boldsymbol{t}-\boldsymbol{\Phi m}\|^2 \\ \Rightarrow\ &\frac{1}{\beta}=\frac{\|\boldsymbol{t}-\boldsymbol{\Phi m}\|^2}{N-Tr[\boldsymbol{I}-A\Sigma]} \\ \Rightarrow\ &\beta=\frac{N-\Sigma_i\gamma_i}{\|\boldsymbol{t}-\boldsymbol{\Phi m}\|^2}\end{aligned} \tag{D.53}$$

The α_i and β, which maximize the marginal likelihood, are then found iteratively by setting the paramters to initial values, finding values for $\boldsymbol{m}$ and $\boldsymbol{\Sigma}$, and then calculating new values for the parameters until the convergence criteria is met. With the values of α_i and β, we can then use the model to compute the predictive probability for a new input $\boldsymbol{x}^*$:

$$\begin{aligned}P(t|\boldsymbol{x}^*,\boldsymbol{\alpha},\beta) &= \int P(t|\boldsymbol{w},\beta)P(\boldsymbol{w}|\boldsymbol{\alpha},\beta)d\boldsymbol{w} \\ &= N(\boldsymbol{m}^T\boldsymbol{\phi}(\boldsymbol{x}^*),\delta^2(\boldsymbol{x}^*))\end{aligned} \tag{D.54}$$

While carrying out the procedure, many of the variables α_i will tend to be ∞, which will force w_i into zero:

$$\begin{aligned}\lim_{\alpha_i\to\infty}\Sigma &= \lim_{\alpha_i\to\infty}(A+\beta\Phi^T\Phi)^{-1}=0 \\ \Rightarrow\lim_{\alpha_i\to\infty}\boldsymbol{m} &= \lim_{\alpha_i\to\infty}\beta\Sigma\Phi^T\boldsymbol{t}=0\end{aligned} \tag{D.55}$$

This means that each w_i such that α_i relate to will be distributed $\alpha_i\sim N(0,0)$ and will be equal to zero, producing a *sparse* model. The $\boldsymbol{x}_i$ corresponding to the remaining non-zero weights after pruning are called relevance vectors and are analogous to the support vectors of an SVM.

Bibliography

[Agarwal 04] Ankur Agarwal & Bill Triggs. *3D human pose from silhouettes by relevance vector regression*. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–882. IEEE, 2004. 65, 163

[Aggarwal 07] P. Aggarwal, A. Dutta & B. Bhattacharya. *Cooperation between a 4 DOF robotic hand and a human for carrying an object together*. In SICE, 2007 Annual Conference, pages 2354–2359, 2007. 26, 145

[A.Jensen 01] A.Jensen & A.la Cour-Harbo. Ripples in mathematics. Springer, 2001. 55, 58

[Argall 09] Brenna D. Argall, Sonia Chernova, Manuela Veloso & Brett Browning. *A survey of robot learning from demonstration*. Robotics and Autonomous Systems, vol. 57, no. 5, pages 469 – 483, 2009. 2, 24, 99, 144

[Berchtold 94] S. Berchtold & B. Glavina. *A scalable optimizer for automatically generated manipulator motions*. In IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., 1994. 18

[Biagiotti 08] L. Biagiotti & C. Melchiorri. Trajectory planning for automatic machines and robots. Springer, 2008. 21, 168

[Birbaumer 00] Niels Birbaumer, Andrea Kubler, Nimr Ghanayim, Thilo Hinterberger, Jouri Perelmouter, Jochen Kaiser, Iver Iversen, Boris Kotchoubey, Nicola Neumann & Herta Flor. *The thought translation device (TTD) for completely paralyzed patients*. Rehabilitation Engineering, IEEE Transactions on, vol. 8, no. 2, pages 190–193, 2000. 59, 158

[Bishop 06] Christopher M Bishop *et al*. Pattern recognition and machine learning, volume 4. springer New York, 2006. 28, 65, 163

[Bounab 08] B. Bounab, D. Sidobre & A. Zaatri. *Central axis approach for computing n-finger force-closure grasps*. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 1169–1174, May 2008. 22, 86, 171

[Brady 82] M. Brady, J. Hollerbach, T. Johnson & T. Lozano-Perez. Robot motion, planning and control. The MIT Press, Cambridge, Massachusetts, 1982. 21

[Broquère 08a] X. Broquère, D. Sidobre & I. Herrera-Aguilar. *Soft motion trajectory planner for service manipulator robot*. In IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., 2008. 74, 75, 76, 77, 81

[Broquere 08b] Xavier Broquere, Daniel Sidobre & I. Herrera-Aguilar. *Soft Motion Trajectory Planner for Service Manipulator Robot*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. 22

[Broquere 08c] Xavier Broquere, Daniel Sidobre & Ignacio Herrera-Aguilar. *Soft motion trajectory planner for service manipulator robot*. Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 2808–2813, Sept. 2008. 22, 72, 92, 168, 175

[Broquère 10] X. Broquère & D. Sidobre. *From motion planning to trajectory control with bounded jerk for service manipulator robots*. In IEEE Int. Conf. Robot. And Autom., 2010. 72, 75, 77, 82, 168

[Broquère 11] X. Broquère. *Planification de trajectoire pour la manipulation d'objets et l'interaction Homme-robot*. PhD thesis, LAAS-CNRS and Université de Toulouse, Paul Sabatier, 2011. 72, 74, 75, 82

[Burges 98] Christopher JC Burges. *A tutorial on support vector machines for pattern recognition*. Data mining and knowledge discovery, vol. 2, no. 2, pages 121–167, 1998. 64, 162

[Buttazzo 94] G Buttazzo, B. Allotta & F. Fanizza. *Mousebuster: A robot for real-time catching*. IEEE Control Systems Magazine, vol. 14(1), 1994. 24, 143

[Calinon 04] S. Calinon & A. Billard. *Stochastic gesture production and recognition model for a humanoid robot*. In Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, pages 2769 – 2774 vol.3, sept.-2 oct. 2004. 24, 144

[C.H.An 88] C.H.An, C.G.Atkeson & J.M.Hollerbach. *Model-based Control of a Robot Manipulator*. The MIT Press series in artificial intelligence, 1988. 25, 145

[Chang 11] Chih-Chung Chang & Chih-Jen Lin. *LIBSVM: a library for support vector machines*. ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, page 27, 2011. 68, 117, 166

[Chaumentte 06] F. Chaumentte & S.A. Hutchinson. *Visual servo control. Part I: Basic approaches*. IEEE Robotics and Automation Magazine, vol. 4(13), December 2006. 24, 144

[Chaumentte 07] F. Chaumentte & S.A. Hutchinson. *Visual servo control. Part II: Advanced approaches*. IEEE Robotics and Automation Magazine, vol. 1(14), March 2007. 24, 144

[Chui 98] C.K. Chui & G. Chen. Kalman filtering. Springer, 1998. 105

[Coifman 92] Ronald R. Coifman & M. Victor Wickerhauser. *Entropy-based algorithms for best basis selection*. Information Theory, IEEE Transactions on, vol. 38, no. 2, pages 713–718, 1992. 56

[Cortes 95] Corinna Cortes & Vladimir Vapnik. *Support-vector networks*. Machine learning, vol. 20, no. 3, pages 273–297, 1995. 117

[Cortés 04] J Cortés & T. Siméon. *Sampling-based motion planning under kinematic loop-closure constraints*. In International Workshop on Algorithmic Foundations of Robotics, Utrecht, Netherland, 2004. 81

[Daubechies 98] Ingrid Daubechies & Wim Sweldens. *Factoring wavelet transforms into lifting steps*. Journal of Fourier analysis and applications, vol. 4, no. 3, pages 247–269, 1998. 57

[De Luca 08] A. De Luca & L. Ferrajoli. *Exploiting robot redundancy in collision detection and reaction*. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 3299 –3305, sept. 2008. 87, 172

[der Merwe 01] R.van der Merwe & E.Wan. *Efficient Derivative-Free Kalman Filters for Online Learning*. In Proceedings of the 9th European Symposium on Artificial Neural Networks(ESANN), pages 205–210, 2001. 45

[Devos 12] C Devos, A Alix, V Chedru-Legros, C Hecquard & G Saint-Lorant. *Overall equipment effectiveness: a production work tool applied to a parenteral nutrition robot in a University Hospital*. European Journal of Hospital Pharmacy: Science and Practice, vol. 19, no. 2, pages 143–143, 2012. 2

[Dornaika 98] Fadi Dornaika & Radu Horaud. *Simultaneous robot-world and hand-eye calibration*. Robotics and Automation, IEEE Transactions on, vol. 14, no. 4, pages 617–622, 1998. 40, 152

[(Ed.) 12] Bruno Siciliano (Ed.). Advanced bimanual manipulation: Results from the dexmart project. Springer, 2012. 83

[Elmenreich 07] Wilfried Elmenreich. *A review on system architectures for sensor fusion applications*. In Software Technologies for Embedded and Ubiquitous Systems, pages 547–559. Springer, 2007. 25, 145

[Englehart 99] Kevin Englehart, Bernard Hudgins, Philip A Parker & Maryhelen Stevenson. *Classification of the myoelectric signal using time-frequency based representations*. Medical engineering & physics, vol. 21, no. 6, pages 431–438, 1999. 28, 146

[Farrokh 11] Janabi-Sharifi Farrokh, Deng Lingfeng & J.Wilson William. *Comparison of Basic Visual Servoing Methods*. IEEE/ASME Transactions on Mechatronics, vol. 16, no. 5, October 2011. 24, 144

[Fatourechi 04] M Fatourechi, SG Mason, GE Birch & RK Ward. *A wavelet-based approach for the extraction of event related potentials from EEG*. In Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on, volume 2, pages ii–737. IEEE, 2004. 59, 158

[Fleury 97] S. Fleury, M. Herrb & R. Chatila. *Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture*. In IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., 1997. 12, 142

[Fong 03] T. Fong, I. Nourbakhsh & K. Dautenhahn. *A survey of socially interactive robots*. Robotics and autonomous systems, vol. 42, no. 3-4, pages 143–166, 2003. 2

[Garcia 05] J.Gamew Garcia, A.Robertsson, J.Gomew Ortega & R.Johansson. *Force and Acceleration Sensor Fusion for Compliant Robot Motion Contrl*. In Proceedings of the 2005 IEEE Internationla Conference on Robotics and Automation, 2005. 26

[Gosselin 93] G Gosselin, J. Cote & D Laurendeau. *Inverse kinematic functions for approach and catching operations*. IEEE Trans. Systems, Man, and Cybernetics, vol. 23(3), 1993. 24, 144

[Gu 11] Quanquan Gu, Zhenhui Li & Jiawei Han. *Linear discriminant dimensionality reduction*. Machine Learning and Knowledge Discovery in Databases, pages 549–564, 2011. 62, 161

[Hall 63] Edward T Hall. *A system for the notation of proxemic behavior1*. American anthropologist, vol. 65, no. 5, pages 1003–1026, 1963. 15

[Hall 04] David David Lee Hall & Sonya Anne Hall McMullen. Mathematical techniques in multisensor data fusion. Artech House, 2004. 25

[Han 01] L. Han & N.M. Amato. *A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems*. In B. R. Donald, K. M. Lynch & D. Rus, editeurs, International Workshop on Algorithmic Foundations of Robotics, pages 233–246. A K Peters, Wellesley, MA, 2001. 81

[Haschke 08] R. Haschke, E. Weitnauer & H. Ritter. *On-Line Planning of Time-Optimal, Jerk-Limited Trajectories*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008, pages 3248–3253, 2008. 22

[Jaillet 10] L. Jaillet, J. Cortés & T. Siméon. *Sampling-based path planning on configuration-space costmaps*. IEEE Transactions on Robotics, 2010. 18

[Julier 96] Simon Julier & Jeffrey K.Uhlmann. *A General Method for Approximating Nonlinear Transformations of Probability Distributions*. Rapport technique, 1996. 44, 105

[Kajikawa 95] S. Kajikawa, T. Okino, K. Ohba & H. Inooka. *Motion planning for hand-over between human and robot*. In 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings, volume 1, 1995. 27, 146

[Kavraki 08] L.E. Kavraki & S.M. LaValle. Motion Planning In Springer Handbook of Robotics, by B. Siciliano and O. Khatib. 2008. 18

[Khalil 99] W. Khalil & E Dombre. Modélisation identification et commande des robots, volume 56. Hermes, 1999. 21

[Khan 12] M Tahir Khan, Bessie Chan, Afzal Khan, Zia Haq & Javaid Iqbal. *Optimized Dynamic Task Allocation and Priority Assignments in an Immunized Autonomous Multi-Robot Search and Rescue Operation*. ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY, vol. 15, no. 2, pages 129–145, 2012. 2

[Khansari-Zadeh 11] SMohammad Khansari-Zadeh & Aude Billard. *Learning stable non-linear dynamical systems with gaussian mixture models*. Robotics, IEEE Transactions on, vol. 27, no. 5, pages 943–957, 2011. 27, 99, 146

[Kim 02] I. Kim, N. Nakazawa & H. Inooka. *Control of a robot hand emulating human's hand-over motion*. Mechatronics, vol. 12, no. 1, pages 55–69, 2002. 26, 145

[Koenig 10] Nathan Koenig, Leila Takayama & Maja Matarić. *Communication and knowledge sharing in human–robot interaction and learning from demonstration*. Neural Networks, vol. 23, no. 8, pages 1104–1112, 2010. 2

[Kröger 06] T. Kröger, A. Tomiczek & F.M. Wahl. *Towards on-line trajectory computation*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China. Citeseer, 2006. 22

[Kröger 10a] T. Kröger. On-line trajectory generation in robotic systems, volume 58 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Heidelberg, Germany, first edition, jan 2010. 168

[Kroger 10b] Torsten Kroger & Friedrich M Wahl. *Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events*. Robotics, IEEE Transactions on, vol. 26, no. 1, pages 94–111, 2010. 21

[Kröger 11] Torsten Kröger, Bernd Finkemeyer & FriedrichM. Wahl. Manipulation primitives: A universal interface between sensor-based motion control and robot programming, volume 67 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, 2011. 86, 170

[Kröger 12] T. Kröger & Jose Padial. *Simple and Robust Visual Servo Control of Robot Arms Using an On-Line Trajectory Generator*. 2012 IEEE International Conference on Robotics and Automation, 2012. 24, 144

[Kubus 08a] Daniel Kubus, Torsten Kroger & Friedrich M.Wahl. *On-Line Estimation of Inertial Parameters Using a Recursive Total Least-Squares Approach*. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. 25, 145

[Kubus 08b] Daniel Kubus, Kroger Torsten & M.Wahl Friedrich. *Improving Force Control Performance by Computational Elimination of Non-Contact Forces/Torques*. In IEEE International Conference on Robotics and Automation, 2008. 34, 149

[Larsen 99] E. Larsen, S. Gottschalk, M.C. Lin & D. Manocha. *Fast proximity queries with swept sphere volumes*. 1999. 87, 172

[LaValle 01] S. M. LaValle & J. Kuffner. *Rapidly-exploring random trees: Progress and prospects*. In Workshop on the Algorithmic Foundations of Robotics, 2001. 18

[LaValle 06] S.M. LaValle. Planning algorithms. Cambridge Univ Pr, 2006. 18

[Learned 95] Rachel E Learned & Alan S Willsky. *A wavelet packet approach to transient signal classification*. Applied and Computational Harmonic Analysis, vol. 2, no. 3, pages 265–278, 1995. 28, 56, 146

[Lemaignan 12] Séverin Lemaignan. *Grounding the interaction: knowledge management for interactive robots*. KI-Künstliche Intelligenz, pages 1–3, 2012. 2, 137

[Liu 02] Steven Liu. *An on-line reference-trajectory generator for smooth motion of Impulse-Controlled Industrial Manipulators*. In 7th International Workshop on Advanced Motion Control, pages 365–370, 2002. 22

[Llinas 98] James Llinas & David L Hall. *An introduction to multi-sensor data fusion*. In Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on, volume 6, pages 537–540. IEEE, 1998. 25, 144

[Luo 12] Ren C Luo & Chih-Chia Chang. *Multisensor fusion and integration: A review on approaches and its applications in mechatronics*. Industrial Informatics, IEEE Transactions on, vol. 8, no. 1, pages 49–60, 2012. 25

[Mainprice 10] Jim Mainprice, E Akin Sisbot, Thierry Siméon & Rachid Alami. *Planning safe and legible hand-over motions for human-robot interaction*. In IARP Workshop on Technical Challenges for Dependable Robots in Human Environments, volume 2, page 7, 2010. 13, 143

[Mainprice 11] J. Mainprice, E.A. Sisbot, L. Jaillet, J Cortés, T. Siméon & R. Alami. *Planning Human-aware motions using a sampling-based costmap planner*. In IEEE Int. Conf. Robot. And Autom., 2011. 18

[Mallat 08] Stephane Mallat. A wavelet tour of signal processing: The sparse way. 2008. 27, 55, 57, 60, 146, 159

[Mianji 11] Fereidoun A Mianji & Ye Zhang. *Robust hyperspectral classification using relevance vector machine*. Geoscience and Remote Sensing, IEEE Transactions on, vol. 49, no. 6, pages 2100–2112, 2011. 65, 163

[M.Niebergall 01] M.Niebergall & H.Hahn. *Development of a measurement robot for identifying all inertia parameters of a rigid body in a single experiment*. In IEEE Transactions on Control Systems Technology, 2001. 25, 145

[Nagata 98] K. Nagata, Y. Oosaki, M. Kakikura & H. Tsukune. *Delivery by hand between human and robot based on fingertipforce-torque information*. In 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998. Proceedings., volume 2, 1998. 26, 145

[Nakasawa 99] N. Nakasawa, Il-Hwan Kim, H. Inooka & R. Ikeura. *Force control of a robot hand Emulating human's grasping Motion*. In Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on, volume 6, pages 774–779 vol.6, 1999. 26, 145

[Nakazawa 01] N. Nakazawa, I. Kim, H. Inooka & R. Ikeura. *Force control of a robot gripper based on human grasping schemes*. Control Engineering Practice, vol. 9, no. 7, pages 735–742, 2001. 26, 145

[Ott 06a] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke,

M. Suppa, T. Wimbock, F. Zacharias & G. Hirzinger. *A Humanoid Two-Arm System for Dexterous Manipulation*. In IEEE/RAS International Conference on Humanoid Robots, pages 276–283, December 2006. 10, 141

[Ott 06b] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer *et al*. *A Humanoid Two-Arm System for Dexterous Manipulation*. In Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pages 276–283, 2006. 2

[Rehman 12] Abdul Rehman, Yang Gao, Jiheng Wang & Zhou Wang. *Image classification based on complex wavelet structural similarity*. Signal Processing: Image Communication, 2012. 28

[Romano 11] Joseph M Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta & Katherine J Kuchenbecker. *Human-inspired robotic grasp control with tactile sensing*. Robotics, IEEE Transactions on, vol. 27, no. 6, pages 1067–1079, 2011. 27, 145

[Sato 12] Munehiko Sato, Ivan Poupyrev & Chris Harrison. *Touche: enhancing touch interaction on humans, screens, liquids, and everyday objects*. In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, pages 483–492. ACM, 2012. 28

[Saut 12] Jean-Philippe Saut & Daniel Sidobre. *Efficient Models for Grasp Planning with a Multi-Fingered Hand*. Robotics and Autonomous Systems, vol. 60, March 2012. 22, 86, 171

[Shoemake 85] Ken Shoemake. *Animating rotation with quaternion curves*. ACM SIGGRAPH computer graphics, vol. 19, no. 3, pages 245–254, 1985. 20

[Siciliano 08] B. Siciliano & O. Khatib. Springer Handbook of Robotics. 2008. 17

[Sidobre 09] Daniel Sidobre, Matthieu Herrb, Jéröme Manhes, Xavier Broquère, Rachid Alami, Raja Chatila, juan Cortes, Felix Ingrand, Jean-Philipp Saut & Thierry Siméon. *Models of information exchanged by two humans exchanging an object for several strategies of handling*. Rapport technique DEXMART report D1.2, LAAS-CNRS, September 2009. 28

[Sidobre 12] D. Sidobre, X. Broquère, J. Mainprice, E. Burattini, A. Finzi, S. Rossi & M. Staffa. *Human–Robot Interaction*. Advanced Bimanual Manipulation, pages 123–172, 2012. 72, 84, 168

[Sisbot 07a] E. A. Sisbot, L. F. Marin-Urias, R. Alami & T. Siméon. *Spatial Reasoning for Human-Robot Interaction*. In IEEE/RSJ Int. Conf. on Intel. Rob. And Sys., San Diego, CA, USA, November 2007. 15

[Sisbot 07b] E. Akin Sisbot, Luis F. Marin Urias, Rachid Alami & Thierry Siméon. *Spatial Reasoning for Human-Robot Interaction*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, San Diego, CA, USA, November 2007. 12, 143

[Sisbot 11] E.Akin Sisbot, Raqual Ros & Rachid Alami. *Situation Assessment for Human-Robot Interactive Object Manipulation*. 20th IEEE International Symposium on Robot and Human Interactive Communication, July-August 2011. 83, 168

[Smith 06] Duncan Smith & Sameer Singh. *Approaches to multisensor data fusion in target tracking: A survey*. Knowledge and Data Engineering, IEEE Transactions on, vol. 18, no. 12, pages 1696–1710, 2006. 25, 145

[Swevers 97] Jan Swevers, Chris Ganseman, Dilek Bilgin Tukel, Joris De Schutter & Hendrik Van Brussel. *Optimal Robot Excitation and Identification*. In IEEE Transactions on Robotics and Automation, 1997. 25, 145

[Takubo 02] T. Takubo, H. Arai, Y. Hayashibara & K. Tanie. *Human-robot cooperative manipulation using a virtual nonholonomic constraint*. International Journal of Robotics Research, vol. 21, no. 5, pages 541–553, 2002. 26, 145

[Tipping 01] Michael E Tipping. *Sparse Bayesian learning and the relevance vector machine*. The Journal of Machine Learning Research, vol. 1, pages 211–244, 2001. 28, 64, 65, 162, 163

[Uchiyama 85] M. Uchiyama, M. Yokota & K. Hakomori. *Kalman Filtering the 6-Axis Robot Wrist Force Sensor Signal*. In Proceedings of '85 International Conference on Advanced Robotics, 1985. 26

[Vakanski 12a] A. Vakanski, I. Mantegh, A. Irish & F. Janabi-Sharifi. *Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping*. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 42, no. 4, pages 1039 –1052, aug. 2012. 24, 144

[Vakanski 12b] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish & Farrokh Janabi-Sharifi. *Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping*. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 42, no. 4, pages 1039–1052, 2012. 27, 99, 146

[van der Merwe 04] Rudolph van der Merwe & A.Wan Eric. *Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion -Application to Integrated*

Navigation-. Rapport technique, American Institute of Aeronautics and Astronautics, 2004. 44

[Vapnik 99] Vladimir Vapnik. The nature of statistical learning theory. springer, 1999. 64, 162

[Widodo 09] Achmad Widodo, Eric Y Kim, Jong-Duk Son, Bo-Suk Yang, Andy CC Tan, Dong-Sik Gu, Byeong-Keun Choi & Joseph Mathew. *Fault diagnosis of low speed bearing based on relevance vector machine and support vector machine*. Expert Systems with Applications, vol. 36, no. 3, pages 7252–7261, 2009. 65, 163

[Xue 03] Jian-Zhong Xue, Hui Zhang, Chong-Xun Zheng & Xiang-Guo Yan. *Wavelet packet transform for feature extraction of EEG during mental tasks*. In Machine Learning and Cybernetics, 2003 International Conference on, volume 1, pages 360–363. IEEE, 2003. 59, 158

[Yang 04] Jingzhou Yang, R Timothy Marler, HyungJoo Kim, Jasbir Arora & Karim Abdel-Malek. *Multi-objective optimization for upper body posture prediction*. In 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, volume 30, 2004. 15

[Yousuf 12] Mohammad Abu Yousuf, Yoshinori Kobayashi, Yoshinori Kuno, Akiko Yamazaki & Keiichi Yamazaki. *Development of a Mobile Museum Guide Robot That Can Configure Spatial Formation with Visitors*. In Intelligent Computing Technology, pages 423–432. Springer, 2012. 2

E

Résumé en français

Fusion de données et commande réactive pour l'Interaction humain-robot et la manipulation d'objets

E.1 Introduction

Les tâches exécutées par les robots industriels sont le plus souvent prédéfinies et réalisées dans un environnement statique et structuré dont les humains sont exclus. Les chercheurs en robotique de service ont commencé à construire des robots qui évoluent et travaillent parmi les humains, ce qui n'a longtemps été qu'un rêve de science-fiction. Les manipulateurs mobiles ne sont pas encore disponibles à l'achat, mais ils sont déjà une réalité dans les laboratoires du monde entier. Même s'ils ne sont encore utilisés que comme plates-formes de recherche, les résultats prometteurs publiés chaque année et l'enthousiasme qu'ils suscitent font que les robots de services ne peuvent plus être considéré comme des rêves d'enfants.

E.1.1 Motivation and Contribution

Les robots de services travaillent dans un environnement dynamique et non structuré, aussi est-il impossible de prévoir toutes les actions qu'un robot de service devra réaliser. L'autonomie nécessaire pour réagir aux événements soulève de nombreuses difficultés. Tout d'abord, la prise en compte des codes sociaux et éthiques entre les personnes pour choisir son action n'est pas une chose aisée. Ce niveau d'interaction est dite cognitive ([Lemaignan 12]). Deuxièmement, pour travailler dans un environnement évolutif, un robot doit être réactif.

Par exemple, il doit être capable d'attraper un objet en mouvement ou de réagir à des actions imprévues des personnes qui l'entourent et donc de les modéliser pour les interpréter. Le contrôle de bas niveau qui doit impérativement garantir la sécurité du système et des personnes doit être développé en parallèle.

Par analogie, l'objectif de ce travail pourrait correspondre à faire en sorte qu'une personne saisisse un objet inconnu et le donne à une seconde personne dans un environnement sombre. Très souvent les systèmes de vision sont instables et les robots ne voient pas les objets correctement, un peu comme nous dans l'obscurité. Comme l'objet est inconnu, le robot ne connaît pas sa masse qui est un paramètre important pour les lois de commande. Pour que l'échange soit naturel, intuitif et fiable, les deux personnes doivent interagir de manière réactive. Le donneur doit prévoir où et comment le receveur va saisir l'objet puis lâcher l'objet lorsque il est sûr que son partenaire le tient fermement. Les travaux présentés dans ce mémoire visent à résoudre ces problèmes et s'appuient sur des outils de planification de mouvement développés par d'autres collègues de notre équipe de recherche. Nous présentons ce travail en trois parties : la fusion des données de force, la classification pour la détection d'évènements et le contrôle de trajectoire. Un exemple de scénario où le robot donne un objet à un humain est présenté sur les figures E.1 et E.2. Le robot tend un objet qu'il a préalablement saisi dans une position permettant une deuxième saisie par la personne située à droite. Lorsque cette personne aura attrapée l'objet, le robot devra le détecter et relâcher l'objet. Les relations entre les différents éléments logiciels sont présentés sur la figure E.3.

Figure E.1: Une tâche d'échange d'objet typique : le robot donne l'objet à un humain.

E.1.1.1 Fusion et estimations de force

Travaillant dans des environnements non structurés, le robot ne possède pas toujours un modèle de l'environnement. Un premier défi est de construire un modèle géométrique de

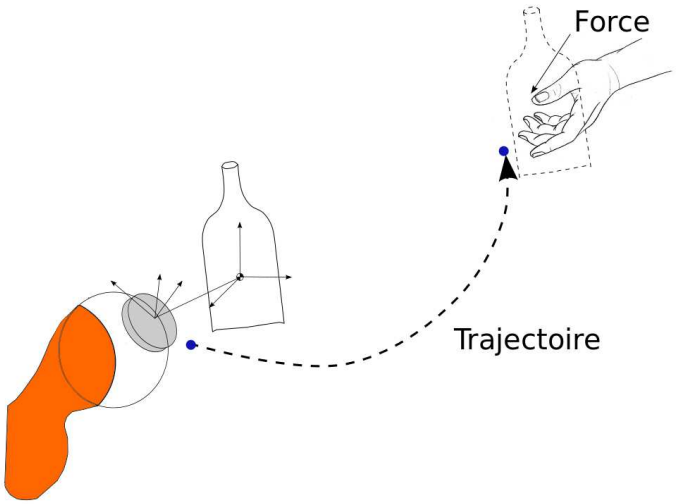


Figure E.2: Lorsque le robot exécute la trajectoire, il doit estimer la dynamique de l'objet manipulé, calculer la force de contact entre l'objet et l'humain, et contrôler le mouvement pour atteindre la position relative définie dans un repère local lié à la main du partenaire.

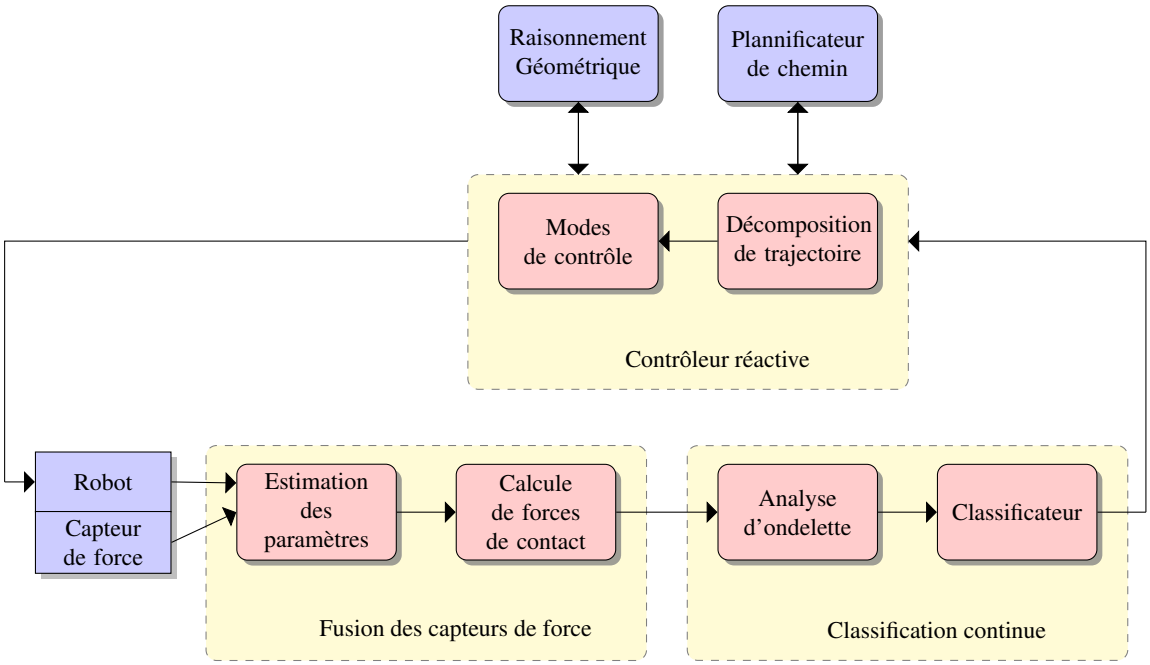


Figure E.3: Relation entre les différentes contributions de cette thèse: estimation de force, classification et contrôle réactif de trajectoire.

l'environnement à partir de la vision et de l'actualiser en temps réel. Pour cela nous utilisons des résultats de travaux des collègues du *LAAS*. En ce qui concerne plus particulièrement une tâche d'échange d'objet, le robot ne connait pas non plus les paramètres dynamiques de l'objet, il doit donc les estimer. C'est ce que nous faisons lorsque nous observons un objet que nous ne connaissons pas, nous pouvons le saisir et le bouger pour voir s'il est lourds,

si sa masse est répartie de manière homogène ou s'il contient du liquide à l'intérieur. Pour un bras manipulateurs, ces informations sont plus importantes encore car les contrôleurs dynamiques ont besoins de connaître ces paramètres dynamiques pour être efficaces. Les avantages de l'estimation en ligne de ces paramètres sont nombreux et sont présentés un peu plus loin. La précision géométrique du suivi de l'objet manipulé et des éléments du corps de l'humain avec lequel le robot interagit est cruciale pour le succès et la sécurité de la tâche de manipulation. L'un des objectifs de cette thèse sera de montrer que la fusion des données provenant de plusieurs types de capteurs peut être bénéfique pour la perception.

E.1.1.2 Apprendre pour manipuler

Dans ce travail, nous présentons aussi nos efforts pour construire un objet destiné à apprendre au robot comment échanger un objet avec un partenaire humain. L'idée conductrice est que pour une tâche où le robot donne un objet à un partenaire humain, le robot doit être capable de lâcher l'objet correctement lorsque il a été saisi par le partenaire. Dans ce but, nous proposons d'enregistrer des échanges entre humains, puis d'utiliser l'analyse par ondelette du signal de force couplé avec l'utilisation d'une machine à vecteur de pertinence (RVM pour Relevance Vector Machine) pour construire un classificateur robuste. Ce modèle peut ensuite être transféré au robot pour réaliser des échanges d'objets entre robot et humains. Il peut aussi être utilisé pour détecter d'autres évènements comme des collisions qui peuvent être exploitées par le contrôleur. L'apprentissage du mouvement et de la loi de contrôle en force peuvent aussi être envisagé, mais ils ne sont ni étudié en profondeur ni expérimenté.

E.1.1.3 Contrôle de trajectoire basé sur les capteurs

Un robot qui a la capacité de construire un modèle dynamique de l'objet manipulé, de suivre le mouvement des objets et des différentes parties du corps de l'humain, et de détecter des évènements particuliers doit pouvoir accomplir la tâche tout en réagissant aux mouvements et aux évènements. Basé sur des travaux précédents, nous proposons un contrôleur de trajectoire pour réaliser des manipulations réactives. Le contrôleur intègre les informations provenant de plusieurs sources et utilise un générateur de trajectoire en ligne comme élément central des algorithmes de suivi de cible mobile, de saisi d'un objet mobile et d'évitement d'obstacle. Nous introduisons et définissons le concept de primitive de contrôle afin de décomposer les tâches de manipulation en une suite de primitives de contrôle. Ces primitives de contrôle nous permettent de définir les stratégies de contrôle de bas-niveau comme des contrôles de position/vitesse ou de force.

E.1.2 État de l'Art

Dans cette partie nous donnons une courte présentation du contexte et faisons une revue de la littérature associée.

E.1.2.1 Mobile Manipulators

La manipulation mobile s'est développée ces dernières années comme plateforme de recherche sur les interactions entre humains et robots. Par exemple, *Justin* [Ott 06a] est un robot de *l'Institut de Robotique et de Mécatronique* du centre aérospatial allemand *(DLR)* et *Rosie* de l'université technique de Munich *(TUM)* sont composé de deux bras *Kuka LWR-IV*. Le robot *Jido* développé au *LAAS-CNRS* (Figure E.4) est composé d'un base mobile *Neobotix MP-L655*, de deux bras *Kuka LWR-IV*, et d'une main à quatre doigts *SAH de Schunk*. Les bras de ce robot sont équipés de capteur de couple intégré à chaque articulation. La figure E.5 présente les composants essentiels du robot et notamment une paire de caméra stéréo et un capteur de profondeur *Kinect* de *Microsoft* pour suivre le mouvement des humains.

Figure E.4: Jido, le robot manipulateur mobile développé au *LAAS-CNRS*

Le deuxième manipulateur mobile du *LAAS* est un *PR2 (Personal Robot 2)* de *Willow Garage*[1]. C'est un robot de recherche et un incubateur de technologie dédié au développement de matériel et de logiciel libre pour les applications de robotique personnelle. Son architecture logicielle ouverte est basée sur le logiciel médiateur *ROS (Robot Operating System)*[2]. Les modules logiciels de type *GenoM*[3] sont compatibles avec ROS, ce qui permet d'utiliser les mêmes développements sur Jido et le *PR2*. Bien que doté sept degrés de liberté, les bras du *PR2* ne sont pas équipé de capteur de couple au niveau des articulations, pas plus que de capteur de force et couple au niveau du poignet.

E.1.2.2 Architecture logicielle pour l'interaction humain-robot

Pour interagir avec un humain le robot doit exécuter plusieurs tâches en parallèle et traiter des informations provenant de sources diverses tout en accomplissant des tâches à plusieurs

[1] http://www.willowgarage.com/pages/pr2/overview
[2] http://www.ros.org
[3] http://www.openrobots.org/wiki/genom

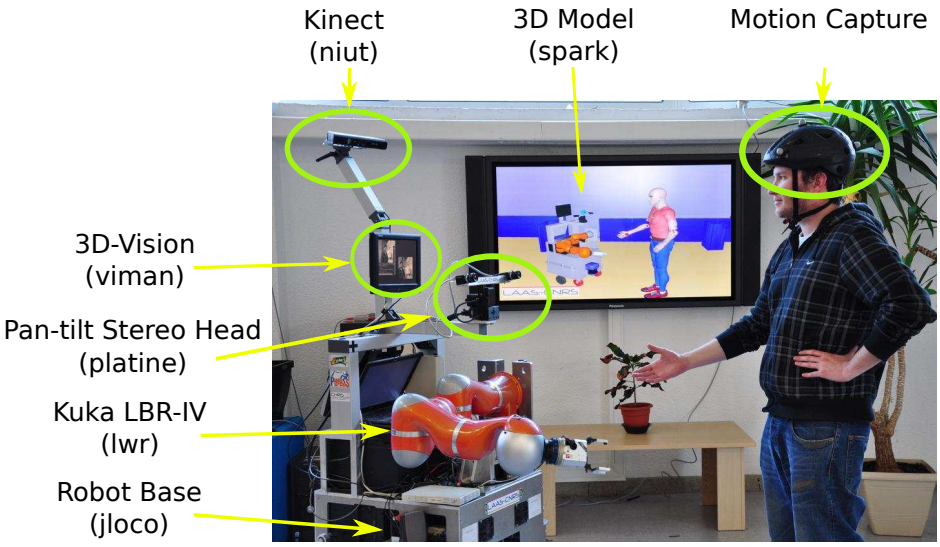


Figure E.5: Quelques éléments de *Jido*. le capteur Kinect assure le suivi du mouvement des humains et une paire de caméra stéréo est montée sur une platine orientable. Le modèle de la scène construit par le robot est affiché sur le grand écran arrière.

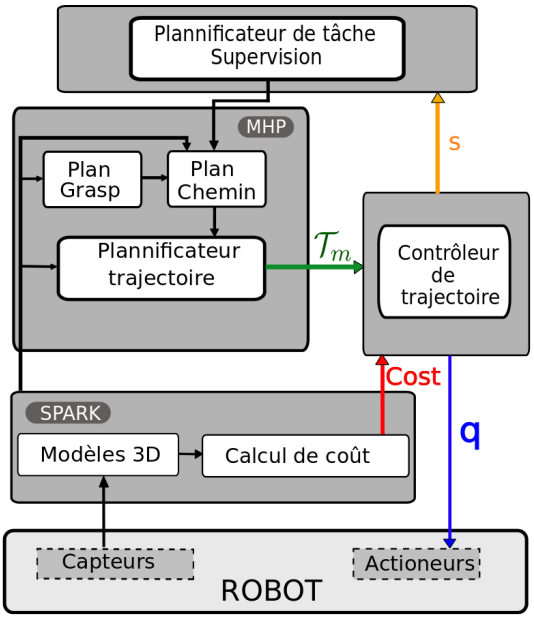


Figure E.6: Architecture logicielle pour la manipulation interactive. $\mathcal{T}_m$ est la trajectoire initiale calculée par MHP. Le contrôleur de trajectoire tient aussi compte des coûts issus de SPARK. Le contrôleur envoie les consignes de position des articulations (q sur la figure) au contrôleur du bras. Pendant l'exécution le contrôleur retourne son état (s) au superviseur.

niveaux. La figure E.6 présente l'architecture proposée où chaque composante est implémentée à l'aide d'un module GENOM. GENOM [Fleury 97] est un environnement de développement de logiciel temps-réel embarqué complexe.

Au plus haut niveau, un planificateur de tâche et un superviseur planifient une tâche

comme nettoyer la table ou amener un objet à une personne, puis supervise l'exécution. Le module SPARK construit et maintient un modèle de l'environnement 3D du robot. Celui-ci inclut les objets, les robots et les humains et leurs positions relatives [Sisbot 07b]. Il assure aussi des fonctions de raisonnement géométrique pour évaluer le risque de collision entre les éléments du robot et avec son environnement. Une autre caractéristique importante de SPARK est la production de cartes de coûts qui décrivent une propriété spatiale comme la zone accessible par un humain. Pour des raisons de simplicité, les logiciels de perception ne sont pas présentés ici. Le module tourne à la fréquence de 20Hz qui est limitée par essentiellement par la complexité de la vision 3D et de la perception de l'humain.

Un autre module important est le module MHP qui planifie des mouvements en présence d'humains. Il intègre la planification de prise et de chemin. L'algorithme RRT (exploration aléatoire et rapide d'arbre) et ses variantes [Mainprice 10] sont utilisés par le planificateur de chemin. Le chemin peut être défini dans l'espace cartésien ou dans l'espace articulaire en fonction du type de tâche. A partir du chemin défini par une ligne brisée, une trajectoire associant chemin et évolution temporelle est calculée. MHP calcule une nouvelle trajectoire chaque fois que le planificateur de tâche définit une nouvelle tâche ou que le superviseur décide qu'une nouvelle trajectoire est nécessaire pour réagir à une modification de l'environnement.

Lorsque une modification du mouvement est réalisée en planifiant une nouvelle trajectoire, le calcul complexe de la nouvelle trajectoire pénalise la réactivité du robot. De plus, si l'objet ou l'humain bouge pendant l'exécution de la trajectoire planifiée, le robot risque de s'arrêter et de devoir planifier une nouvelle trajectoire. Ceci donne des mouvements qui semblent peu naturels et intuitifs aux partenaires humains.

Pour éviter ce problème de re-planification pas toujours nécessaire, nous avons conçu un contrôleur réactif basé sur un générateur de trajectoire. Il permet de relier les modules supérieurs aux modules de contrôle inférieurs. Le contrôleur de trajectoire tourne à 100 Hz qui est une fréquence intermédiaire entre celle du planificateur MHP et celle de 1kHz des contrôleurs des moteurs. Ce contrôleur de trajectoire permet au système d'adapter rapidement la trajectoire aux modifications de l'environnement. Le principe de base qui sous-tend ce travail est que toutes les variations de l'environnement ne nécessitent pas une re-planification de la trajectoire qui est très consommatrice de ressources.

E.1.2.3 Contrôle de trajectoire

Le contrôle réactif pour la manipulation d'objet est un des domaine de recherche fondamentaux de la robotique de manipulation. Tout d'abord, les approches basées sur la génération de trajectoire ont été développées. Dans [Buttazzo 94], les résultats d'un système de vision sont d'abord filtré par un filtre passe-bas. Le mouvement de l'objet est modélisé par une trajectoire à accélération constante. Sur cette base, la position et l'instant de saisie sont estimés. Ensuite une trajectoire quintique est calculée pour attraper l'objet avant d'être en-

voyée à contrôleur PID. Les valeurs maximales de l'accélération et de la vitesse ne sont pas vérifiées lors de la planification. Aussi le robot perd la trajectoire lorsque le robot bouge trop vite et que les accélérations et vitesses maximales excèdent les capacités maximales du contrôleur.

Dans [Gosselin 93], les modèles géométrique et cinématique inverses sont étudiés et une application intégrant la saisie d'un objet mobile est implémentée. Le robot utilise une trajectoire quintique pour rejoindre le point le plus près sur la trajectoire. Les systèmes utilisés pour ce travail restent assez simple et il n'y a pas d'humains présents dans la zone de travail. Un travail plus récent est présenté dans [Kröger 12] où le contrôleur réalisant l'asservissement visuel est basé sur un générateur de trajectoire en ligne (Online Trajectory Generation OTG). Les résultats sont très prometteurs.

D'autre part, le domaine de recherche autour du contrôle visuel fournit aussi de nombreux résultats, Chaumette et Hutchinson [Chaumentte 06, Chaumentte 07] fournit une revue et [Farrokh 11] propose une comparaison de différentes méthodes. Les méthodes classiques d'asservissement visuel fournissent des résultats plutôt robustes dont la stabilité peut être étudié rigoureusement, mais elles sont difficiles à intégrer avec un planificateur de chemin et peuvent occasionner des difficultés lorsque les positions initiales et finales sont éloignées.

Une autre approche pour réaliser des mouvements réactifs s'appuie sur l'apprentissage par démonstration (Learning from Demonstration LfD). Dans [Calinon 04] et [Vakanski 12a] les points appartenant aux trajectoires apprises sont regroupés puis utilisés pour construire un modèle de Markov caché (Hidden Markov Model HMM). La classification et la reproduction des trajectoires sont ensuite basées sur les HMM. Une synthèse de cette approche est présentée dans [Argall 09]. Bien que l'apprentissage par démonstration puisse produire le mouvement complet des objets manipulés, de nombreux problèmes apparaissent dans le contexte des HRI car le LfD nécessite un important ensemble de données à apprendre. Les politiques de contrôle apprises peuvent ne pas être adaptées à l'environnement dynamique et imprévisible où évoluent les robots de service.

Le contrôleur intègre des informations très diverses provenant de logiciel de haut niveau comme SPARK ou MHP que nous avons déjà présenté.

E.1.2.4 Fusion multi-sensorielle de données et mesure de force

La fusion multi-sensorielle permet de combiner les informations provenant de plusieurs acquisitions ou de sources différentes. Le but de la fusion de donnée est d'obtenir des informations qui soient de meilleure qualité que les informations prises séparément. En général chaque capteur a ses propres limites que les algorithmes de fusion permettent de dépasser pour obtenir de meilleures données. Le suivi multi-capteur est un des plus importants domaines où la fusion est utilisée pour améliorer las résultats de suivi. On peut trouver une introduction à ces techniques dans [Llinas 98] et une synthèse sur les architectures pour la

fusion de capteur dans [Elmenreich 07]. Smith [Smith 06] propose une synthèse de travaux sur la fusion de capteur pour le suivi de cible que nous appliquerons en robotique dans cette thèse. On peut remarquer que l'apprentissage automatique peut être considéré comme un cas particulier de fusion multi-sensorielle. Pour des raisons de clarté, nous traiterons ces deux aspects séparément dans ce mémoire.

L'estimation des paramètres d'inertie et des forces externes sont étudiées de manière séparée dans la littérature. L'estimation hors-ligne des paramètres d'inertie d'un objet transporté par un manipulateur industriel a été étudié par An et al.[C.H.An 88]. L'approche utilise la méthode d'estimation classique des moindres carrés avec des trajectoires prédéfinies. Swever et al. [Swevers 97] ont étudié le choix des trajectoires d'excitation pour l'estimation. De nombreuses autres approches ont été proposées comme celle de [M.Niebergall 01] qui est essentiellement basée sur une méthode des moindres carrés et ignore de ce fait les incertitudes des données. *Kubus* et al [Kubus 08a] ont proposé une méthode en ligne basée sur les moindres carrés récursifs. La méthode des moindres carrés est appropriée lorsque les paramètres d'inertie sont exprimés dans un repère lié au capteur de force. Nous souhaitons utiliser les paramètres d'inertie pour les comparer avec ceux des objets manipulés qui sont stockés dans une base de données. Mais leurs valeurs dans le repère du capteur de force ne sont pas constantes car l'objet peut être saisi dans différentes positions. Par exemple, attraper une bouteille par le bas ou le centre ne donne pas la même matrice d'inertie dans le repère de l'organe terminal du robot. Pour ce travail, nous calculons la matrice d'inertie dans le repère de l'objet, ce qui rend le système non-linéaire et la méthode des moindres carrés insuffisante.

E.1.2.5 Ondelettes et classification

Nous essayons ici de doter le robot de la capacité de détecter des événements à partir des forces extérieures. Nous avons conçu et réalisé un objet pour enseigner au robot cette capacité qui consiste à extraire des caractéristiques d'un signal temporel et à les classer. La saisie et le lâcher de objet lors d'un échange a été étudie dans plusieurs articles. Nagata et al. ont présenté une solution pour échanger un objet entre un humain et une main à quatre doigts [Nagata 98]. Ils utilisent un capteur de force à 6 axes monté sur chaque doigt pour contrôler la force de saisie et évaluer l'évolution de la saisie de l'humain. Dans le domaine de la manipulation coopérative avec des humains [Aggarwal 07, Takubo 02] essaie de détecter les différentes étapes de coopération comme le contact ou le glissement.

Le travail de Nakazawa et al. [Kim 02, Nakazawa 01, Nakasawa 99] est basé sur une approche similaire de la notre qui mesure les forces pendant l'échange de l'objet. D'autres résultats intéressants sont présenté dans [Romano 11], les chercheurs y mesure les conditions de *vibration* pour détecter le contact avec l'environnement lorsque le robot pose des objets. La condition de vibration est définie par un seuil pour les hautes fréquences de l'accélération de la main. Le seuil y est déterminé par essais et erreurs, ce qui est partic-

ulièrement difficile lorsque des contacts de différents types doivent être classés.

D'autre part, les trajectoires utilisées naturellement par les humains lorsque ils s'échangent un objet ont été étudiées par Kajikawa et al. [Kajikawa 95] pour planifier des mouvements d'échange entre robots et humains. Certaines techniques bien établies peuvent être utilisées pour apprendre les trajectoires et leur synchronisation, elles sont basées, par exemple, sur les modèles de Markov caché [Vakanski 12b] ou les modèles dynamique du système [Khansari-Zadeh 11].

Dans ce travail nous nous focalisons sur la synchronisation de la saisie et de l'ouverture de doigts. Les méthodes que nous utilisons sont basées sur une analyse par ondelettes, le lecteur peut se référer à [Mallat 08] pour une présentation des ondelettes et la comparaison avec l'analyse de Fourier et ses variantes. L'utilisation de la transformé par paquet d'ondelette (Wavelet Packet Transformation WPT) pour l'extraction de caractéristiques a été utilisé dans plusieurs travaux comme [Englehart 99] pour des signaux myoélectriques ou [Learned 95] pour des signaux acoustiques sous-marins.

E.2 Fusion des données de force et de vision

Dans cette partie, nous proposons de résoudre deux problèmes de fusion de données pour réaliser un contrôleur pour la manipulation réactive. Tout d'abord, nous présentons le problème de la mesure de force. Ici l'objectif est d'utiliser un capteur de force 6 axes monté sur le poignet d'un bras de robot pour estimer les paramètres d'inertie de la charge manipulée. Ensuite nous utiliserons les résultats de la mesure des forces pour améliorer le suivi de trajectoire par la fusion force-vision, ainsi que la reconnaissance des objets à partir des paramètres d'inertie et la localisation des objets à partir de la vision.

E.2.1 Estimation des paramètres d'inertie

De nombreuses applications tant pour l'assemblage robotisé que pour la robotique de service peuvent tirer bénéfice de l'estimation des paramètres d'inertie qui regroupent la masse, la position du centre de masse et les éléments de la matrice d'inertie. De plus, les outils complexes ne sont pas toujours fournis avec leur modèle dynamique et l'estimation expérimentale de ces paramètres peut être utile et plus précise. En utilisant le modèle inertiel de l'outil ou de l'objet manipulé, les forces externes exercées sur l'organe terminal ou sur l'objet manipulé sont accessibles pour améliorer le contrôle de la tâche.

Les techniques présentées dans ce chapitre sont aussi utilisées dans la suite du manuscrit pour déterminer les forces et couples externes exercés sur l'objet que nous avons réalisé pour l'apprentissage de l'échange d'objet.

Dans ce paragraphe, nous proposons un modèle des relations entre la dynamique de l'objet manipulé et les forces et couples au niveau du poignet du robot. La figure E.7 présente les repères importants pour le problème. La main, le capteur de force et couple

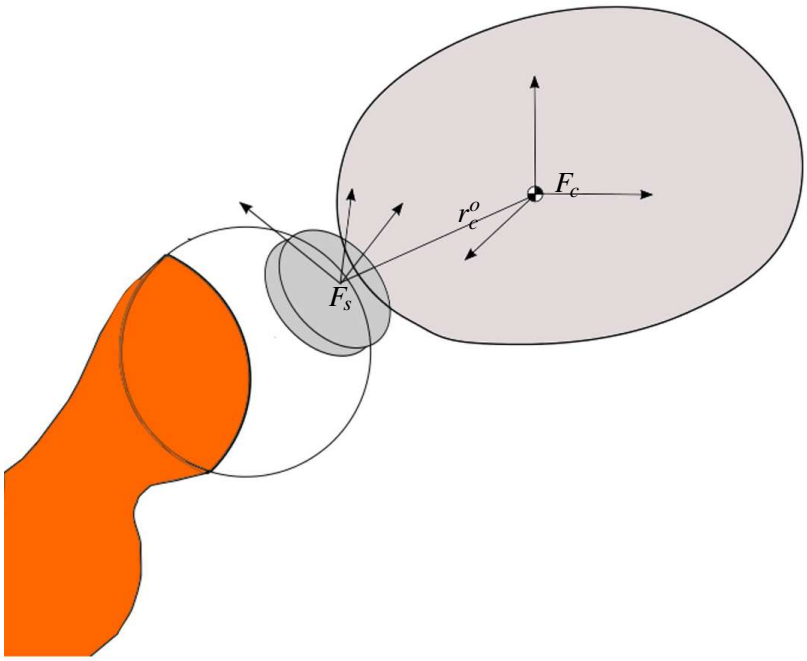


Figure E.7: Les repères utilisés pour la mesure des forces et couples

et éventuellement l'accéléromètre sont tous montés à l'extrémité du bras du robot. Tout d'abord, nous choisissons la base du robot comme repère inertiel de ce système. Cette hypothèse est valide si on suppose que le robot ne manipule pas des objets en faisant des déplacements rapides de sa base. Les matrices de transformation entre les capteurs et la main sont fixes et mesurées expérimentalement. Celle entre la main et la base du bras est calculée à l'aide du modèle géométrique direct du bras.

Une fois la position de l'objet manipulé estimée dans le repère du capteur de force et couple, la position dans la main peut être calculée. Pour la clarté du document, nous ne détaillons pas les termes des matrices. Afin de simplifier le modèle dynamique, nous considérons que les repères de l'objet et de la main sont parallèles. Cette hypothèse n'est pas restrictive car, dans la suite, nous ne cherchons pas à associer les matrices d'inertie, mais leurs caractéristiques essentielles. Les caractéristiques de masse, de position du centre de masse et les valeurs propres de la matrice d'inertie ne sont pas modifiées par ces rotations. Un choix plus spécifique des repères peut être effectué après une première estimation des paramètres. Le capteur de force et couple mesure les forces entre le bras du robot et l'organe terminal, une pince ou une main dans notre cas.

Notations:

$\boldsymbol{f}_m^s$: Forces mesurées par le capteur de force et couple dans le repère du capteur.

$\boldsymbol{\tau}_m^s$: Couples mesurés par le capteur de force et couple dans le repère du capteur.

$\boldsymbol{f}^s$: Forces réelles exercées par l'objet sur le capteur.

$\boldsymbol{\tau}^s$: Couples réel entre le capteur et l'objet.
$\boldsymbol{f}^c_{ext}$: Force de contact sur l'objet exprimées dans le repère de l'objet.
$\boldsymbol{\tau}^c_{ext}$: Couples de contact exercés sur l'objet exprimées dans le repère de l'objet.
$\boldsymbol{f}^s_{off}$: Offsets de force du capteur.
$\boldsymbol{\tau}^s_{off}$: Offsets de couple du capteur.
$\boldsymbol{f}^c$: Force totale exercée au centre de masse.
$\boldsymbol{\tau}^c$: Couple total exercé au centre de masse.

Nous obtenons $\boldsymbol{f}^s_m$, $\boldsymbol{\tau}^s_m$, les forces et couples mesurés dans le repère du capteur par :

$$\boldsymbol{f}^s = \boldsymbol{f}^s_m - \boldsymbol{f}^s_{off} \tag{E.1}$$
$$\boldsymbol{\tau}^s = \boldsymbol{\tau}^s_m - \boldsymbol{\tau}^s_{off} \tag{E.2}$$

L'équation de l'équilibre statique de l'objet donne :

$$\boldsymbol{f}^c = \boldsymbol{R}^c_s \boldsymbol{f}^s + \boldsymbol{f}^c_{ext} \tag{E.3}$$
$$\boldsymbol{\tau}^c = \boldsymbol{R}^c_s \boldsymbol{\tau}^s + \boldsymbol{r}^c_s \times (\boldsymbol{R}^c_s \boldsymbol{f}^s) + \boldsymbol{\tau}^c_{ext} \tag{E.4}$$

où $\boldsymbol{R}^c_s$ est la matrice de rotation 3×3 du repère du capteur $\mathscr{S}$ au repère de l'objet $\mathscr{C}$ and $\times$ est le produit vectoriel. Lorsque les repères du capteurs et de l'objet sont parallèles, $\boldsymbol{R}^c_s$ devient la matrice identité. $\boldsymbol{r}^c_s$ est le vecteur de position $\mathscr{S}$ à $\mathscr{C}$:

$$\boldsymbol{r}^c_s = (c_x \; c_y \; c_z)^T \tag{E.5}$$

Nous considérons un processus de calcul en deux étapes :

1. Déplacer le bras dans l'espace libre pour estimer l'ensemble des paramètres d'inertie.

2. À l'aide des paramètres d'inertie, la dynamique de l'objet est simulée pour éliminer les forces internes du signal de force et couple pendant les manipulations.

Lorsque le robot bouge dans l'espace libre, les forces de contact sont nulles et les forces et couples mesurés se réduisent aux forces et couples inertiels.

$$\boldsymbol{f}^c = \boldsymbol{R}^c_s \boldsymbol{f}^s \tag{E.6}$$
$$\boldsymbol{\tau}^c = \boldsymbol{R}^c_s \boldsymbol{\tau}^s + \boldsymbol{r}^c_s \times (\boldsymbol{R}^c_s \boldsymbol{f}^s) \tag{E.7}$$

Dans un repère placé au centre de masse, les équations de Newton-Euler s'écrivent :

$$m\boldsymbol{a}^o = \boldsymbol{R}^o_c \boldsymbol{f}^c + m\boldsymbol{g}^o \tag{E.8}$$
$$\boldsymbol{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{\tau}^c \tag{E.9}$$

où la matrice d'inertie:

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{E.10}$$

L'équation E.8 est écrite dans le repère de base du robot $\mathscr{O}$. Dans ce repère :

$\boldsymbol{R}_c^o$**:** est la matrice de passage du repère de l'objet $\mathscr{C}$ au repère $\mathscr{O}$

a^o**:** est l'accélération linéaire du centre de masse par rapport au repère $\mathscr{O}$.

Les variables de l'équation (E.9) sont toutes écrites dans le repère $\mathscr{C}$. il en résulte :

$\boldsymbol{\omega}$ est la vitesse angulaire de l'objet par rapport au repère local de l'objet $\mathscr{C}$

Remplaçant f^c et $\boldsymbol{\tau}^c$ dans les équations (E.8),(E.9) par (E.6),(E.7), et comme nous avons fait l'hypothèse que les repères $\mathscr{C}$ er $\mathscr{S}$ sont parallèles, $\boldsymbol{R}_s^c$ est la matrice identité, aussi $\boldsymbol{R}_c^o$ et $\boldsymbol{R}_s^o$ sont égales, on obtient le système complet pour les mouvements dans l'espace libre :

$$m\boldsymbol{a}^o = \boldsymbol{R}_s^o(\boldsymbol{f}_m^s - \boldsymbol{f}_{off}^s) + m\boldsymbol{g}^o \tag{E.11}$$

$$\boldsymbol{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{\tau}_m^s - \boldsymbol{\tau}_{off}^s + \boldsymbol{r}_s^c \times (\boldsymbol{f}_m^s - \boldsymbol{f}_{off}^s) \tag{E.12}$$

Dans ces équations, l'accélération linéaire $\boldsymbol{a}^o$, la vitesse angulaire $\boldsymbol{\omega}$ et l'accélération angulaire $\boldsymbol{\alpha}$ peuvent être mesuré par une centrale inertielle montée sur la main ou la pince du robot. Comme proposé par Kroger dans [Kubus 08b], ces paramètres peuvent aussi être obtenus à partir des capteurs de position des bras du robot. $\boldsymbol{R}_s^o$ peut être calculé en inversant $\boldsymbol{R}_o^s$ qui est définie par le modèle géométrique. $\boldsymbol{f}_m^s$ et $\boldsymbol{\tau}_m^s$ sont mesurés avec le capteur de force et couple. Les paramètres à estimer sont :

$$\boldsymbol{\varphi} = (m \;\; c_x \;\; c_y \;\; c_z \;\; I_{xx} \;\; I_{xy} \;\; I_{xz} \;\; I_{yy} \;\; I_{yz} \;\; I_{zz} \;\; f_{offx} \;\; f_{offy} \;\; f_{offz} \;\; \tau_{offx} \;\; \tau_{offy} \;\; \tau_{offz})^T \tag{E.13}$$

Où $\boldsymbol{f}_{offx}$ est un l'offset du capteur de force dans la direction x.

Pour représenter la dynamique du système sous la forme d'espace d'état, nous choisissons $\boldsymbol{\varphi}$ comme vecteur d'état. De même le vecteur d'observation est:

$$\boldsymbol{y} = (f_x^s \;\; f_y^s \;\; f_z^s \;\; \tau_x^s \;\; \tau_y^s \;\; \tau_z^s)^T \tag{E.14}$$

Les équations (E.11) et (E.12) correspondent au modèle de l'observation que nous notons $\boldsymbol{h}$. Le système s'écrit alors :

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\varphi}_k + \boldsymbol{v}_k \tag{E.15}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{\varphi}_k, R_s^o, \boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{a}^o, \boldsymbol{g}^o) + \boldsymbol{n}_k \tag{E.16}$$

où $\boldsymbol{v}_k$ et $\boldsymbol{n}_k$ sont les bruits de processus et de mesure respectivement. La formulation des filtres de Kalman est présentée en annexe.

Il est évident que le système est non-linéaire. Les paramètres du modèle $R_s^o, \boldsymbol{\alpha}, \boldsymbol{\omega}$ et $\boldsymbol{a}^o$ sont issus des capteurs et du modèle géométrique du robot. Ainsi la dynamique de l'objet est modélisée par un système dynamique où les paramètres inertiels doivent être estimé. Le filtrage de Kalman est adapté pour ce type de problème. Ici la difficulté provient de la nature de $\boldsymbol{h}$ qui varie non-linéairement avec le temps. Parmi la grande variété de filtres non-linéaires, nous avons choisi un filtre de Kalman inodore (Unscented Kalman Filter UKF). Comme nous utilisons les mêmes techniques de filtrage pour le problème de fusion de données multi-capteurs, nous présenterons les UKF après le paragraphe sur la modélisation du problème de suivi.

E.2.1.1 Reconnaissance d'objet en ligne

La reconnaissance d'objet utilise généralement des caméras ou de plus en plus souvent des capteurs de profondeur. Après leur estimation, les paramètres d'inertie d'un objet peuvent aussi être utilisé comme éléments caractéristiques. De nombreuses raisons motivent cette utilisation pour la reconnaissance des objets. Tout d'abord, la vision n'est pas toujours très fiable, c'est notamment le cas pour la robotique de service. L'objet peut, par exemple, être partiellement caché par la main du robot. D'autre part, les paramètres d'inertie contiennent des informations inaccessibles à d'autres capteurs comme, par exemple, la quantité de liquide à l'intérieur d'un bidon.

Un des avantages du modèle proposé dans les paragraphes précédents est que les paramètres estimés sont directement exprimés dans le repère de l'objet. Les valeurs de la matrice d'inertie varient avec l'orientation et la position de l'origine du repère où la matrice est exprimée, mais ses valeurs propres sont invariantes et peuvent être utilisées pour la reconnaissance d'objet. Si on note I_1, I_2 et I_3 les moments principaux d'inertie correspondant aux valeurs propres de la matrice d'inertie, l'ensemble des paramètres utilisables pour l'identification sont défini par :

$$F = [m,\ I_1,\ I_2,\ I_3] \tag{E.17}$$

où m est la masse de l'objet.

E.2.1.2 Calcul des forces et couples de contact

Après l'estimation de l'ensemble des paramètres d'inertie, la dynamique de l'objet peut être simulée. Les forces et couples de contact sont alors calculé par :

$$\boldsymbol{f}_{contact} = \boldsymbol{f}_{sensor} - \boldsymbol{f}_{inertia} \tag{E.18}$$

$$\boldsymbol{\tau}_{contact} = \boldsymbol{\tau}_{sensor} - \boldsymbol{\tau}_{inertia} \tag{E.19}$$

La structure de ces calcul se résume à :

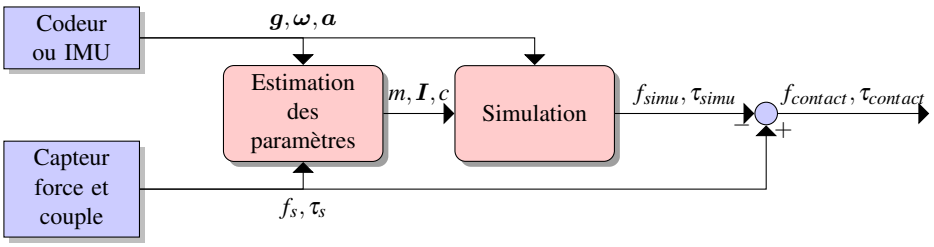


Figure E.8: Calcul des force et couples de contact. Les vecteurs de gravité $\boldsymbol{g}$, d'accélération $\boldsymbol{a}$ et de vitesse angulaire $\boldsymbol{\omega}$ sont calculés à partir de la position des articulations et de la cinématique du robot ou à partir des données d'une centrale inertielle (inertial measurement unit IMU). f_s et τ_s représentent le vecteur des mesures de force et de couple.

Ici, le principal défi est de détecter si l'estimation converge réellement. Une approche consiste à supposer que les trajectoires d'excitation sont prédéfinies pour garantir la convergence. Pour détecter la convergence, il est aussi naturel de considérer que le système converge lorsque les paramètres sont stabilisés. Mais ici les paramètres réels sont inconnus.

E.2.2 Fusion force-vision et suivi multi-modal

Le suivi de cible par vision est un domaine de recherche très vaste. Nous n'abordons pas ici le problème de la reconnaissance, mais utilisons simplement des marqueurs collés sur les objets et un système de vision spécifique. Cet outil spécifique, comme d'autres outils de vision, produit des données bruitées, notamment lorsque l'éclairement varie ou que les marqueurs sont partiellement occultés.

Le système de stéréovision calcule la position relative T_V^C du repère C lié à l'objet par rapport au repère V associé à la paire de caméra. T_V^C est une matrice de passage.

$$T_V^C = \begin{bmatrix} & & & x \\ & R_{3\times3} & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{E.20}$$

où R est une matrice de rotation. Les caméras sont montés sur une platine qui s'oriente pendant les manipulations. La position de l'objet est donc définie par :

$$T_B^C = T_B^V T_V^O \tag{E.21}$$

où T_B^C est calculée à partir du modèle cinématique de la platine. Bien que très simple, ce calcul cache une difficulté : une petite erreur dans cette matrice peut entrainer des erreurs importantes pour la localisation de l'objet et causer de sérieux problème pendant la manipulation. Une procédure d'identification des paramètres définissant la position relative

des caméras par rapport à la main peut être utilisée, la littérature en propose de nombreux exemples [Dornaika 98].

Nous effectuons les calculs dans le repère de base du robot. Ce choix est préférable à celui d'un repère fixe de la scène du point de vu précision car la position du robot dans la scène n'est pas toujours connue avec précision. L'expression de la position de l'objet dans le repère W de la scène peut être obtenu par :

$$T_W^C = T_W^B T_B^V T_V^C \tag{E.22}$$

où T_W^B est la position de la base par rapport à la scène. Les erreurs de localisation du robot se retrouvent dans cette position. Cette approche est nécessaire dans le cas où la base du robot bouge durant la tâche de manipulation, ce qui n'est pas le cas pour les travaux effectué pendant cette thèse.

Comme nous avons choisi d'utiliser le filtrage de Kalman non-linéaire pour la fusion des données, nous devons construire un modèle différentiel du mouvement de l'objet ou un modèle récurent car le calcul sera itératif. Nous supposons que l'objet est tendu par une personne ou par le robot et bouge à vitesse constante avec un bruit sur la connaissance des position et vitesse.

Ce choix se justifie car nous travaillons dans le domaine du robot de service et la vitesse de l'objet est faible. D'autre part, considérer l'accélération et plus encore le jerk n'améliorerait pas nécessairement les performances car la procédure de filtrage ajuste déjà la vitesse à chaque itération. Étant donné que l'objet est tendu par un autre agent, le modèle à vitesse constante est aussi suffisant pour assurer le suivi de l'objet. L'état du modèle $\boldsymbol{x}$ est de dimension 12 :

$$\boldsymbol{x} = (\boldsymbol{x}_p^T \ \dot{\boldsymbol{x}}_p^T)^T \tag{E.23}$$

où $\boldsymbol{x}_p = (x \ y \ z \ \varphi \ \theta \ \phi)$. Ce vecteur correspond à la position et à l'orientation suivies.

Si on suppose que le système de suivi fonctionne avec une période de δt entre chaque actualisation, au temps t l'équation d'état de la dynamique du système peut s'écrire :

$$\boldsymbol{x}(t+\delta t) = A_{\delta t}\boldsymbol{x}(t) + \omega_t \tag{E.24}$$

où la matrice $A_{\delta t}$ peut s'exprimer :

$$A(\delta t) = \begin{bmatrix} I_6 & I_6\delta t \\ 0 & I_6 \end{bmatrix} \tag{E.25}$$

I_3 est la matrice identité de dimension 3. Ce modèle est basé sur l'hypothèse que la cible bouge avec une vitesse constante dans toutes les directions pendant une période. Le bruit du processus que nous supposons être un bruit blanc gaussien est le vecteur ω_t. La matrice

de covariance devient alors :

$$E\{\omega_t \omega_{t+\varepsilon}^T\} = \begin{cases} Q(\delta t) & \varepsilon = 0, \\ 0 & \varepsilon \neq 0. \end{cases} \tag{E.26}$$

Le terme $Q(\delta t)$ doit être calculé à partir du bruit réel. Les valeurs peuvent être ajustées pour améliorer la dynamique du filtre, toutefois le plupart du temps le filtre identifie le processus même avec un réglage grossier des termes de covariance du bruit. La partie relative à l'identification de l'orientation a été placée dans le modèle d'observation plutôt que dans le modèle dynamique. Durant chaque itération, les valeurs des incréments d'orientation sont simplement des valeurs de l'état et à la fin de chaque itération ils sont remis à zéro.

Pour éviter les problèmes des angles d'Euler qui n'ont pas une représentation unique, nous utilisons des incréments d'angle. Ces incréments sont remis à zéro à la fin de chaque itération de l'identification et l'observation de ces angles est simplement la différence avec la dernière mise à jour. À la fin de la fonction de filtrage de Kalman, on fait le calcule:

$$(\varphi\ \theta\ \phi) = (0\ 0\ 0) \tag{E.27}$$

Les rotations absolues de l'objet est gardées à l'extérieur du filtre de suivi, sous la forme de

$$\boldsymbol{q} = \{q_0\ q_1\ q_2\ q_3\} \tag{E.28}$$

E.2.2.1 Modèles de mesure

L'utilisation du filtrage de Kalman nécessite un modèle de mesure pour chaque source d'information. Comme nous ne considérons que des angles incrémentaux dans le modèle dynamique, les valeurs absolues des angles doivent être incluses dans les valeurs externes du filtre. Les modèles de mesure définissent les relations qui relient les incréments de rotation avec les données mesurées disponibles sur le robot. Dans notre cas, ce sont la vision 3D, le capteur de force et les mesures au niveau des articulations.

Mesure avec la vision: Pour des raisons de concision et de clarté, nous utilisons les quaternions pour représenter les rotations absolues. La vision 3D fournit la position des objets dans le repère du robot. L'expression de l'incrément de quaternion à partir des angles incrémentaux s'écrit :

$$\Delta\boldsymbol{q} = h(\varphi\ \theta\ \phi) \tag{E.29}$$

où, φ, θ et ϕ sont les incréments des angles d'Euler comme dans l'équation E.23 qui est non-linéaire. φ, θ, et ϕ représentent l'état du modèle du processus (E.24). Cette équation est l'équation de l'observation du processus d'identification. A chaque étape, le quaternion

conservant la rotation absolue est mis à jour par le produit de quaternion suivant :

$$\widehat{\boldsymbol{q}} = \widehat{\boldsymbol{q}} \bigotimes \Delta \boldsymbol{q} \tag{E.30}$$

Mesure des forces et couples : Le processus de mesure des forces présenté dans ce paragraphe fournit la position P_S^C de l'objet C dans repère S du capteur. Nous utilisons les notations homogènes, aussi $P_S^C = [c_x\ c_y\ c_z 1]$ comporte un 1 comme quatrième élément. La position du capteur de force par rapport au repère terminal du bras est donnée par la matrice T_E^S. Le modèle géométrique du robot est défini par la matrice de passage T_E^S. La position de l'objet dans le repère du robot s'exprime alors :

$$P_R^C = T_R^E T_E^S P_S^C \tag{E.31}$$

La fréquence d'actualisation de la position T_R^E du robot est plus rapide que celle de la fusion. Il faut aussi mentionner que la mesure de la position des articulations est bruité et doit être filtré.

E.2.3 Résultats de simulation et expérimental pour la fusion

La procédure de mesure des forces incluant l'estimation des paramètres inertiel, la simulation dynamique des corps rigides et la compensation des forces d'inertie a été testée en simulation et hors-ligne avec des données réelles. La procédure complète incluant l'identification de l'objet n'a pas été testée en-ligne avant l'écriture de ce document à cause de difficultés pour intégrer le capteur de force sur le bras manipulateur. Toutefois, le suivi a été implémenté et testé en-ligne avec le capteur de vision 3D et l'estimation de la position à partir de la position des articulations et des forces mesurés au niveau du poignet.

Le suivi d'une cible par vision 3D donne un résultat instable lorsque les conditions d'éclairement sont mauvaises comme on peut le voir sur la figure E.9 pour une translation et pour une rotation sur la figure E.10. Le filtrage de Kalman améliore le suivi de la cible en réduisant le bruit comme cela a été discuté dans plusieurs articles de la littérature. Il agit comme un filtre passe bas. Ici, nous pouvons voir que dans de mauvaises conditions d'éclairage, le filtre réduit les oscillations de manière importante. Les oscillations peuvent déstabiliser un contrôleur basé sur un suivi visuel de cible.

E.3 Apprendre pour échanger des objets

La manipulation mobile commence à sortir des laboratoires et la manipulation coopérative entre humains et robots dans des environnements domestique ou industriel soulève de nouveaux défis. Par exemple, l'humain peut n'avoir aucune expérience pour manipuler un robot alors que les chercheurs en ont. Aussi, ils attendent du robot qu'il agisse comme un humain. Pour atteindre cet objectif, nous avons conçu et réalisé un appareil pour apprendre au robot

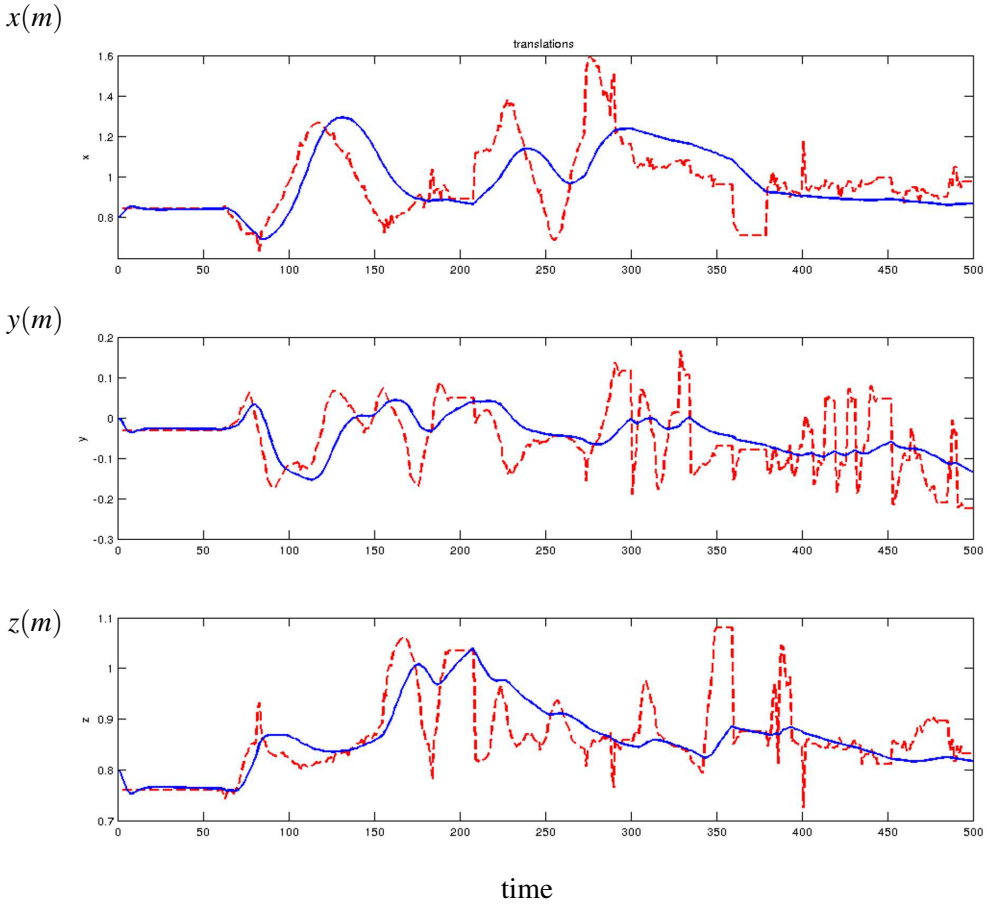


Figure E.9: Suivi visuel pendant 10s du mouvement de translation avant et après filtrage. La courbe en pointillé rouge correspond au résultat de la vision avant filtrage et la courbe bleu au résultat après filtrage. Les données ont été acquises dans de mauvaises conditions d'éclairement pour montrer la dynamique du filtre. Le temps est exprimé en période de 0.02s.

à échanger des objets avec les humains de manière naturelle et intuitive. Avec cet appareil nous avons étudié l'échange d'objets entre deux humains, puis nous avons porté le modèle obtenu sur le robot pour améliorer l'échange d'objet entre humains et robots.

E.3.1 Bidule: un appareil pour apprendre la manipulation

Le modèle à apprendre sera utilisé sur un bras manipulateur équipé d'un capteur de force 6D monté sur le poignet même si le modèle doit aussi fonctionner avec les forces estimées. Les capteurs de forces et couples sont assez courant sur les bras de robots et peuvent être utilisé à d'autres fins comme on le verra au chapitre suivant. Pour détecter les événements, nous mesurons la force d'interaction pendant l'échange. Pour rendre l'échange aussi naturel et intuitif que possible, nous avons développé un appareil nommé "Bidule" qui est

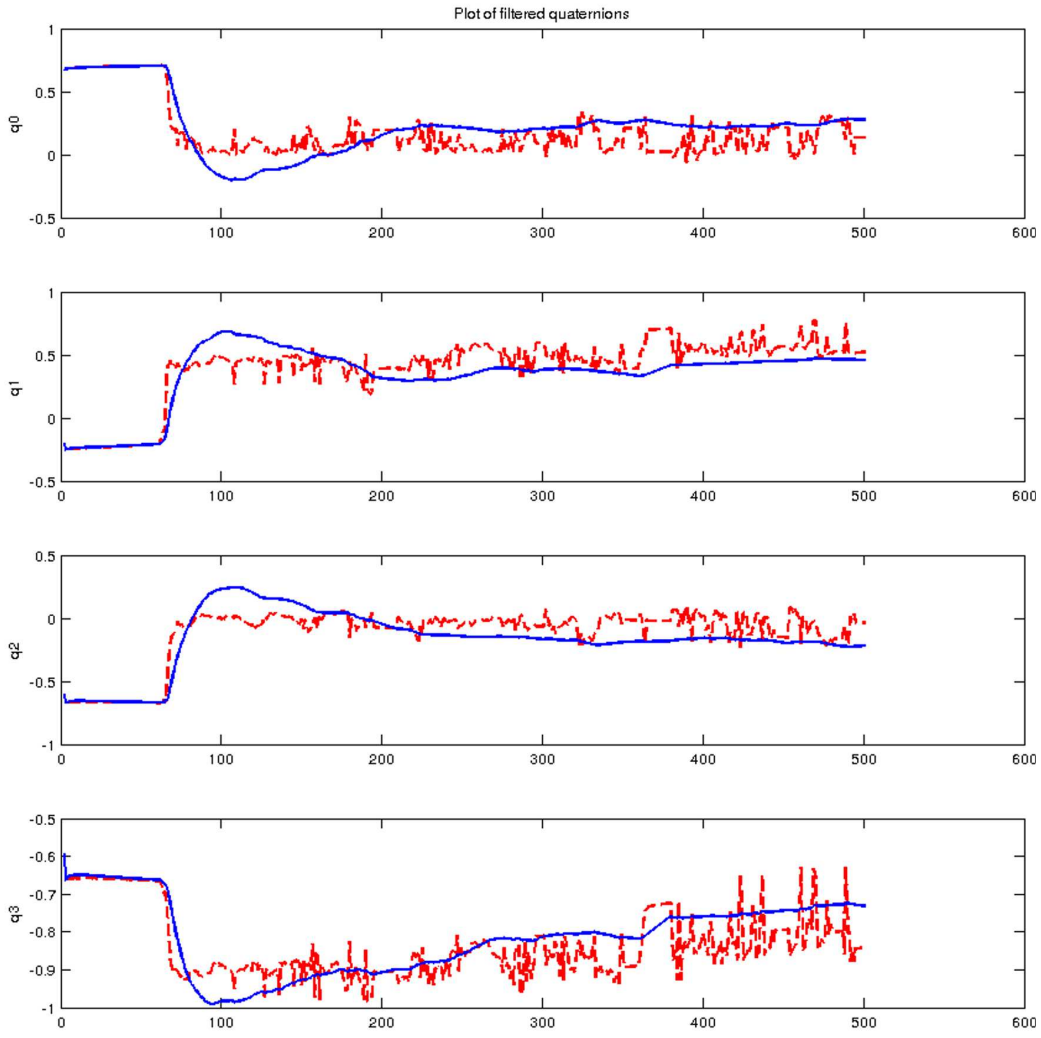


Figure E.10: Suivi visuel par quaternion d'un mouvement de rotation pendant 10s. La ligne pointillée rouge correspond au quaternion avant filtrage et la ligne bleu au quaternion après filtrage. Les données ont été acquises dans de mauvaises conditions d'éclairement pour montrer la dynamique du filtre. Le temps est exprimé en période de 0.02s.

équipé de capteurs. Il peut être utilisé pour enregistrer des informations lorsque l'objet est échangé par deux personnes. Pour étudier l'échange d'objet entre personnes et détecter la prise, nous utilisons l'analyse par ondelette et des techniques de classification. Le modèle de classification appris à partir des expériences effectuées par des humains est utilisé pour

apprendre au robot comment si l'humain saisi l'objet échangé et décider quand réagir. Le même modèle peut être utilisé pour d'autres types de manipulation comme déposer un objet sur une table. L'appareil est aussi destiné à aider à l'apprentissage des trajectoires et des lois de contrôle en force. Toutefois ces derniers aspects ne sont pas abordés dans ce document. L'objectif de ce document est de présenter un ensemble de techniques pour extraire efficacement des évènements d'un signal pour la manipulation robotisée. Le signal peut provenir de différente source capteur de force, capteur tactile ou autre.

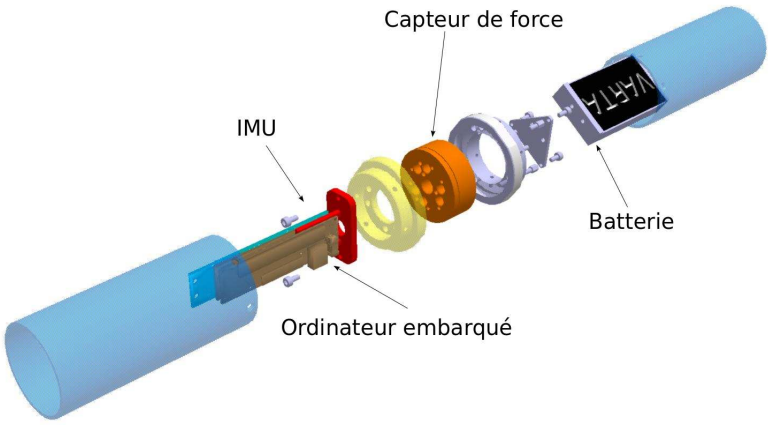


Figure E.11: Vue éclatée de Bidule avec le capteur de force 6D au centre en orange, les poignées bleus sont fixés sur le capteur, le micro-ordinateur est sur la gauche et la batterie sur le coté droit.

Bidule est construit autour d'un micro-ordinateur embarqué, communique par WiFi et est équipé d'un capteur de force et couple 6D. Une centrale inertielle (IMU) est utilisée pour compenser les forces d'inertie et pour enregistrer le mouvement de Bidule. La forme et la taille ont été choisies pour être adaptés à la main humaine (Fig.E.11). Les deux poignées peuvent être saisis par un humain et transmettent les efforts au capteur de force.

Les signaux analogiques du capteur de force et de l'IMU sont amplifiés et filtrés avant d'être numérisé. Deux anneaux en bronze peuvent être ajoutés pour accroitre la masse de Bidule afin d'étudier l'influence de la masse de l'objet lors des échanges.

E.3.2 Traitement du signal et détection d'événements

E.3.2.1 Transformé par paquet d'ondelette

Les différents signaux de force enregistrés sont irréguliers et dépendent des humains et de l'objet manipulé. Sur la figure E.12, un exemple de force exercée lorsque l'objet est saisi par une personne et d'une collision avec la table sont montré. Nous pouvons remarquer que les vibrations sont de nature différentes. La saisie fait apparaître un double pic alors que la collision engendre de plus hautes fréquences. Dans les deux cas, ni l'amplitude, ni la

durée des vibrations ne permettent de distinguer facilement le type de signal. Cependant des caractéristiques peuvent être extraites pour identifier l'événement, c'est précisément ce que nous souhaitons extraire à l'aide des techniques d'apprentissage.

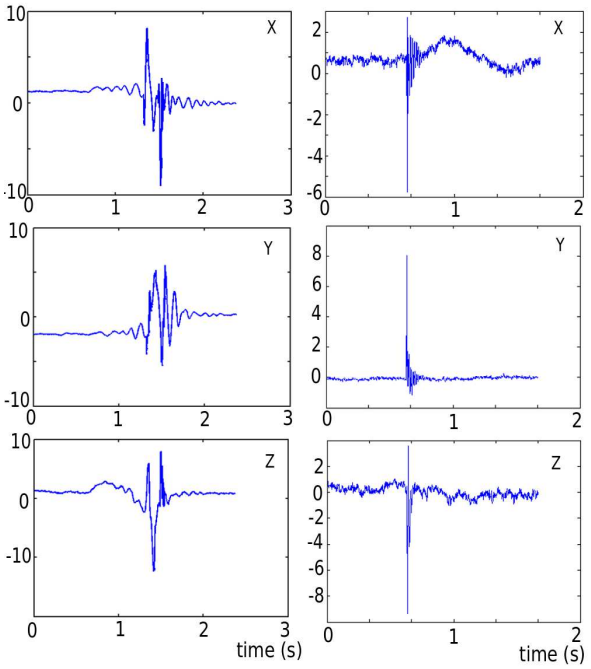


Figure E.12: Deux cas typiques de signaux de forces en N. À gauche, les forces mesurées par Bidule lorsque une personne a saisi l'objet tendu par une autre personne. À droite, les forces mesurées lorsque bidule a heurté une table.

À cause de la redondance de la Transformé par paquet d'ondelette (WPT), plusieurs représentations existent pour une caractéristique, une partie des coefficients de décomposition, des informations statistiques sur les coefficients ou la carte d'énergie [Fatourechi 04, Birbaumer 00, Xue 03]. L'ensemble des coefficients capture toutes les classes de variations et de bruits. Mais il faudrait un grand nombre d'expérience pour les apprendre tous. Deux raisons nous poussent à sélectionner proprement les caractéristiques et à réduire la taille de l'ensemble des caractéristiques. Tout d'abord, la construction d'un grand ensemble de données de manipulation est un travail fastidieux. Ensuite, le classement doit être effectué en ligne à la fréquence du contrôleur. Nous proposons de considérer le temps et la fréquence séparément en combinant d'une part des coefficients et d'autre part une partie de la carte d'énergie.

La carte d'énergie relative représente la carte de la distribution du signal par rapport au temps et à la fréquence. La carte d'énergie est utilisée pour capturer l'information fréquentielle. Chaque niveau de la WPT décompose le signal en une représentation temps-fréquence de différente précision. Pour chaque sous-bande de l'arbre, voir figure E.13, une valeur de l'énergie est calculée à partir de tous les coefficients de la sous-bande. Si nous appelons b une sous-bande et $x(k)$ les coefficients de cette sous-bande, alors son énergie e_b

est :

$$e_b = \sum_k (x(k))^2 \tag{E.32}$$

L'énergie est calculée à travers tout l'arbre jusqu'au niveau qui fournit une précision suffisante pour la fréquence, celle-ci est définie préalablement. Ensuite toutes ces énergies sont normalisées par rapport à l'énergie totale (e_0) du signal dans la fenêtre temporelle :

$$E_b = \frac{e_b}{e_0} \tag{E.33}$$

où e_0 est l'énergie totale du signal. Pour un échantillon à apprendre, nous écrivons l'ensemble des caractéristiques $\boldsymbol{E} = \{E_b\}$, il inclut l'énergie relative de toutes les sous-bandes sélectionnées. Remarquons qu'un grand nombre d'éléments à apprendre sont nuls, ce qui est normal car les ondelettes produisent une représentation clairsemée du signal originel. Voir [Mallat 08] pour plus de détail sur cette propriété de WPT. Nous devons ensuite choisir les caractéristiques les plus discriminantes de $\boldsymbol{E}$ pour réduire encore plus la taille du vecteur de caractéristique.

La carte d'énergie est indépendante de l'amplitude et de la durée des vibrations dans le signal originel. Elle ne conserve pas non plus l'information temporelle.

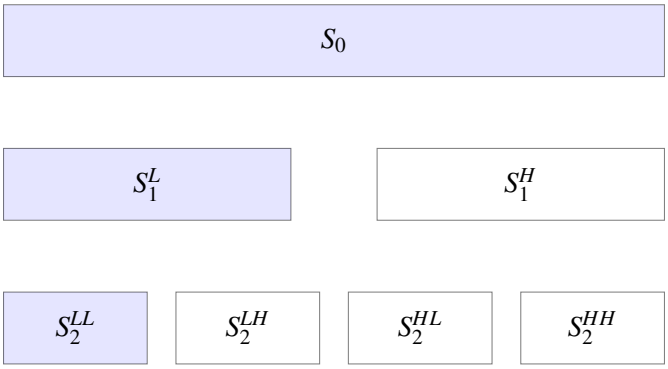


Figure E.13: Les sous-bandes d'un arbre WPT à trois niveaux. *L* signifie que la sous-bande est filtrée par un filtre passe-bas, puis sa fréquence d'échantillonnage réduite. *H* signifie que la sous-bande est filtrée par un filtre passe-haut, puis sa fréquence d'échantillonnage réduite. Les sous-bande bleus représentent la partie basse fréquence de chaque niveau.

Les coefficients des sous-bandes sont sélectionnés pour capturer la forme globale du signal. Dans notre cas, l'objectif est double : conserver l'information sur la variation du poids lorsque le receveur commence à tenir l'objet échangé et déduire le bon instant pour ouvrir les doigts. Le robot soit aussi différentier deux signaux identiques mais décalés dans le temps. Nous choisissons les coefficients de la première sous-bande d'un niveau de l'arbre WPT. Il contient l'information de basse fréquence du signal avec une taille réduite. Nous notons $\boldsymbol{S}$ cet ensemble de caractéristique. Sur la figure E.13, les candidats sont de couleur bleu. Comme chaque niveau contient une sous-bande de basse fréquence, le choix du niveau

est fait à partir d'un compromis entre l'erreur de classification et la taille de l'ensemble des caractéristiques produites. Plus le niveau de la sous-bande est élevé et moins elle est précise en temps. Les coefficients sont ensuite normalisés entre zéro et un comme pour la carte d'énergie relative.

Le choix des coefficients de la première sous-bande d'un niveau est basé sur une connaissance préalable du signal pour toutes les caractéristiques du problème considéré et notamment cette sous-bande doit bien capturer les pics du signal de force. Pour certains signaux qui contiennent uniquement des composantes de haute fréquence, le choix de la sous-bande doit être fait par un algorithme de sélection des caractéristiques.

E.3.2.2 Analyse discriminante linéaire de Fisher

Nous proposons d'utiliser l'analyse discriminante linéaire de Fisher (Fisher LDA) pour sélectionner les caractéristiques et réduire la dimension du problème de classification. La LDA de Fisher est un outil de classification et de réduction de la dimension des problèmes. Un des outils les plus utilisés pour la réduction de la dimension est l'analyse en composantes principales (PCA). Les deux techniques sont comparables, mais la LDA est mieux adapté à notre problème. En effet la PCA est une transformation linéaire orthogonale qui transforme les données dans un nouveau référentiel dans lequel la plus grande variance d'une projection des données sur une direction est obtenue pour la première coordonnée, dite composante principale, la seconde plus grande sur le deuxième axe et ainsi de suite.

Considérons un ensemble de données $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ dans lequel $x_i \in \mathbb{R}^P, 0 \leq i \leq N$ est une donnée de l'ensemble. La moyenne est supposé nulle pour cet ensemble, si elle n'est pas nulle l'ensemble peut être normalisé. Les N lignes représentent les N données d'entré qui comprennent chacune P éléments. L'objectif est de trouver une matrice de projection telle que :

$$\boldsymbol{Y} = W\boldsymbol{X} \tag{E.34}$$

telle qu'une variable individuelle de $\boldsymbol{Y}$ considérée hérite du maximum possible de variance des $\boldsymbol{X}$ avec la contrainte que chaque vecteur w_i de W considéré soit unitaire.

La LDA de Fisher cherche le meilleur poids $\boldsymbol{w}$ pour maximiser l'objectif :

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} \tag{E.35}$$

où S_B est la matrice de dispersion entre les classes et S_W la matrice de dispersion avec les classes. La valeur de J représente la séparabilité des classes pour l'ensemble ou le sous-ensemble considéré de caractéristiques. Les deux matrices sont calculées par :

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{E.36}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{E.37}$$

où $\bar{x}$ est la moyenne pour l'ensemble des points, c représente l'étiquette de la classe et μ_c est la valeur moyenne pour la classe de données. Une propriété important de la projection J est qu'elle est invariante par mise à l'échelle du vecteur $\boldsymbol{w}$. Ceci est utile car il est toujours de choisir $\boldsymbol{w}$ pour que $\boldsymbol{w}^T S_W \boldsymbol{w} = 1$. Le problème de la maximisation de J se transforme alors en un problème d'optimisation défini par :

$$min_w \quad -\frac{1}{2}\boldsymbol{w}^T S_B \boldsymbol{w} \tag{E.38}$$

$$s.t. \quad \boldsymbol{w}^T S_w \boldsymbol{w} = 1 \tag{E.39}$$

E.3.3 Sélection des caractéristiques pour la LDA de Fisher

Ici nous définissons le critère de sélection des meilleures caractéristiques de l'ensemble des caractéristiques considérés par la carte des énergies relatives $\boldsymbol{E}$. Nous pouvons reformuler le critère dans l'équation E.40. Il est calculé pour chaque caractéristique de E pour évaluer la séparabilité de cette caractéristique [Gu 11]. Comme la transformation en ondelette par paquet (WPT) produit une représentation clairsemée du signal, c'est à dire avec beaucoup de zéro, il raisonnable d'exclure ces caractéristiques qui ne différentient pas les classes.

$$J = tr(S_W^{-1} S_B) \tag{E.40}$$

Par rapport à la définition cette expression E.35 définit la séparabilité et omet le vecteur de projection. Nous n'avons pas utilisé directement la LDA comme outil de classification car notre problème n'est pas linéairement séparable et l'ensemble de donnée est assez petit. Pour choisir un outil de classification pour un ensemble de données qui se chevauchent et dont les informations sont incomplètes, nous utilisons un outil de classification basé sur le noyau. Nous voulons réduire la taille des caractéristiques et nous avons observé que de nombreuses caractéristiques sont nulles ou presque nulles (WPT produit une représentation clairsemée). La LDA semble comme nous le verrons dans les résultats bien adapté dans ce cas.

D'autre part, pendant l'échange d'un objet, l'orientation des efforts change relativement à l'objet et l'exclusion de ces informations permet de simplifier le problème, de réduire le nombre de caractéristique et d'accélérer la classification. Nous pouvons résumer la construction de l'ensemble des caractéristiques par :

- La construction de l'arbre WPT à partir des forces F et des moments T avec $F =$

$||Fx+Fy+Fz||$ et $T = ||Tx+Ty+Tz||$.

- Le calcul de la carte des énergies relatives pour l'arbre WPT.
- Le choix d'une sous-bande de l'arbre WPT et sa mise à l'échelle entre 0 et 1.
- Utilisation du critère de la LDA de Fisher pour réduire la taille de l'ensemble des caractéristiques retenues de la carte d'énergie $\boldsymbol{E}$ et le niveau de la sous-bande $\boldsymbol{S}$ retenue en faisant un compromis entre la précision de la classification et la taille de l'ensemble des caractéristiques.

E.3.4 Classification et machine à vecteur de pertinence

Après avoir sélectionné les caractéristiques, le problème peut se formuler comme un problème de classification de plusieurs classes. Dans le paragraphe précédent nous avons construits un ensemble de caractéristiques à partir des signaux de force et couple. Nous pouvons écrire l'ensemble de ces caractéristiques $\boldsymbol{x}$, le problème peut alors s'écrire : étant donné $(\boldsymbol{x}_i, d_i)_{i=1,2,...N}$ un ensemble de N données à apprendre où d_i est l'étiquette de la classe, comment calculer une fonction de classification $y(\boldsymbol{x})$ qui classe correctement les signaux de force.

La machine à vecteurs de support (Support Vector Machine SVM) est maintenant bien établie dans l'état de l'art avec de solides fondements théoriques pour la classification et la régression [Vapnik 99, Burges 98]. Toutefois la nouvelle approche par machine à vecteur de pertinence (RVM) est réellement prometteuse et mieux adaptée à notre problème. En effet, RVM présente quelques avantages par rapport à SVM [Tipping 01]. Le plus important pour notre application est que RVM utilise un modèle plus clairsemé qui simplifie et accélère la prédiction en-ligne, même s'il nécessite un apprentissage plus long, mais ce dernier est fait hors-ligne.

Nous allons regarder comment le modèle obtenu par RVM est plus clairsemé. Pour clarifier la présentation, nous choisissons un cas simple de classification entre deux états $d_i \in \{0,1\}$. Considérons un nouveau vecteur d'entré $\boldsymbol{x}^*$, la probabilité de sa classe est donnée par :

$$p(d|\boldsymbol{x}^*) = \frac{1}{1+exp(-y(\boldsymbol{x}^*))} \tag{E.41}$$

La fonction y de classification de RVM est donnée par :

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} w_i K(\boldsymbol{x}^*, \boldsymbol{x}_i) \tag{E.42}$$

où $K(\cdot,\cdot)$ est la fonction noyau et $\boldsymbol{x}_i (i = 1,2,3...,N)$ sont les N données à apprendre. Comme c'est présenté dans [Tipping 01], les paramètres w_i de E.42 sont déterminé par une estimation bayesienne pour les concentrer autour de zéro. Ensuite ils sont calculés pour maximiser une distribution à postériori des classes d'étiquettes données comme entré. Un

petit nombre de termes de w_i non nuls signifie qu'un très petit nombre de données à apprendre sont utilisé par la fonction de classification définie par E.42, Comme avec SVM, l'astuce du noyau permet d'étendre les fonctions de base de $y(\boldsymbol{x})$. Pour ce travail, nous utilisons une fonction noyau :

$$K(\boldsymbol{x}^*, \boldsymbol{x}) = exp(-\frac{||\boldsymbol{x}^* - \boldsymbol{x}||^2}{2\sigma^2}) \tag{E.43}$$

où l'optimum de $\sigma > 0$ définit la largeur du noyau. Le lecteur peut se référer à [Bishop 06] et [Tipping 01] pour voir la généralisation de RVM à des problèmes à plus de deux classes. RVM a été utilisé avec succès pour diagnostiquer des défauts [Widodo 09], superviser la classification hyper-spectrale [Mianji 11] et pour reconnaître la position 3D d'humains par régression [Agarwal 04]. Les développements relatifs à RVM et sa comparaison avec SVM est présentée en annexe.

E.4 Acquisition de données et résultats

E.4.1 Le protocole expérimental

Nous avons effectués des expérimentations avec plusieurs paires de volontaires et avons fait varier plusieurs paramètres. Nous avons défini trois vitesse qualitatives : lente normale et rapide et les avons expliqué à chaque paire de volontaires. La vitesse normale correspond à un échange naturel, la vitesse lente à l'échange précautionneux de l'objet supposé fragile et précieux et pour la troisième nous avons demandé aux volontaires d'échanger l'objet rapidement.

Nous avons défini quatre orientations différentes pour Bidule, deux verticales, une horizontale et une inclinée. Nous avons réalisé une partie des expérimentations avec une masse additionnelle. Chaque paire de volontaires a réalisé plusieurs fois chaque type d'échange, soit comme donneur soit comme receveur. Chaque enregistrement dure quelques secondes.

Nous cherchons à déterminer l'instant où le donneur doit lâcher l'objet, aussi le reste du temps de l'échange est assimilé à du bruit que nous utilisons avec l'étiquette "noise" pour alimenter l'apprentissage. La troisième classe que nous utilisons est la classe étiquetée "collision", elle correspond à des enregistrements de collisions avec différents éléments de l'environnement : main humaine, table ou autre.

E.4.2 Résultats et interprétations

Tout d'abord, la fenêtre d'observation est fixée à environ 1 seconde avec une fréquence d'acquisition de 1 kHz, soit 2048 échantillons qui correspondent à 1024 échantillons pour la force et 1024 pour les moments. Nous avons sélectionné 213 expériences pour chaque type d'événement. La classe "noise" est extraite des mêmes données par deux méthodes : les périodes où rien ne se passe et des intervalles où le signal de collision ou de prise est

décalé. Au total nous avons construits un ensemble de 639 séquences pour l'apprentissage et un deuxième ensemble de 240 séquences pour tester le modèle.

Le tableau E.1 présente les résultats pour des choix différents des caractéristiques : la carte des énergies relatives, la sous-bande de niveau 1 et la combinaison des deux. Le choix de combiner deux caractéristiques différentes se justifie par un plus petit niveau d'erreur de classification.

Table E.1: Taux d'erreur pour différentes caractéristiques en utilisant RVM avec un noyau RBF pour la classification.

Features	$\boldsymbol{E}$	$\boldsymbol{S}$	$\{\boldsymbol{E}\ \boldsymbol{S}\}$
Error(%)	15.42	21.67	2.5

Pour réduire la taille de la carte d'énergies relatives $\boldsymbol{E}$ et de la sous-bande $\boldsymbol{S}$, nous avons effectué deux études : tout d'abord, nous avons évalué $\boldsymbol{E}$ par le critère LDA de Fisher, puis nous avons fixé $\boldsymbol{E}$ pour évaluer les différents choix possibles pour $\boldsymbol{S}$ en comparant directement les erreurs de classification.

Nous avons d'abord calculé la carte d'énergie issue de la transformée en ondelette par paquet WPT jusqu'au niveau 6, ce qui produit au total 126 caractéristiques de force et autan pour les couples.

Nous avons examiné la séparabilité J de l'ensemble $\boldsymbol{E}$ des caractéristiques de la carte des énergies relatives. La figure E.14 montre les résultats pour les forces F. Plusieurs termes de l'ensemble des caractéristiques domine la mesure de discrimination, ce qui permet de réduire grandement la taille des caractéristiques sans perte notable d'information discriminante de l'ensemble des caractéristiques. Le choix final de l'ensemble des caractéristiques est fait en combinant les résultats les plus discriminant des résultats de WPT pour les forces et les couples.

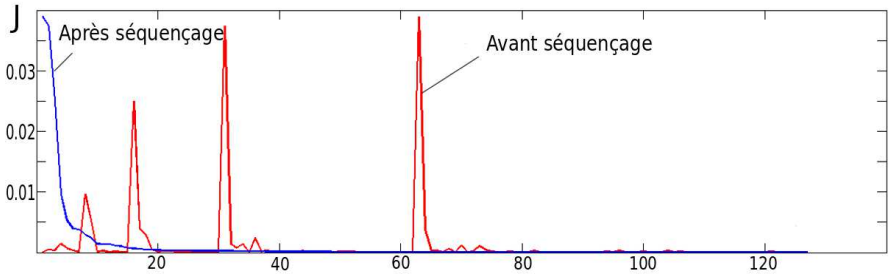


Figure E.14: La mesure de la séparabilité des caractéristiques de la carte des énergies relatives $\boldsymbol{E}$ pour les forces. J est calculé avec (4.13) au travers de l'ensemble des caractéristiques de $\boldsymbol{E}$. Dans ces mesures, les zéros sont essentiellement dû à la propriété de rareté des éléments non nuls de l'arbre WPT. Des résultats analogues sont obtenus avec les couples.

Le choix du niveau à partir duquel est extrait la sous-bande de basse fréquence utilise RVM avec un noyau RBF en comparant directement le taux d'erreur de classification. Pour

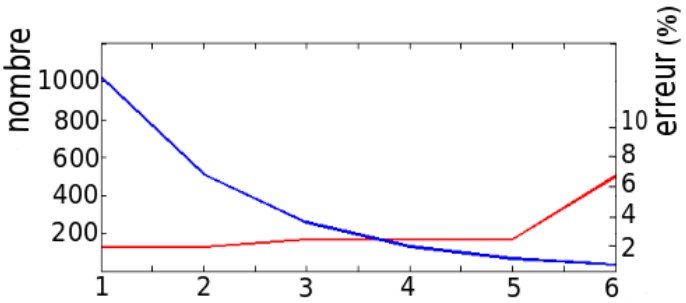


Figure E.15: Le compromis entre le nombre de caractéristiques (courbe bleue) et l'erreur de classification (courbe rouge). Plus petit est le nombre de caractéristiques et plus rapide est la fonction de prédiction en ligne.

cette étude la carte d'énergie est réduite aux 13 caractéristiques les plus discriminantes. Nous choisissons d'abord $\boldsymbol{S}$ au premier niveau et évaluons l'erreur de classification. Ensuite, nous faisons l'évaluation au niveau 2 et ainsi de suite pour chaque niveau jusqu'au 6 ou l'erreur augmente significativement. Ceci s'explique par la perte de précision pour l'information temporelle lorsque on monte dans les niveaux de l'arbre WPT.

Table E.2: Pour cette application, RVM produit des résultats beaucoup plus clairsemés que SVM pour une précision de classification comparable.

	Dimension	Précision(%)
SVM	151	97.92
RVM	9	97.5

A l'aide de cette étude, nous avons finalement sélectionné 13 caractéristiques parmi 252 de la carte des énergies relatives $\boldsymbol{E}$ et fixé le niveau de la sous-bande à 5 avec 64 caractéristiques. Après réduction de la dimension, nous avons obtenu un taux de réussite de 97.5% avec un modèle compact de dimension 9 (les poids non nuls de RVM) et un ensemble de 77 caractéristiques. Lorsque l'arbre WPT complet est utilisé directement pour la phase RVM d'apprentissage, l'apprentissage ne converge pas à cause de la variabilité des classes, du trop petit nombre de données utilisées pour l'apprentissage et de la taille de 12288 pour l'arbre WPT décomposé jusqu'au niveau 6 (chaque niveau comporte 1024 éléments pour les forces et autant pour les couples).

Lorsque on utilise le même ensemble de caractéristiques avec SVM, la précision de la classification obtenue est similaire. Comme le montre la table E.2, RVM produit un modèle beaucoup plus clairsemé, ce qui accélère la classification. Dans cette table, la dimension correspond au nombre de *v*ecteurs supports ou de *v*ecteurs de pertinence. Comme notre objectif est d'effectuer une classification en ligne avec une vitesse comparable à celle du contrôleur qui effectue d'autres calculs complexes, la réduction de la complexité du modèle est importante. Pour la classification en ligne, seules les caractéristiques utilisées doivent être calculées et pas le calcul de l'arbre WPT complet. Pour SVM, les résultats ont été

obtenus en utilisant *libsvm* [Chang 11] (noyau RBF).

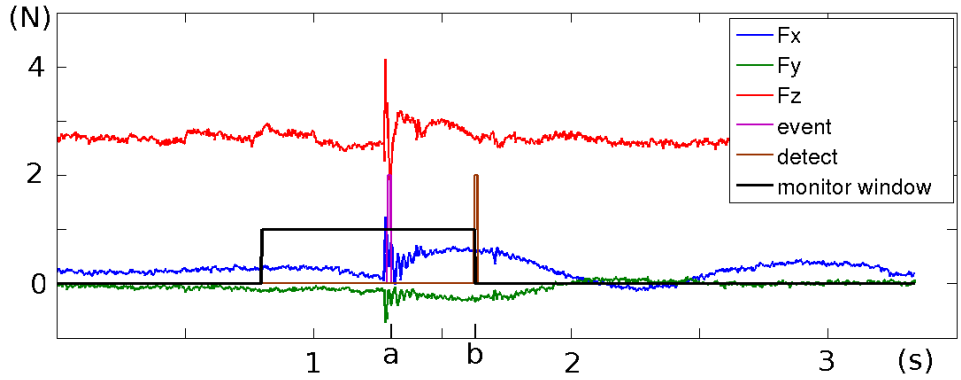


Figure E.16: Fonctionnement du classificateur à partir de signaux de force. Le délai entre l'instant 'a' où l'objet est saisi (événement "event") et l'instant 'b' où le robot réagit (événement "detect") doit se trouver à l'intérieur de la fenêtre d'observation (monitor window) qui est d'une seconde ici.

Le classificateur est implémenté sur le robot pour détecter des événements à partir des efforts mesurés au niveau du poignet. Il utilise le modèle appris des expériences réalisées par les volontaires. Il fonctionne à 50 Hz sur le robot en parallèle avec d'autres éléments logiciels tel que perception et planification. La figure E.16 montre le fonctionnement du classificateur. Pour cette expérimentation, l'objet est tendu par le robot à une personne qui attrape l'objet, mais ne le saisit pas fermement, elle ne fait que le toucher. La figure montre le délai entre la prise et l'instant de réaction du robot (ici le robot n'ouvre pas la pince) par la détection basée sur le signal de force. Pour ce type d'échange, le délai est plus petit qu'une demi seconde. Un aspect important à noter est que l'objectif n'est pas de réagir le plus vite possible, mais de réagir au bon moment et à bon escient. Le délai est aussi appris par le modèle à partir des cas acquis lors des échanges humain/humain. Typiquement, la réaction à une collision est beaucoup plus rapide que la réaction à une prise, mais ces réactions sont automatiquement apprises à partir des cas d'apprentissage.

De premiers tests de manipulation entre le robot et une personne ont été effectués et ont montré des résultats très prometteurs. La figure E.17 montre un exemple d'échange entre le robot et un humain où l'on voit comment le logiciel est utilisé depuis le signal jusqu'à la classification pour réaliser une tâche de manipulation. Lors de cette manipulation, le robot a décidé d'ouvrir la pince. Il apparaît intéressant à ce niveau de réaliser une enquête auprès d'utilisateurs pour évaluer cette approche et voir comment améliorer encore la qualité des échanges pour les utilisateurs humains. Cette étude pourrait intégrer de nombreux aspects comme les préférences humaines concernant la vitesse de réaction du robot ou la comparaison de l'influence sur les utilisateurs des faux négatifs et des faux positifs pour la détection des prises.

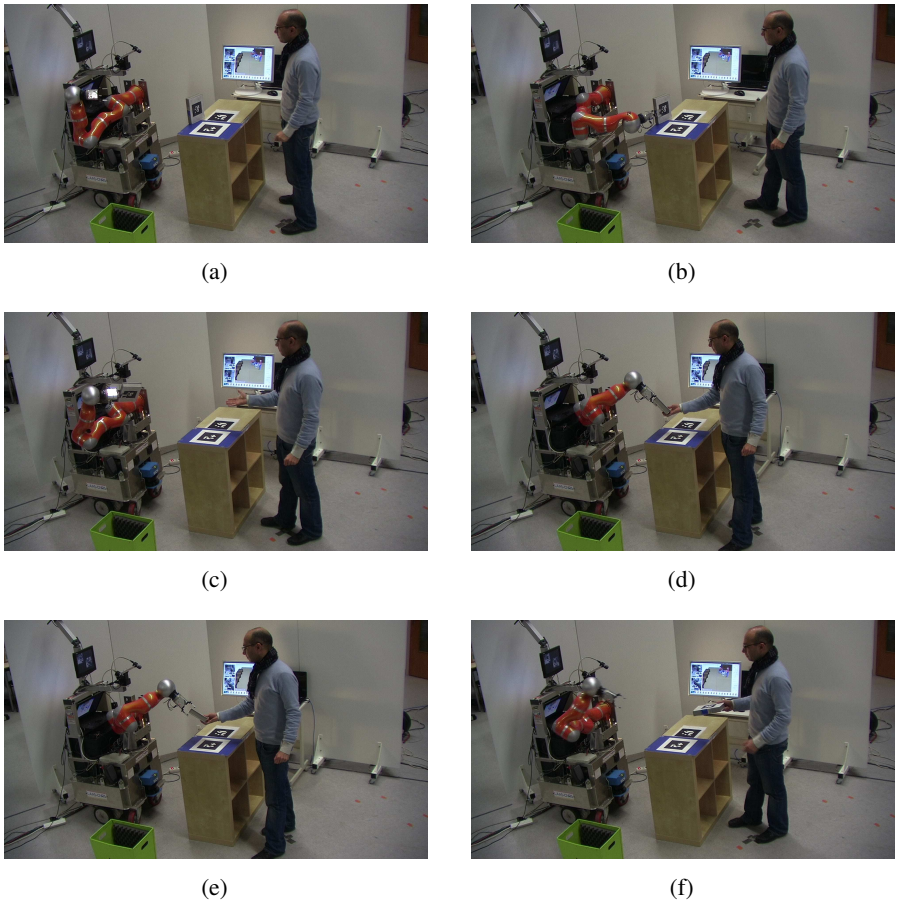


Figure E.17: Test du logiciel intégré pendant une manipulation. (a): le robot Jido, un partenaire humain et un objet à manipuler. (b): Le robot décide de prendre l'objet sur la table. (c): Le partenaire humain demande l'objet. (d): Le robot tend l'objet à l'humain qui le prend. (e): Le robot détecte que son partenaire à saisi l'objet et ouvre la pince. (f): l'échange d'objet est terminé avec succès.

E.4.3 Conclusion

Nous avons proposé un objet pour enseigner au robot comment échanger des objets comme les humains et obtenu des résultats très prometteurs avec l'obtention d'informations très riches. Nous avons étudié la synchronisation de la fermeture et de l'ouverture des préhenseurs. Le choix de la transformation en ondelette par paquet, puis le calcul de la carte des énergies et d'une sous-bande pour extraire des caractéristiques et la sélection de la machine à vecteur de pertinence RVM, nous ont permis de réduire l'ensemble des caractéristiques. Nous avons aussi choisi RVM pour effectuer la classification car elle produit un modèle clairsemé. Ceci nous a permis d'obtenir quelques résultats pour sélectionner les caractéristiques et pour effectuer la classification. On peut aussi remarquer que la construction d'un classificateur fournit un système de contrôle robuste qui est obtenu sans réglage de seuil manuel empirique qui deviendrait particulièrement difficile si on augmente le nombre de classes à apprendre.

Finalement nous voulons mentionner que la détection de collision à partir du capteur de force et couple peut aussi être utilisé sur un robot pour poser proprement un objet sur une table. Par exemple, le contact de l'objet avec la table doit produire des motifs de force différents en fonction de l'élément qui touche la table. Avec les méthodes proposées dans ce chapitre pour construire des modèles compacts et efficaces, nous pouvons conclure que la classification en ligne peut améliorer de nombreux aspects de la manipulation robotisée.

E.5 Contrôleur de trajectoire réactif

E.5.1 Génération de trajectoire en ligne

Les trajectoires sont des fonctions du temps définies dans un espace géométrique qui peut être l'espace cartésien ou l'espace articulaire du robot. Le livre de Biagiotti [Biagiotti 08] et celui de Kroëger [Kröger 10a] en présentent les fondements. Les générateurs de trajectoire que nous utilisons ici ont été présenté dans [Broquere 08c] and [Broquère 10], nous ne reprenons pas les détails dans ce résumé.

E.5.2 Primitives de contrôle

Pour interagir avec un humain, le robot doit effectuer des tâches variées comme prendre un objet, donner un objet à l'humain ou prendre un objet tendu par l'humain. Pour réaliser chacune de ces tâches, le robot planifie un chemin, puis le transforme en trajectoire. Le contrôleur que nous présentons ici prend directement la trajectoire en entré et la découpe en segments sur la base de carte de coûts.

La figure E.18 montre les repères nécessaires pour définir la tâche. La trajectoire $\mathscr{T}_m$ définie le mouvement qui permet au robot de réaliser la tâche de saisi de l'objet tendu par l'humain.

À partir de coûts associés à chaque point de la trajectoire, la trajectoire est divisée en segments associés chacun à une stratégie de contrôle. Les cartes de coûts utilisées sont de différents types : une carte représente le risque de collision qui est calculé à partir de la distance aux obstacles les plus proches de la trajectoire. Une carte représente les domaines visibles et atteignables par l'humain [Sisbot 11] et une carte représente la sécurité et le confort de l'humain. Nous présenterons plus loin deux exemples de carte de coûts. Par exemple, lorsque le risque de collision avec la base du robot est grand, la trajectoire peut être contrôlée dans le repère de base du robot. De même, dans le cas ou l'humain tend un objet au robot, la saisie doit être contrôlée dans le repère de l'objet. [Sidobre 12] détaille d'autres aspects de l'utilisation des cartes de coûts pour planifier des tâches.

Pour simplifier la présentation, dans la suite du document nous considérons la tâche de manipulation relative à un humain donnant l'objet au robot. Pendant la manipulation, l'humain bouge et plusieurs repères sont utilisés pour définir la tâche. A partir de

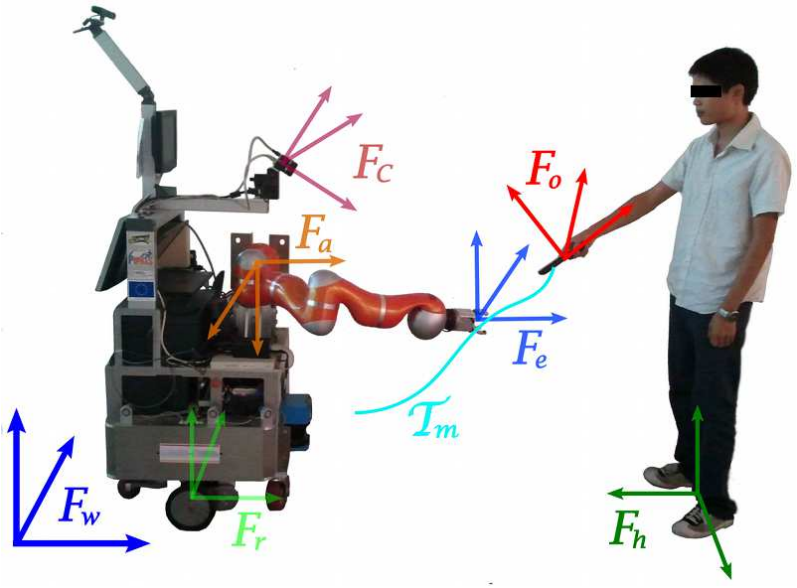


Figure E.18: Les repères qui permettent de définir la tâche de manipulation : F_w: repère de la scène; F_r: repère du robot; F_c: repère de la caméra; F_e: repère de l'organe terminal; F_o: repère de l'objet; F_h: repère de l'humain. La trajectoire qui définit la tâche de manipulation sera contrôlée dans différents repères.

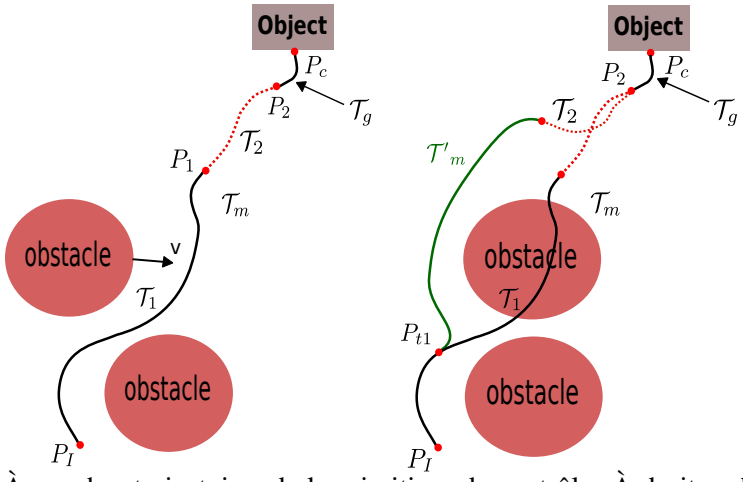


Figure E.19: À gauche: trajectoires de la primitives de contrôle. À droite: changement de la trajectoire du contrôleur suite au déplacement d'un obstacle.

l'évolution des coûts, nous divisons la trajectoire $\mathcal{T}_m$ de la figure E.18 en trois segments comme l'illustre le partie gauche de la figure E.19. Dans cette figure, les points rouges représentent les transitions entre les segments de trajectoire. Le premier segment $\mathcal{T}_1$ qui est défini dans le repère du robot est associé à un risque de collision élevé avec l'environnement, mais n'est pas influencé par les mouvements de l'humain. Le segment $\mathcal{T}_2$ a un coûts associé au risque de collision faible, aussi la modification de la trajectoire n'augmente pas le risque de collision. Le dernier segment de trajectoire $\mathcal{T}_g$ qui va attraper l'objet a un coût associé au risque de collision élevé. Pour garantir le succès de la prise sans collision, ce segment de

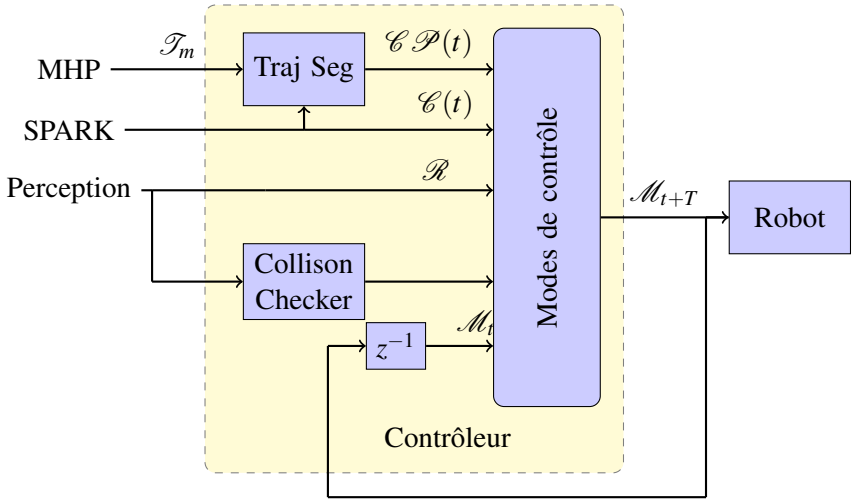


Figure E.20: Éléments d'entré et de sortie du contrôleur. $\mathscr{T}_m$ est la trajectoire calculée par le planificateur MHP qui prend en compte les humains, cette trajectoire est segmenté en primitives de contrôle ($\mathscr{CP}(t)$) par le module "Traj Seg". $\mathscr{C}(t)$ représente les valeurs de coût. $\mathscr{R}$ définit la matrice de transformation qui donne la position de la cible dans le repère du robot. $\mathscr{M}_t$ est l'état courant du robot. $\mathscr{M}_{t+T}$ représente la consigne d'entré pour le prochain cycle de contrôle. z^{-1} représente la duré du cycle de contrôle.

trajectoire doit être contrôlé dans le repère mobile de l'objet.

Nous appelons *repère de la tâche*, le repère dans lequel la trajectoire doit être contrôlée. Nous définissons une *primitive de contrôle* $\mathscr{CP}$ la combinaison de cinq éléments : un segment de trajectoire, une fonction de coût, un repère de la tâche, un mode de contrôle et une condition d'arrêt.

$$\mathscr{CP}(t) = (\mathscr{T}_{seg}(t),\ \mathscr{C}(t),\ \mathscr{F},\ \mathscr{O},\ \mathscr{S})^T \tag{E.44}$$

$\mathscr{C}(t)$ représente la valeur de coût calculée par le module SPARK et associé au coût surveillé pendant l'exécution de la primitive de contrôle. $\mathscr{F}$ définit le repère de la tâche et $\mathscr{O}$ définit le mode de contrôle que nous allons définir dans le paragraphe suivant.. $\mathscr{S}$ définit la condition d'arrêt de la primitive de contrôle. Par exemple, le mouvement de prise comprends cinq éléments : le segment de trajectoire $\mathscr{T}_g$, la valeur $\mathscr{C}(t)$ du coût du risque de collision, le repère de la tâche $\mathscr{F}_o$, le mode de contrôle de la trajectoire et la condition d'arrêt $\mathscr{S}$ qui est un seuil prédéfini pour la distance entre l'organe terminal et la fin de la trajectoire $\mathscr{T}_g$. Dans la littérature, les concepts de primitive de manipulation ou de "skill primitive" sont souvent utilisées pour désigner un niveau intermédiaire entre un planificateur et un contrôleur, de nombreux travaux les ont étudiées comme par exemple [Kröger 11].

La figure E.20 utilise les définitions de primitive de contrôle ($\mathscr{CP}(t)$) et de condition de mouvement $M(t) = (X(t), V(t), A(t))$ pour présenter les différents éléments du contrôleur

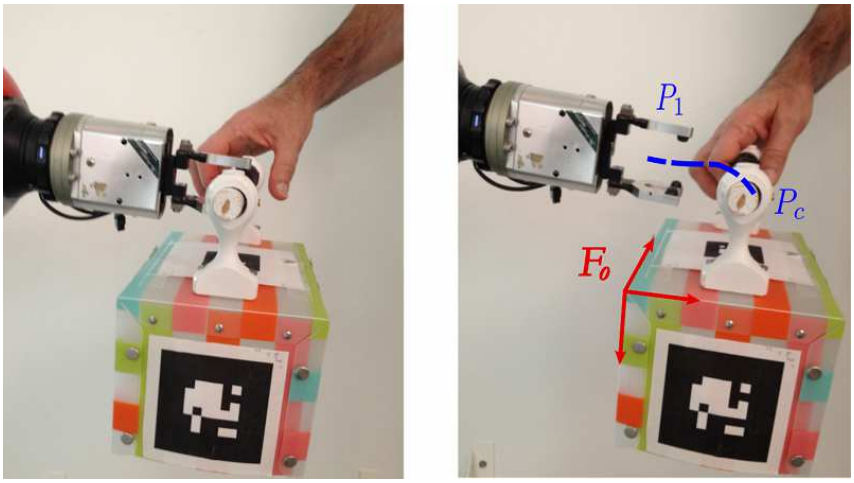


Figure E.21: Un cas de saisie simple. À gauche: la prise planifiée définit les points de contact entre la pince et l'objet. À droite: pour terminer la prise, le manipulateur doit suivre la trajectoire bleu P_1 - P_c, ant ensuite fermer la pince. Ce mouvement doit être contrôlé dans le repère de l'objet F_o.

de trajectoire, ses entrés et ses sorties. La trajectoire initiale $\mathscr{T}_m$ est segmentée en une suite de segments $\mathscr{C}\mathscr{P}(t)$. Les valeurs de coûts $\mathscr{C}(t)$ sont utilisées pendant la segmentation et pour surveiller l'exécution de la primitive de contrôle. Un détecteur de collision intègre les données provenant de la vision, de la perception de l'humain et des codeurs du robot pour limiter le risque de collision en réduisant la vitesse ou en suspendant l'exécution de la tâche. A partir de ces données et de l'état courant du robot M_t, plusieurs modes de contrôle peuvent être utilisé pour calculer la condition de mouvement pour le prochain cycle de contrôle, laquelle est envoyé au servo-moteurs.

La dernier segment de la primitive de contrôle est présenté sur la figure E.21 où le robot saisi l'objet. Un planificateur de prise à choisi une prise qui a été utilisée pour planifier la trajectoire finale. Le planificateur de prise est présenté plus en détail dans [Bounab 08] et [Saut 12]. Lorsque l'objet bouge, le repère de l'objet F_o et le chemin de la trajectoire bougent aussi. Aussi, pour éviter les collisions, la trajectoire de cette primitive de contrôle doit être contrôlée dans le repère de l'objet F_o.

E.5.3 Contrôleur de trajectoire réactif

Au niveau du contrôleur, la tâche est définie par une suite de primitives de contrôle. Chaque primitive de contrôle est définie par un quintuplet. Le premier niveau du contrôleur de trajectoire que nous proposons est une machine d'état qui surveille l'exécution, contrôle la succession des modes de contrôle et gère le risque de collision et les autres situations particulières. Après la présentation de la machine d'état, nous présentons les modes de contrôle et notamment le suivi de trajectoire et le suivi de cible.

E.5.3.1 Machine à état

La machine à état contrôle la commutation entre les différents modes de contrôle associés à chaque primitive et surveille l'exécution. À cause de la présence des humains, l'environnement du robot évolue au cours du temps et la tâche de contrôle doit s'adapter. La machine à état peut aussi suspendre ou arrêter la primitive de contrôle comme présenté sur la figure E.22.

suspension : lorsque le vision est en défaut, lorsque la cible est inaccessible ou lorsque une activité spécifique de l'humain induit un coût trop élevé, le contrôleur doit suspendre la tâche.

arrêt : quel que soit le mode de contrôle choisi, des collisions imprévisibles peuvent se produire en nécessiter l'arrêt du robot. Notre contrôleur utilise deux modules pour détecter ces collisions.

Le premier est un détecteur de collision géométrique basé sur les travaux de Larsen et al. [Larsen 99]. Il actualise un modèle 3D de la scène et vérifie les collisions à la fréquence du contrôleur. Basé sur la géométrie, il peut prédire les collisions avant qu'elles ne se produisent.

Le second, basé sur [De Luca 08] surveille les couples externes. La méthode a été conçue pour détecter les collisions physiques inattendues entre le robot et les obstacles. La détection rapide des collisions est obtenue par une méthode basée sur les couples moteurs et ne nécessite aucun capteur externe. Ce détecteur apporte une garanti de sécurité dans le contexte de l'interaction entre humain et robot car le robot s'arrête dès qu'un couple inattendu est détecté.

Ralentissement sur la trajectoire: Basé sur une fonction de coût, le contrôleur peut diminuer la vitesse de la trajectoire en modifiant la loi d'évolution temporelle $s(t)$. Si un mouvement rapide risque d'effrayer une personne proche du robot, à partir des fonctions de coût le robot modifie $s(t)$ pour diminuer la vitesse. À partir d'un modèle géométrique du robot et de l'humain mis à jour périodiquement, le robot calcule un coût. Ce coût représente une moyenne pondérée du coût de sécurité et du coût de visibilité introduit précédemment.

Chaque contrôleur de trajectoire élémentaire incorpore une machine à état.

E.5.4 Modes de contrôle de trajectoire

En fonction du contexte défini par la primitive de contrôle, la stratégie de contrôle doit être différente. La génération de trajectoire en ligne fournit un cadre flexible pour construire ces stratégies de contrôle qui ont pour principale caractéristique la réactivité aux aux événements imprévus. Cette réactivité peut consister à adapter les paramètres cinématiques, par exemple modifier la vitesse pour l'adapter aux nouvelles conditions. La commutation vers une nouvelle trajectoire ou la modification du repère dans lequel la trajectoire est contrôlée constitue d'autres adaptations possibles.

L'idée principale du contrôleur de trajectoire est de calculer, à chaque itération, une trajectoire rejoignant la trajectoire ou la cible à partir de l'état courant. Nous allons maintenant

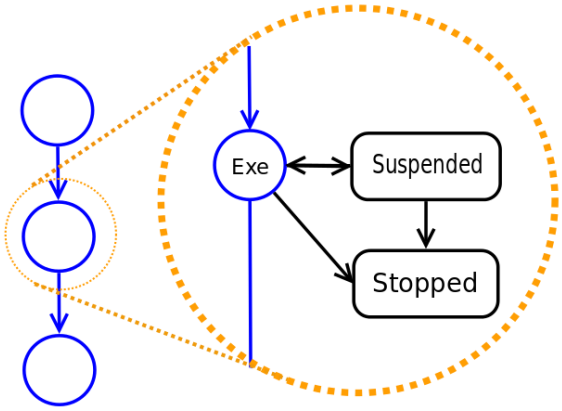


Figure E.22: Chaque cercle de gauche représente la machine à état du contrôleur d'une primitive de contrôle. Le zoom montre que chacune de ces machines à état peut suspendre ou arrêter la tâche.

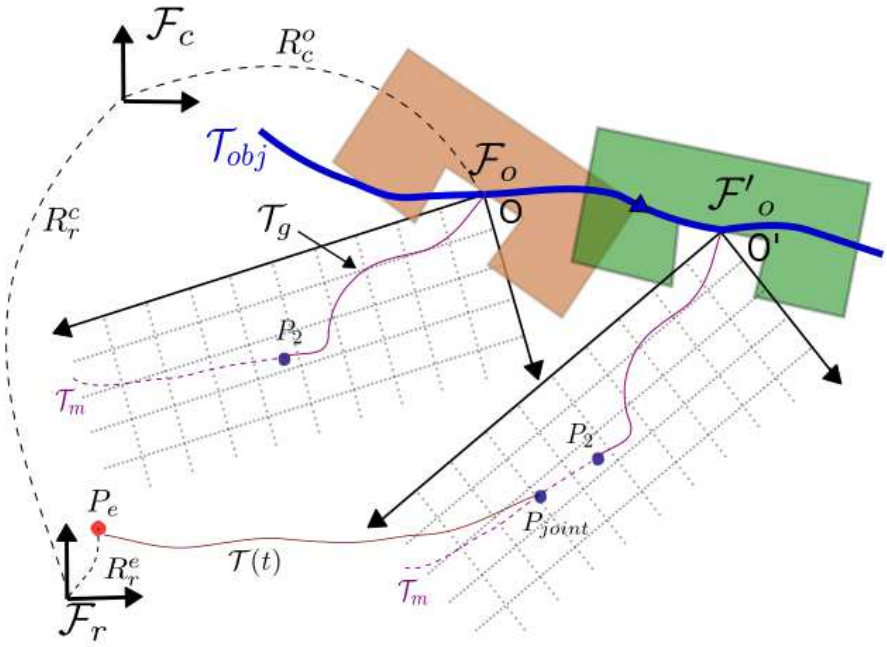
définir plusieurs modes de contrôle pour répondre à ces différents besoins de réactivité pour la manipulation interactive.

Mode de contrôle 1: suivi de cible Si l'on suppose que le robot est dans une zone sans risque de collision, le système peut suivre le dernier point de la trajectoire. Dans ce cas le contrôleur génère itérativement une trajectoire pour rejoindre le point final et envoie le premier pas de cette trajectoire au contrôleur de bas niveau. Dans le cas particulier où ce contrôleur utilise un retour visuel pour faire le suivi de cible, il fait de l'asservissement visuel.

La figure E.23 montre les détails du mode de contrôle de trajectoire pour le *suivi de cible*. L'objet est à la position O au temps courant t_1 et se déplace suivant la courbe $\mathscr{T}_{obj}$.

Cette courbe est obtenu par un simple filtre de Kalman pour construire un modèle de mouvement issu du système de vision 3D. $\mathscr{F}_r$ représente le repère du robot, $\mathscr{F}_c$ et $\mathscr{F}_o$ représentent respectivement le repère de la caméra et de l'objet. De même, R_r^c est la matrice de transformation 4×4 de $\mathscr{F}_r$ vers $\mathscr{F}_c$ et R_c^o la matrice de transformation de $\mathscr{F}_c$ vers $\mathscr{F}_o$. Ces liens sont tous représenté en ligne pointillée et évoluent dans le temps avec le mouvement de l'objet ou de l'humain. La direction de la caméra est ajustée pour centrer l'objet dans l'image. Initialement le robot est au point P_e, Comme il n'y a pas de risque de collision, le contrôleur peut simplement suivre le point P_2 qui est le point terminal du segment de trajectoire. Il est aussi possible pour le robot de rejoindre la trajectoire en un autre point P_{joint} défini dans le repère de la tâche qui est ici le repère de l'objet.

Mode de contrôle 2: suivi de trajectoire dans le repère de la tâche. Une fois que le robot à atteint le point P_2, il commence le mouvement de prise qui correspond à la trajectoire $\mathscr{T}_g$ sur les figures E.19 et E.24. L'objet continue de se déplacer, mais comme le robot se trouve



point

Figure E.23: *Mode de contrôle 1.* Le robot suit P_2. L'objet se déplace vers la droite, il est représenté à deux instants : d'abord en marron au temps t_1 puis en vert au temps t_2. Dans les deux cas le point d'entré P_2 de la trajectoire $\mathcal{T}_g$ est représenté relativement au repère $\mathcal{F}_o$.

maintenant dans une zone de coût élevé, il doit suivre la trajectoire dans le repère de la tâche.

La figure E.24 montre les détails du mode de contrôle. Les repères et le déplacement de l'objet sont identiques à la figure E.23, mais le robot est maintenant au point P'_e. Le robot suit la trajectoire $\mathcal{T}_g$ dans le repère de l'objet $\mathcal{F}_o$ et au final aura suivi la trajectoire $\mathcal{T}'(t)$ dans le repère du robot.

Mode de contrôle 3: Re-planification et changement de trajectoire : pendant l'exécution, le chemin peut être re-planifié pour, par exemple, éviter un obstacle qui s'est déplacé (voir figure E.19). Lorsque une nouvelle trajectoire est produite par le planificateur MHP, elle est envoyée au contrôleur qui commute vers cette nouvelle trajectoire. Lorsque, pendant que le contrôleur assure le suivi d'une trajectoire $\mathcal{T}_m$, un obstacle se déplace et invalide la trajectoire initiale, le contrôleur du robot peut suspendre le suivi de la trajectoire puis commuter sur la nouvelle trajectoire $\mathcal{T}'_m$ produite par MHP. Cette trajectoire $\mathcal{T}'_m$ commence à l'instant t_1 dans le futur, aussi le contrôleur peut anticiper la commutation et lorsque le robot atteint le point P_{t_1} à l'instant t_1, le contrôleur commute sur la nouvelle trajectoire $\mathcal{T}'_m$. Comme cette nouvelle trajectoire $\mathcal{T}'_m$ a été calculée à partir de l'état prédit du robot à l'instant t_1, la commutation de trajectoire ne pose aucun problème. De plus, la boucle du contrôleur, en calculant à chaque cycle une trajectoire pour rejoindre la trajectoire de consigne, peut accepter des erreurs de commutation assez importantes.

Dans ce paragraphe nous avons étudié principalement une tâche de saisi d'un objet

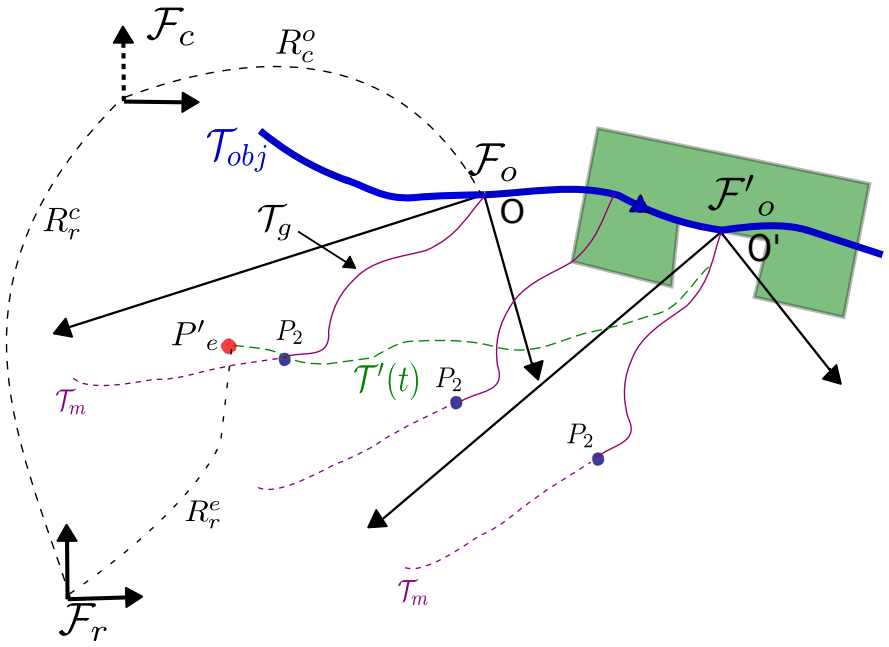


Figure E.24: *Mode de contrôle 2.* L'objet est représenté en marron à l'instant t_1 et coloré en vert à l'instant t_2. il suit le mouvement représenté par la trajectoire bleu $\mathcal{T}_{obj}$. La trajectoire de prise $\mathcal{T}_g$ est représenté en violet, elle reste invariante dans le repère de l'objet $\mathcal{F}_o$. Le robot suit la trajectoire $\mathcal{T}_g$ pour réaliser le mouvement de prise.

mobile tendu par un partenaire humain. Les mêmes fonctionnalités peuvent être utilisées pour d'autres tâches, comme prendre un objet, donner un objet à un humain ou poser un objet sur la table. Par exemple, poser un objet sur une plate-forme mobile peut être réalisé en contrôlant le dernier segment de trajectoire dans un repère lié à la plate-forme qui est mobile par rapport au repère du robot. De même, pour tendre un objet à la main mobile d'un humain, le robot doit suivre une trajectoire planifiée par le planificateur MHP qui tient compte des humains. Le suivi de cette trajectoire doit être assuré dans un repère lié à la main mobile. Même si l'algorithme de décomposition d'une tâche en primitives de contrôle n'est qu'une ébauche, le concept de *primitives de contrôle* associées à un mode de contrôle permet de contrôler simplement les tâches interactives de base.

E.5.5 Résults et comparaison

Nous nous focalisons ici sur quelques résultat qui montre comment le contrôleur est intégré dans un manipulateur qui interagit avec les humains. Les performances de générateur de trajectoire sont présentés plus en détail dans [Broquere 08c].

La figure E.25 présente les résultats d'un suivi de cible par le contrôleur pendant 25s. Afin de simplifier, nous ne présentons que l'axe X. La ligne pointillée noire correspond à la position de la cible générée par le système de vision 3D. La ligne rouge correspond à la position du robot. Les deux courbes du bas montrent la vitesse et l'accélération. Tout

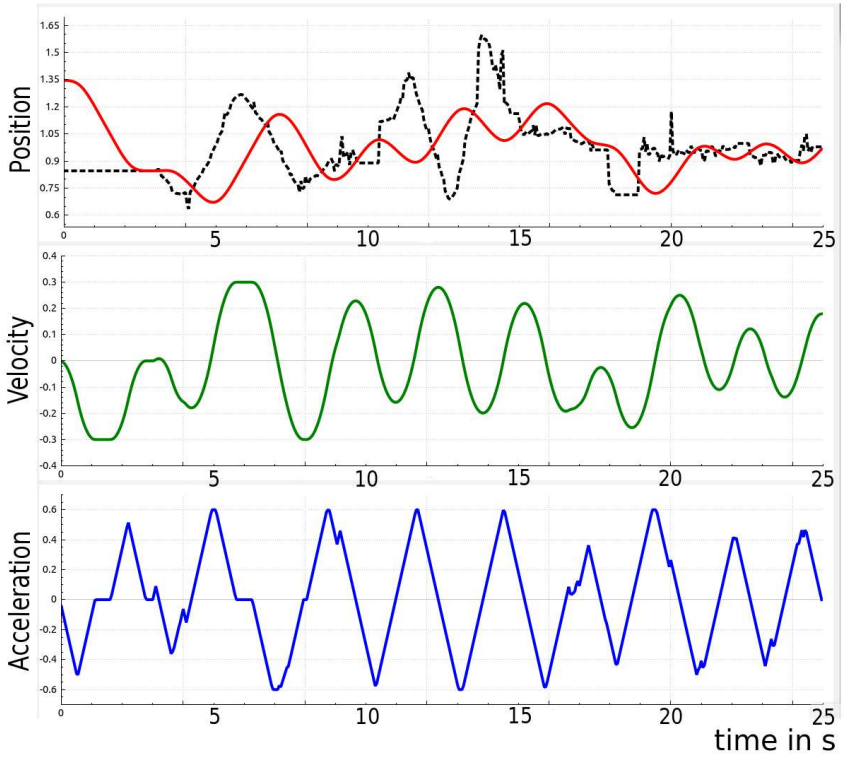


Figure E.25: Résultats d'un suivi de cible : position en m, vitesse en ms^{-1} et accélération en ms^{-2} de l'organe terminal du robot pendant une phase de suivie de 25 s. La ligne pointillée noire correspond à la position de la cible, on peut observer le bruit du système de vision 3D. La ligne rouge correspond à la position du robot qui suit la cible avec un retard. La position du robot est calculée à l'aide du modèle géométrique et des positions articulaires mesurées alors que sa vitesse et son accélération sont obtenues par dérivation. La vitesse, l'accélération étant le jerk sont limités, le mouvement de suivi est souple.

d'abord, nous pouvons remarquer que le contrôleur se comporte de manière robuste en présence de bruit sur le signal d'entrée. D'autre part, la vitesse, l'accélération et le jerk du robot sont bornés car le système calcule des trajectoires de type V.

La figure E.26 montre une tâche complète de manipulation qui illustre le comportement du contrôleur. L'évolution de la position du robot est présentée sur la figure E.27 dans les coordonnées du repère de base du robot (voir figure E.18). Lorsque le robot voit l'objet tendu par l'humain, la tâche de planification calcule une prise, puis un chemin et enfin la trajectoire pour que le robot réalise la tâche.

Le contrôleur divise la trajectoire en trois segments et choisit un mode de contrôle pour construire une primitive de contrôle pour chaque segment. Le premier segment est un suivi de trajectoire définie dans le repère du robot, le second définit un suivi du point d'entré de la trajectoire du troisième et le dernier segment définit un suivi de trajectoire défini dans le repère de l'objet.

Pendant le suivi de cible, l'humain est distrait par l'arrivée d'un second humain et prend un moment pour donner un deuxième objet à cette personne. Le logiciel détecte cet

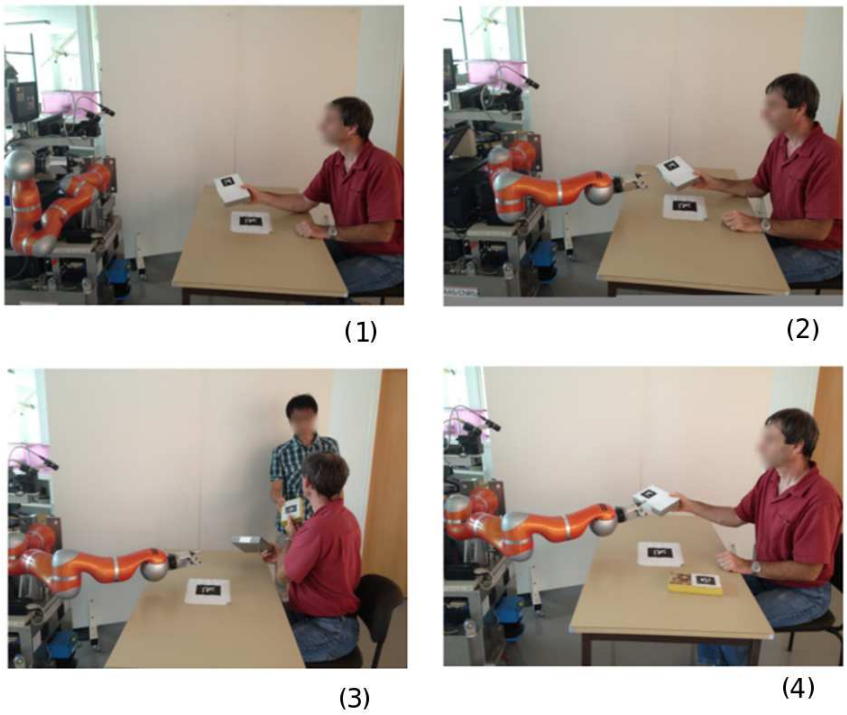


Figure E.26: (1): Le contrôleur doit assurer la saisie de l'objet tendu par un humain. Il suit le premier segment de trajectoire dans le repère du robot. (2): L'objet de déplace et le robot suit la cible. (3): Un deuxième humain distrait l'humain qui tend l'objet, et le contrôleur suspend la tâche. (4): L'humain reprend la tâche et tend à nouveau la cassette au robot qui reprend lui aussi la tâche et attrape l'objet.

événement en surveillant le coût de la carte de visibilité de l'humain. Suite à cet événement, le contrôleur suspend la tâche. Lorsque l'humain regarde à nouveau le robot et qu'il ramène l'objet dans la zone de l'échange, le contrôleur reprend le suivi, puis termine la tâche.

La performance du suivi de cible peut être observé entre les instants *b* et *c* d'une part et *d* et *e* d'autre part. Le contrôleur a accompli la tâche sans que le planificateur ait dû re-planifier la tâche ou la trajectoire. Ces résultats montrent la réactivité des contrôleurs basés sur la génération de trajectoire en ligne et en particulier la réactivité vis à vis des humains. Ce type de contrôleur facilite l'interaction avec les humains. Avant la réalisation de ce contrôleur, l'humain devait tendre l'objet sans bouger pour que le robot le saisisse, maintenant le robot peut réaliser la tâche même si l'humain bouge sa main pendant l'échange.

E.5.6 Conclusion

Nous avons présenté un contrôleur de trajectoire réactif et montré quelques résultats relatifs à la saisie par le robot d'un objet tendu par un humain.

Les premiers résultats présentés illustrent la polyvalence des contrôleurs basés sur la génération de la trajectoire en ligne. Dans l'exemple présenté ici, le contrôleur change de repère et suspend la tâche pendant que l'humain est distrait.

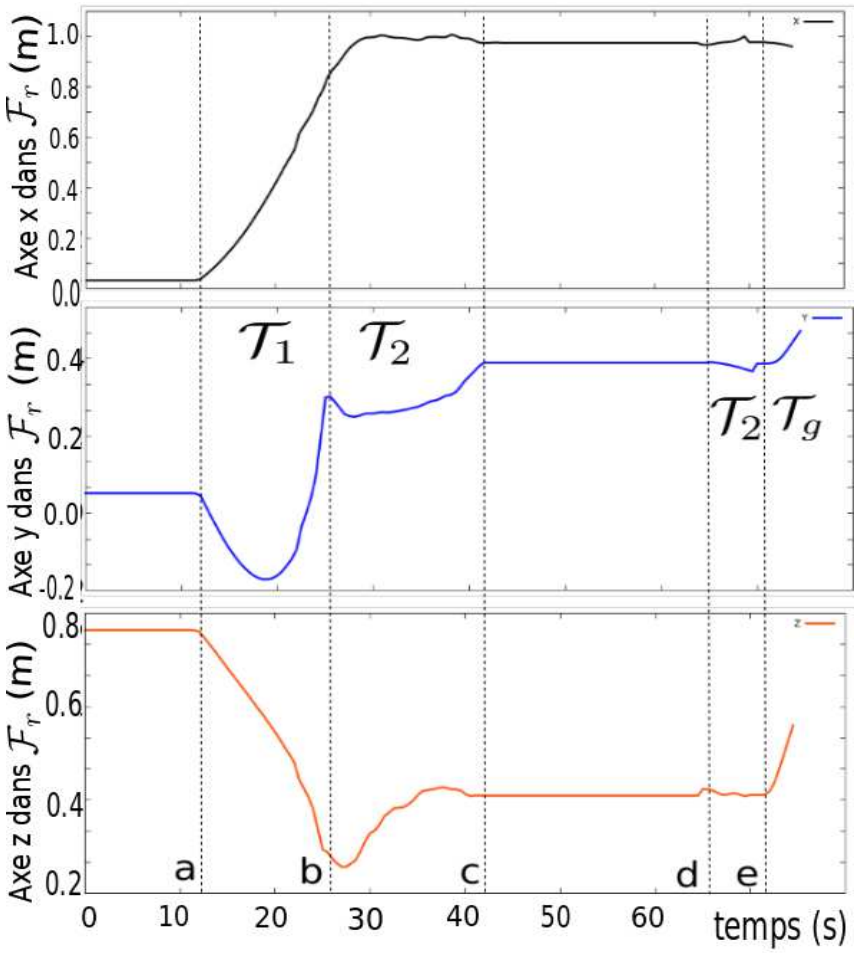


Figure E.27: Mouvement de l'organe terminal du robot dans le repère du robot. Le mouvement commence à l'instant *a*. De *a* à *b*, le contrôleur suit le premier segment de trajectoire $\mathcal{T}_1$ dans $\mathcal{F}_r$. De *b* à *c* et de *d* à *e*: le contrôleur réalise un suivi de cible. De *c* à *d*: le contrôleur suspend la tâche. De *e* à la fin: le mouvement de saisi est contrôlé dans le repère $\mathcal{F}_o$.

Le contrôleur de trajectoire proposée utilise un générateur de trajectoire en ligne pour construire une trajectoire qui rejoint la trajectoire à suivre. Il est très simple à utiliser et à mettre en œuvre et fournit une solution efficace pour suivre des trajectoires ou des objets en mouvement dans le contexte de l'interaction entre humains et robots. Plus précisément, il peut adapter les limites cinématiques à l'état évolutif de la scène et commuter entre deux trajectoires ou deux modes de contrôle.

Le défi est maintenant d'étendre ce type de contrôleur de trajectoire et le concept de primitives de contrôle pour gérer les forces et donc les événements produit par la surveillance des forces. De même l'extension au contrôle de manipulateurs mobiles à deux bras soulève de nombreux problèmes intéressants.