UNIVERSIDADE D
COIMBRA

Mohammad Safeea

# SAFE COLLABORATIVE ROBOTIC MANIPULATORS

**Tese no âmbito do Doutoramento em Engenharia Mecânica, Gestão e Robótica Industrial orientada pelo Professor Doutor Pedro Mariano Simões Neto e pelo Professor Doutor Richard Béarée e apresentada ao Departamento de Engenharia Mecânica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.**

Junho de 2020

# Safe Collaborative Robotic Manipulators

by

Mohammad Safeea

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy under the joint supervision

between

Universidade de Coimbra, Portugal

and

École Nationale Supérieure d'Arts et Métiers, France

Supervisors:

Professor Pedro Neto

Professor Richard Béarée

2020

# Safe Collaborative Robotic Manipulators



by

Mohammad Safeea

## Abstract

Safe Collaborative Robotic Manipulators

by

Mohammad Safeea

Doctorate in Mechanical Engineering

University of Coimbra and École Nationale Supérieure d'Arts et Métiers

Collaborative robotic manipulators are new type of industrial manipulators, that allow the coexistence of humans in the same working area around the robot. Consequently, a human can work safely beside a collaborative manipulator, without having physical barriers or safety fences separating between them. In light of this, we can see that the main features of collaborative robots are centred around two main aspects, safety and collaboration, which are the main topics of this thesis.

On the subject of safe interaction with the robot, we revisit the problem of real-time collision avoidance for robotic manipulators in unstructured and dynamic environments, where humans are working side-by-side with the robot. We propose a method for endowing the manipulator with sensing capabilities, so it can detect the proximity of humans around it. Also, we propose a control algorithm, based on the potential fields method, for providing the robot with human like reflexes, which allows it to avoid collision with people nearby. Afterwards, we propose a method for implementing the collision avoidance controller to modify off-line generated paths (necessary for performing industrial tasks). For testing, we implement the proposed method in a real industrial robotic cell, utilising an industry standard collaborative manipulator (KUKA iiwa) in human centred environment, where a human worker (coworker) and the robot are sharing the same working area.

On the subject of collaboration we study the topic of precise positioning of the end-effector of the robot in an intuitive manner. We present the precision hand-guiding technique, and we present a lightweight algorithm for performing motion in the null space of redundant manipulators while preserving the accuracy of end-effector's pose. Afterwards, we exploit the redundancy for performing secondary tasks, where using the torque feedback at the joints of sensitive robots and the force/torque measurements from an external Force/Torque (FT) sensor at the flange of the robot we were able to perform precise hand-guiding at the end-effector level while exploiting redundancy to perform in-contact obstacle navigation.

For developing the aforementioned applications, and for achieving real-time performance, we implemented various light weight numerical methods. Consequently, several efficient algorithms for performing the calculations and the control have been realized in software packages, most of which are provided as open source libraries in public repositories.

Throughout this work, the proposed algorithms were implemented in various programming languages, while the real-time virtual-reality simulations were carried out using the program V-REP (Virtual Robot Experimentation Platform). Using these tools several controllers, algorithms, techniques and simulations were applied and the results achieved were discussed throughout this document. Experimental tests were carried out using industrial-grade hardware, the collaborative manipulator KUKA iiwa7R800, laser scanner, IMUs, force torque sensor and magnetic trackers.

## Keywords

Collision avoidance, safety, potential fields, Newton method, Hessian, collaborative robots, hand-guiding, redundancy, robot kinematics, robot dynamics, recursive algorithms, Mass matrix, Christoffel symbols.

## Résumé

Des robots manipulateurs collaboratifs sûrs

par

Mohammad Safeea

Doctorat Génie Mécanique

Université de Coimbra et École Nationale Supérieure d'Arts et Métiers

Les robots manipulateurs collaboratifs sont de nouveaux types de manipulateurs dans l'industrie, qui permettent la coexistence homme-robot en partageant le même espace de travail. Par principe, un humain peut travailler au côté d'un robot collaboratif en toute sécurité sans devoir utiliser une barrière physique ou une clôture de sécurité pour les séparer. Actuellement, le développement de ces moyens est encore limité par le niveau de collaboration homme-machine qui se résume généralement à du partage d'espace en mode dégradé. Ainsi, nous proposons dans cette étude plusieurs contributions concernant la sécurité et la collaboration afin d'étendre le champ d'interaction entre l'humain et le robot, et par conséquent la viabilité économique de ce type d'application, dans un contexte industriel.

Concernant l'aspect de l'interaction sécurisée avec le robot, on revisite le problème d'évitement de collisions en temps réel pour des robots manipulateurs dans des environnements dynamiques et non structurés, où des humains et le robot travaillent côte-à-côte. On propose une méthode pour doter le robot manipulateur de capacités de détection, dans le but de repérer les humains autour de lui. On propose également un algorithme de contrôle basé sur la méthode des champs potentiels pour équiper le robot de réflexes comme les humains, qui lui permettrait d'éviter la collision avec les gens aux alentours. Après cela, on présente une méthode d'implantation pratique du contrôle d'évitement de collision permettant la modification hors ligne des chemins générés (nécessaire pour effectuer des tâches industrielles). La méthode proposée est validée experimentalement en exploitant une cellule robotique industrielle, constituée d'un manipulateur collaboratif de type KUKA iiwa dans un environnement principalement humain, où un travailleur humain (collaborateur) et le robot partagent le même espace de travail et potentiellement la même tâche.

Concernant l'aspect de la collaboration homme-robot, nous proposons une première contribution portant sur la thématique du positionnement précis de l'effecteur du robot et ceci de manière intuitive pour l'humain. Ainsi, une nouvelle méthodologie de guidage

manuel (hand-guiding) permettant la maitrise de la précision du guidage à la main est développée et validée expérimentalement. D'autres fonctionnalités innovantes basées sur l'exploitation de la redondance cinématique du robot utilisé sont développées dans ce travail. Une méthode basée sur un algorithme de faible complexité (temps de calcul) est développée afin d'effectuer un mouvement dans le noyau de l'espace redondant des robots tout en préservant la précision de la pose de l'effecteur. Ensuite, la redondance est exploitée pour effectuer des tâches secondaires autorisant un guidage manuel précis de l'effecteur tout en exploitant la redondance, par exemple dans le cas d'un guidage en environnement contraint. Cette dernière méthode utilise les informations de couple de rétroaction au niveau des articulations des robots , ainsi que la mesure de la force et du couple exercée au niveau de l'effecteur par le biais d'un capteur d'effort/couple (F/T) externe.

Le développement des applications précédemment mentionnées a été réalisé en intégrant la contrainte de la faisabilité de calcul temps réel sur un moyen robotisé industriel. Pour ce faire, plusieurs méthodes numériques à faible complexité algorithmique ont été développées est évaluées. Les logiciels et algorithmes exploités proviennent de librairies open source et les développements proposés ont été rendu également accessibles à tous publics.

Tout au long de ce travail, les algorithmes développés ont été codés dans différents langages de programmation, tandis que les simulations de réalité virtuel en temps réel sont réalisée en utilisant le programme V-REP (Virtual Robot Experimentation Platform). Une contribution transverse à cette étude réside dans le fait que tous les algorithmes, techniques et simulations développés sont appliqués expérimentalement sur des moyens industriels (robot manipulateur collaboratif KUKA iiwa7R800, laser scanner, IMUs, capteur d'effort force/couple 6 composantes et traqueurs magnétiques de géolocalisation) et qu'une analyse pratique des contraintes d'implantations est conduite pour chacune des nouvelles fonctionnalités.

### Mots clés

Évitement des collisions, sécurité, champs potentiels, méthode de Newton, Hessian, robots collaboratifs, guidage manuel, redondance, cinématique du robot, dynamique du robot, algorithmes récursifs, matrice de masse, symboles de Christoffel.

## Resumo

Manipuladores robóticos colaborativos seguros

por

Mohammad Safeea

Doutoramento em Engenharia Mecânica

Universidade de Coimbra e École Nationale Supérieure d'Arts et Métiers

Os manipuladores robóticos colaborativos são um novo tipo de manipulador industrial que permite a coexistência de seres humanos na área de trabalho em redor do robô. Consequentemente, um humano pode trabalhar com segurança ao lado de um manipulador colaborativo, partilhando o mesmo espaço e sem barreiras físicas entre eles. As principais características dos robôs colaborativos centram-se em dois aspetos principais, segurança e colaboração, sendo estes os tópicos principais desta tese.

Relativamente à interação segura com manipuladores robóticos colaborativos, revisitamos o problema da prevenção de colisões em tempo real em ambientes não estruturados e dinâmicos, onde humanos e robôs trabalham lado a lado. É proposto um método para dotar o manipulador de recursos de deteção de humanos/obstáculos em seu redor. Além disso, é proposto um algoritmo de controlo baseado no método de campos potenciais que permite ao robô colaborativo evitar colisões com humanos/obstáculos. Posteriormente, propomos um método para o controlador de prevenção de colisões de forma a alterar as trajetórias nominais do robô definidas off-line (necessário para executar tarefas industriais). As metodologias propostas foram implementadas e testadas numa célula robótica real, utilizando um manipulador colaborativo industrial (KUKA iiwa) em tarefas onde robô e humano partilham a mesma área de trabalho.

O tópico da colaboração homem-robô foi estudado considerando o posicionamento preciso do robô de forma intuitiva. É proposto um método de guiamento manual de precisão, assente num algoritmo que permite o movimento do robô no espaço nulo de manipuladores redundantes enquanto preserva a precisão posicional. A redundância é também explorada recorrendo ao uso de medições de torque a partir das articulações do robô e medições de força/torque de um sensor externo de Força/Torque (FT) acoplado à flange do robô. Desta forma conseguimos realizar o guiamento manual do robô ao nível do end-effector enquanto tiramos vantagem da redundância na navegação de obstáculos por contato.

As metodologias acima mencionadas foram implementadas em robôs reais, recorrendo a algoritmos que pela sua natureza de aplicação devem ser computacionalmente leves e eficientes, permitindo assim desempenho em tempo real por parte do sistema

robótico. Neste contexto, vários algoritmos relacionados com a implementação dos métodos, cálculos matemáticos e controlo foram desenvolvidos e agrupados em pacotes de software, a maioria dos quais são fornecidos como bibliotecas de código aberto em repositórios públicos.

Os algoritmos propostos foram implementados em várias linguagens de programação, enquanto as que simulações em tempo real foram realizadas usando o programa V-REP (Virtual Robot Experimentation Platform). Usando essas ferramentas vários controladores e algoritmos foram aplicados, sendo os seus resultados discutidos neste documento. Testes experimentais foram realizados usando hardware de nível industrial, nomeadamente o manipulador colaborativo KUKA iiwa7R800, scanner laser, unidades de medição inerciais (IMUs), sensores de força/torque e rastreadores magnéticos.

**Palavras-chave**

Prevenção de colisões, segurança, campos potenciais, método de Newton, Hessian, robôs colaborativos, guiamento manual, redundância, cinemática, dinâmica, algoritmos recursivos, matriz de massa, símbolos de Christoffel.

ACKNOWLEDGEMENT

I would like to thank my supervisors Professor Pedro Neto, and Professor Richard Bearee for their friendliness, patience, important notes, and the useful guidance that they kindly offered throughout this work. They provided encouragement, and unwavering enthusiasm with my research, their advice and leadership were unequivocally fundamental in the progress of this work. Besides to my supervisors, I would like to thank Dr. Antonio Mourão for his mentorship and advice.

I also express my appreciation to the administrations of both Universities Coimbra and Arts et Métiers, for offering the suitable and rich environment to work in, they provided the required resources which facilitated my work, especially the unrestricted access to the robotics laboratories, with all of their resources, and materials, among others.

A special sense of gratitude to my loving family, particularly my mother, for her faith in me, encouragement and moral support, which provided the needed motivation during times of difficulty.

My most special thanks to the Culture and Communication office of the university of Coimbra, especially to Dr. Teresa Baptista, also to the office of president Jorge Sampaio for their continuous support particularly the president's advisor Dr. Helena Barroco.

Many thanks to my friends and colleagues in the collaborative robotics labs in both Universities.

This thesis was possible thanks to the support of the Foundation for Science and Technology (FCT), through grant SFRH/BD/131091/2017.

# Acronyms List

| | |
|---|---|
| **EEF** | End-Effector |
| **DoF** | Degrees of Freedom |
| **HG** | Hand-Guiding |
| **DH** | Denavit-Hartenberg |
| **APF** | Artificial Potential Fields |
| **DLS** | Damped Least Squares |
| **IMU** | Inertial Measurement Unit |
| **LPF** | Low-Pass Filter |
| **MVA** | Moving Average |
| **3D** | 3 Dimensional |
| **PRM** | Probabilistic RoadMaps |
| **CAD** | Computer Aided Design |
| **HRI** | Human Robot Interaction |
| **PbD** | Programming by Demonstration |
| **CA** | Collision Avoidance |
| **TCP** | Tool Centre Point |
| **KST** | KUKA Sunrise Toolbox |
| **KCT** | KUKA Control Toolbox |
| **ROS** | Robot Operating System |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |
| **PTP** | Point To Point |
| **COM** | Centre of Mass |
| **FT** | Force & Torque |
| **JSIM** | Joint Space Inertia Matrix |
| **CRBA** | Composite Rigid Body Algorithm |
| **GDAHJ** | Geometric Dynamics Algorithm for High number of Joints |
| **HDoF** | High Degrees of Freedom |

# Nomenclature

The mathematical notation used throughout this study is the same notation used in [1], and is described below:

1. Bold capital letters are used to denote matrices, $\mathbf{J}$ for the Jacobian matrix.
2. Bold and Italic small letters are used to denote vectors, $\boldsymbol{q}$ for the joints positions vector.
3. Italic small letters are used to denote scalars, $q_j$ is the angular position of joint $j$.
4. Dot operators are used to denote time derivatives $\dot{\boldsymbol{q}} = d\boldsymbol{q}/dt$.
5. Transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A^T}$.
6. The inverse of a matrix $\mathbf{A}$ is denoted by $\mathbf{A^{-1}}$.
7. The pseudo inverse of a matrix $\mathbf{A}$ is denoted by $\mathbf{A^\dagger}$.
8. Vector cross product is denoted by $\times$.
9. The skew symmetric operator of a vector $\boldsymbol{p}$ is notated by $\hat{\boldsymbol{p}}$. Where $\hat{\boldsymbol{p}}$ is used as the matrix representation of the cross product $\boldsymbol{p}\times$.
10. The gradient of a scalar function $u$ is denoted using the Nabla operator by $\nabla u$. In this case, as reported in [2] P196.
11. The Hessian of a scalar function $u$ is denoted using the Nabla operator by $\nabla^2 u$. In this case, as reported in [2] P196.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

Nowadays, industrial robots are cornerstones in modern factories. Their involvement in workspace contributes to higher production capabilities, flexibility and accuracy. They are used extensively in manufacturing for pick and place applications, painting, welding, metal processing, and even in food industry. Owing to their accuracy, they are also becoming valuable tools for performing precise operations in aerospace industry. In the framework of Industry 4.0, the presence of robots in human centred environments is of vital importance, as it brings to the factory floor the best of both, the cognitive capacity of humans and the special qualities of robots, including their precision, strength and repetitiveness. Such approach will render production lines more productive and flexible. But, industrial robots are currently separated from humans behind fences for safety reasons. In addition, they are still hard to program, even for experienced users. Those drawbacks form a barrier to effective achievement of human-robot cooperation and interaction on the factory floor.

In such a case, it is required to have safer robots with better collaborative capabilities, that are easier to program without requiring high technical knowledge. Which have the ability to avert danger and ensure safety of the human coworker during mutual interaction.

## 1.1 Context and Motivation

In the light of the previous introduction, the proposed research study addresses the issues of safety and collaboration for industrial manipulators. On safety, the study delves into the subject of collision avoidance for robotic manipulators in a human-robot shared workspace. On collaboration, the study explores the subject of precision hand-guiding for robotic manipulators. In both cases, the aim of the study is to apply the acquired results in industry. The following two subsections elaborate on the importance of both collision avoidance and hand-guiding functionalities, and lay down the arguments on the necessity of researching those subjects.

Figure 1.1: KUKA iiwa robot working side-by-side with a human coworker in Aerospace industry, courtesy of ColRobot project [3].

### 1.1.1 Collision Avoidance

Providing robots with advanced motor skills and human like reflexes will allow them to circumvent obstacles and avoid collisions with the dynamic environment around them. This is extremely important in order to give robots more autonomy and minimum need for human intervention, especially when robots are operating in changing workspace and dynamic environments.

When it comes to the new industrial revolution, industry 4.0, collision avoidance is vitally important, where it is expected to have humans and robots working with each others, and collaborating together, a quest that can not be achieved without providing the robot with collision avoidance capacities. Usually, the structure of industrial robots is made of metals (retain immense inertia). Due to productivity reasons, they are also required to move with high speeds, conditions that make it unsafe for humans to work side-by-side with a robot moving blindly on a pre-programmed path. Consequently, it has been the norm to have boundaries and safety fences that stand between humans and industrial robots. When it comes to the factories of the future, the idea is to eliminate these boundaries and have humans and robots sharing the same workspace, Figure 1.1. In such scenario robot's control system shall ensure the safety of the human coworker under all circumstances.

Another example on the importance of collision avoidance algorithms is robots required to work in dangerous environments. For example, when performing (robot operated) repairs inside nuclear reactors, or when working in potentially explosive environments. Figure 1.2 demonstrates such concept, where a robotic arm is actuating a valve in an oil infrastructure while being surrounded by gas pipes. This type of research is motivated by the fact that robots are required if humans want to tap into oil and gas reserves in remote regions where extremely harsh environments prevail,

Figure 1.2: A robot operating in potentially explosive environment, among pipes carrying gas [4].

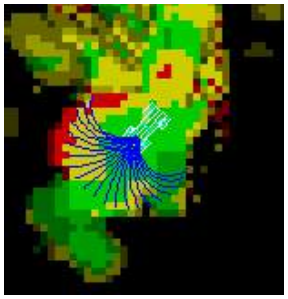

Figure 1.3: Morphin map of Curiosity Mars rover, courtesy of Mars autonomy project.

Figure 1.4: The Curiosity rover on Mars (NASA).

for example the reservoirs in the Barents Sea where temperatures can reach as low as $-50°C$. In such places working conditions for humans are unfeasible. A study focused on such situation was proposed in [7], where the author's work has focused on collision avoidance for industrial robots in gas and oil industry.

Apart from safety, collision avoidance capabilities endow robots with more autonomy. This is especially important when they are operating in remote places and in unknown environments. An example on this is the Curiosity Mars rover, Figure 1.4. The rover works far away from earth, this makes it impossible to control the asset in real time. As a matter of fact, the distance between the two planets is immense, such that the time required for sending a signal to Mars and then receiving it back on earth could be in between 4 to 24 minutes (depending on the relative position of the two planets while they are rotating around the sun). So to keep the robotic rover away from obstacles, a collision avoidance algorithm called Morphin was developed, and is used to guide the rover through the terrains of Mars. This algorithm maintains a map of the environment, Figure 1.3, and based on this map Morphin recommends safe steering commands to the rover.

In robotics literature, the problem of collision avoidance is handled at one of two levels: global level, addressed through planning, or local level treated at the low level control. The global solutions are high-level solutions that guarantee to find a collision-free path from the initial configuration to the final configuration, if such a path exists.

These algorithms treat the problem in configuration space. In such case the manipulator and the environment need to be remapped to configuration space and a collision-free path is searched in the unoccupied portion of the configuration space. However, these algorithms are very costly in terms of computation, thus their use is not feasible for real-time dynamic applications. On the other hand, the local reactive control is suitable for real-time implementation, since that its mathematical formulation is computationally efficient, so it can be embedded directly into the low level control.

### 1.1.2 Hand-guiding

Hand-guiding is a representative functionality of collaborative robots. It allows unskilled users to interact and program a robot rapidly in an intuitive manner. Many industrial applications require precise positioning of the end-effector (EEF) during the teaching process. Normally, the teach pendant is used for performing this task. However, utilizing the teach pendant is not always convenient due to the following drawbacks:

1. When using the teach pendant to position the end-effector in Cartesian space, the user has to keep a track of the orientation of the base frame of the robot, this could become confusing even for experienced users. Not to mention that in some cases, the manipulator is mounted on a mobile platform, which makes the task of keeping a track of robot's base frame even harder.

2. Unlike the hand-guiding, when using the teach pendant the user does not have a feel of the force applied between the end-effector and its surrounding in case of contact. Accidents could happen and the user might over press the end-effector against the surrounding (which might include sensitive instruments) without noticing.

3. The teach pendant convention in describing the orientation is the Euler rotation angles. While it is well suited for computers, this way for describing orientation is not intuitive for humans. Even the experienced roboticist agrees that, in the general case, it is hard for a human to imagine the orientation of an object based on three numbers (Euler angles).

Due to the previous drawbacks, this thesis proposes a more intuitive method for performing precise positioning operation of the end-effector through hand-guiding.

## 1.2 Key Contributions

The thesis contributes into the field of collaborative robotics in various aspects, including the subjects of collision avoidance for collaborative industrial manipulators, hand-guiding, and mathematical formulation of robot dynamics, Figure 1.5. In the following, the main contributions are listed according to each subject.

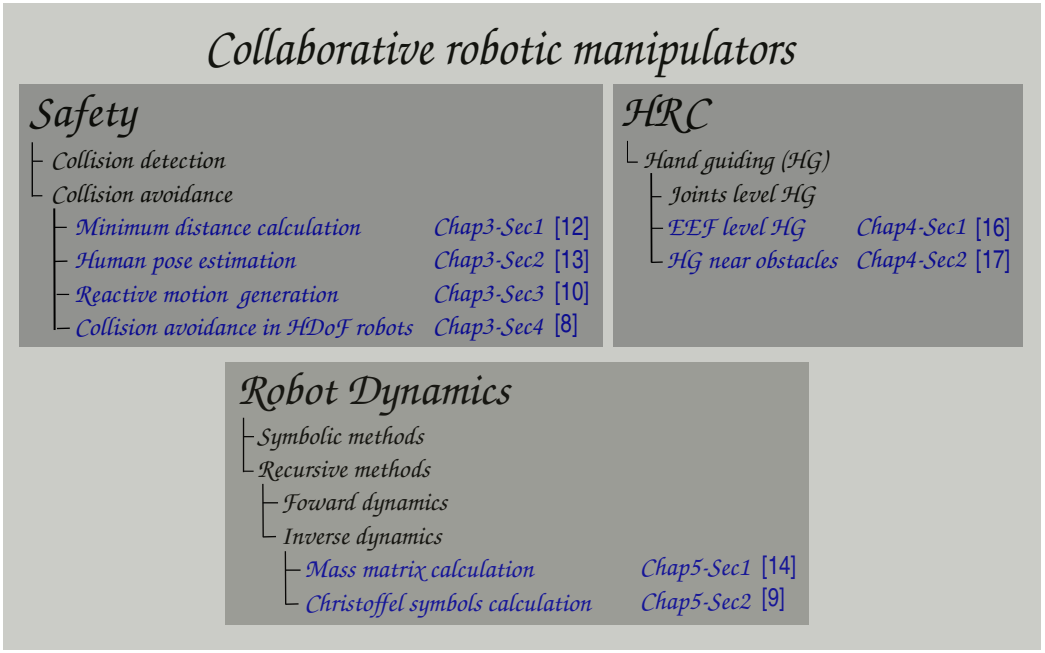


Figure 1.5: Graphical representation of thesis contributions into the field, the contributions are highlighted in blue.

**Key contributions in collision avoidance:**

1. Development of an algorithm for capturing the configuration of the human coworker using laser scanner and IMU sensors. The developed algorithm is implemented in the context of human-robot collision avoidance.

2. Development of an efficient algorithm, based on optimization and QR factorisation, for fast calculation of minimum distances among a set of capsules (hemisphere capped cylinders).

3. Development of a real-time, potential fields based, collision avoidance algorithm. This novel algorithm is suitable for application in industrial robotic cells, allowing the robot to adjust the off-line generated paths of the industrial task. The proposed method also introduces a novel solution that allows the robot to switch between collision avoidance reactive motion and the task in hand while maintaining continuous and smooth movement.

4. Development of a new algorithm for generating reactive collision avoidance motion (for robotic manipulators), where the Newton method (from optimization) is applied on the potential field function for calculating the minimization direction. For achieving efficient execution, the relationship between the Hessian (of the potential function) described in joint space and the Hessian in Cartesian space is also deduced.

**Key contributions in hand-guiding:**

1. Introducing the precision hand guiding method. An intuitive substitute for the teach pendant of the robot for performing precise positioning of the end-effector.

2. Introducing a novel solution for performing precision hand guiding of sensitive redundant manipulators, at the end-effector level, while navigating obstacles during contact.

**Key contributions in robot dynamics:**

1. Development of an algorithm with a minimal $O(n^2)$ cost[1] for efficiently calculating the joint space inertia matrix for serially linked robots with high degrees of freedom.

2. Development of a light weight recursive (non-symbolic) algorithm for calculating Christoffel symbols of robotic manipulators.

## 1.3 Thesis Outline

The thesis is divided into various chapters and sections. Chapter two gives the background, it also lists the basic principles required for the mathematical modelling of robots, all of which will be used throughout this document.

Chapter three is dedicated to the subject of collision avoidance between humans and industrial manipulators. We start this chapter by listing out the various methods in literature for representing the human and the robot. For this thesis, we choose the geometric primitive representation using capsules. Then, we derive an efficient algorithm for calculating the minimum distance between these capsules. Afterwards, a method based on sensor fusion is presented for capturing the configuration of the human around the robot. This method is then implemented in a collision avoidance algorithm derived from the potential fields method and tested on a real industrial robotic cell, where a human coworker and an industrial robot (KUKA iiwa) are sharing the same work space and tasks. In such a case, the robot is able to sense the presence of the human around it, and reacts on-the-fly to avoid collisions while keeping the task target as possible. Finally, we propose coupling Newton method with potential fields collision avoidance algorithm in a second order optimisation framework. A symbolic mathematical formula is deduced for calculating the Hessian and the gradient of the potential field efficiently in the joint space of the robot. The advantages of using such solution for performing collision avoidance is demonstrated through various tests.

Chapter four explores the subject of hand-guiding while preserving the precision at the end-effector level. First, the main method for performing precision hand-guiding is introduced, the method is deduced mathematically and tested on KUKA iiwa robot. Then the precision hand-guiding (at the EEF level) and the hand-guiding at joints level are compared in terms of precision and level of vibrations. Finally, by applying sensor fusion on force/torque measurements from an external force/torque sensor attached

[1] Where $n$ is the number of degrees of freedom of the robot.

at the end-effector and the torque measurements from integrated joint torque sensors, we propose a method that allows the user to hand guide the robot precisely, at the end-effector level, while navigating obstacles in the null space during a contact.

Chapter five deals with the formulation of robot dynamics equations. Two efficient algorithms are proposed. The first algorithm is for calculating the joint space inertia matrix of serially linked rigid bodies. It is an efficient algorithm with a minimal second order cost. The second algorithm describes a recursive method for calculating Christoffel symbols of robotic manipulators efficiently. The proposed algorithms were tested against state of the art methods. Results were reported in terms of number of operations, execution time and numerical error.

## 1.4 Publications and Technical Contributions

The findings and the intermediate results achieved during the research period were the subject for various publications in (international) peer reviewed journals and conferences, including some of the main robotic congresses. Table 1.1 lists the main publications conducted in the scope of this thesis. Table 1.2 lists coauthored publications that have been conducted in cooperation with colleagues during the research period. Moreover, implementing the proposed approach led to the development of a set of open source software packages/toolboxes (listed in Table 1.3), some of which, the Kuka Sunrise Toolbox (KST), gained popularity and is being used in various laboratories around the world.

Table 1.1: List of main publications

| Reference | Description |
|---|---|
| Journal [8] | **M. Safeea**, R. Béarée and P. Neto: Collision avoidance of redundant robotic manipulators using Newton's method. *Journal of Intelligent & Robotic Systems*. |
| Journal [9] | **M. Safeea**, P. Neto and R. Béarée: Robot dynamics: A recursive algorithm for efficient calculation of Christoffel symbols. *Mechanism and Machine Theory*. |
| Journal [10] | **M. Safeea**, P. Neto and R. Béarée: On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case. *Robotics and Autonomous Systems*. |
| Journal [11] | **M. Safeea** and P. Neto: KUKA Sunrise Toolbox: Interfacing collaborative robots with MATLAB. *IEEE Robotics & Automation Magazine*. |
| Journal [12] | **M. Safeea**, P. Neto and R. Béarée: Efficient Calculation of Minimum Distance Between Capsules and its use in Robotics. *IEEE Access*. |
| Journal [13] | **M. Safeea** and P. Neto: Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*. |
| Conference [14] | **M. Safeea**, R. Béarée and P. Neto: Reducing the computational complexity of mass-matrix calculation for high DOF robots. *IROS 2018*. |
| Conference [15] | **M. Safeea** and P. Neto: Human-robot Collision Avoidance for Industrial Robots: A V-REP based Solution. In *Proc. of the 25th International Conference on Transdisciplinary Engineering*. |
| Conference [16] | **M. Safeea**, R. Béarée and P. Neto: End-Effector Precise Hand-Guiding for Collaborative Robots. In *Proc. of the third Iberian Robotics Conference*. |
| Conference [17] | **M. Safeea**, R. Béarée and P. Neto: Precise hand-guiding of redundant manipulators with null space control for in-contact obstacle navigation. In *Proc. of the* 45th Annual Conference of the IEEE Industrial Electronics. |
| Book chapter (accepted) | **M. Safeea**, P. Neto and R. Béarée: Task execution combined with in-contact obstacle navigation by exploiting torque feedback of sensitive robots. *Procedia Manufacturing*. |
| Book chapter (accepted) | **M. Safeea**, P. Neto and R. Béarée: Model-based hardware in the loop control of collaborative robots. *Procedia Manufacturing*. |
| Book chapter [18] | **M. Safeea**, N. Mendes, and P. Neto: Minimum distance calculation for safe human robot interaction. *Procedia Manufacturing*. |
| Book chapter [19] | **M. Safeea**, P. Neto and R. Béarée: A Geometric Dynamics Algorithm for Serially Linked Robots. *Part of the Lecture Notes in Computer Science book series (LNCS, volume 11649)*. |
| Book chapter [20] | **M. Safeea**, P. Neto and R. Béarée: The Third Hand, Cobots Assisted Precise Assembly. *Part of the Lecture Notes in Computer Science book series (LNCS, volume 11650)*. |
| Book chapter [21] | **M. Safeea**, P. Neto and R. Béarée: A Quest Towards Safe Human Robot Collaboration. *Part of the Lecture Notes in Computer Science book series (LNCS, volume 11650)*. |

Table 1.2: List of coauthored publications

| Reference | Description |
|---|---|
| Journal [22] | P. Neto, M. Simão, N. Mendes and **M. Safeea**: Gesture-based human-robot interaction for human assistance in manufacturing. *International Journal of Advanced Manufacturing Technology*. |
| Conference [23] | N. Mendes, **M. Safeea** and P. Neto: Flexible programming and orchestration of collaborative robotic manufacturing systems. In *Proc. of the 16th International Conference on Industrial Informatics*. |
| Book chapter [24] | J. Lopes, M. Simão, N. Mendes, **M. Safeea**, J. Afonso, and P. Neto. Hand/arm gesture segmentation by motion using IMU and EMG sensors. *Procedia Manufacturing*. |
| Book chapter [25] | N. Mendes, J. Ferrer, J. Vitorino, **M. Safeea** and P. Neto: Human behavior and hand gesture classification for smart human-robot interaction. *Procedia Manufacturing*. |

Table 1.3: List of developed software packages

| Package | Description |
|---|---|
| KST [26] | A Toolbox used to control KUKA iiwa robots, the 7R800 and the 14R820, from an external computer using MATLAB. |
| SimulinkIIWA [27] | A graphical programming interface that allows controlling KUKA iiwa robots, the 7R800 and the 14R820, using Simulink block diagrams. |
| iiwaPy [28] | A library that allows controlling KUKA iiwa robots, the 7R800 and the 14R820, using Python 2.7. |
| GDA [29] | A collection of MATLAB functions used for calculating robot dynamics (including Joint Space Inertia Matrix (JSIM), Coriolis matrix, centrifugal matrix, time derivative of JSIM), the algorithms implemented for preforming the calculations are derived from the Geometric Dynamic Algorithm (GDA). |
| GDAHJ [30] | C++ implementation of GDAHJ algorithm for calculating mass matrix of robots with a minimal second order cost |
| Christoffel recursively[31] | Recursive algorithm for calculating Christoffel symbols efficiently implemented in MATLAB |

# Chapter 2

# Background

This chapter reviews the basics which are important for the implementation of the methodologies proposed in this thesis. Section 2.1 gives a review on the kinematics of robotic manipulators, the transformation matrices and the formula for the Jacobian. Section 2.2 gives a review on the pseudo inverse of the Jacobian and the Damped Least Squares (DLS) method. Section 2.3 gives a review on the null space projection matrix for redundant manipulators. Section 2.4 gives a brief introduction into the artificial potential fields.

## 2.1 Kinematics of Robotic Manipulators

Most of industrial manipulators consist of rigid links coupled serially together by revolute joints. In such a case, robot's configuration is described by its joints angular position vector $\boldsymbol{q} \in \Re^n$, where $n$ is the number of degrees of freedom of the robot. On the other hand robots are required to perform tasks in the Cartesian space, specified by the position and orientation of the robot's end-effector (vector $\boldsymbol{x} \in \Re^m$). The forward kinematics function [32] introduces a mapping from the robot's configuration $\boldsymbol{q}$ to the end-effector's position/orientation $\boldsymbol{x}$:

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \tag{2.1}$$

Where $\boldsymbol{f}$ is the manipulator's forward kinematics, a highly non-linear vector function, that gives the position and orientation of the end-effector as a function of joints angles. This function can be calculated using the homogeneous transformation matrices, following Denavit-Hartenberg (DH) convention (first introduced in [33]). From [32] a transformation matrix (using the modified DH parameters) of link $i$ is given by:

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Where $i$ is the joint index, $\theta_i$ is the rotation angle of joint $i$ around the axis $z_i$, and $(d_i, \alpha_{i-1}, a_{i-1})$ are the modified DH parameters of the link $i$. Differentiating (2.1) gives the equation of differential kinematics [1],[34]:

$$\dot{\boldsymbol{x}} = \mathbf{J}\dot{\boldsymbol{q}} \tag{2.3}$$

Where $\dot{\boldsymbol{q}}$ is the joints velocities vector, $\dot{\boldsymbol{x}}$ is the end-effector's velocity and $\mathbf{J}$ is the manipulator's analytic Jacobian matrix, each element of which:

$$J_{ij} = \frac{\delta f_i}{\delta q_j} \tag{2.4}$$

However, the analytic Jacobian can also be calculated from the geometric Jacobian by pre-multiplication with a matrix whose value depends on the choice of orientation representation [1].

## 2.2 Pseudo Inverse

For non-redundant manipulators the inverse mapping, from a particular velocity of EEF to joints velocity is given by:

$$\dot{\boldsymbol{q}} = \mathbf{J}^{-1}\dot{\boldsymbol{x}} \tag{2.5}$$

In the case of redundant manipulators, $\mathbf{J}$ is not square (has more columns than rows) and the matrix $\mathbf{J}^{-1}$ does not exist. In such a case, the least norm solution of the inverse mapping is given by the pseudo inverse:

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\dagger}\dot{\boldsymbol{x}} \tag{2.6}$$

Where $\mathbf{J}^{\dagger}$ is the Moore-Penrose or pseudo inverse of the Jacobian:

$$\mathbf{J}^{\dagger} = \mathbf{J}^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{\mathrm{T}})^{-1} \tag{2.7}$$

In some configurations the previous equation suffers from singularity problem. This happens when the matrix product $\mathbf{J}\mathbf{J}^{\mathrm{T}}$ losses rank. In such a case the Damped Least Squares [35] can be used:

$$\dot{\boldsymbol{q}} \approx \mathbf{J}^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{\mathrm{T}} + \lambda^2\mathbf{I})^{-1}\dot{\boldsymbol{x}} \tag{2.8}$$

Where $\mathbf{I}$ is the identity matrix $(m \times m)$ and $\lambda$ is a damping factor. For choosing an optimised value of the damping factor the method in [36] can be used.

## 2.3 Null Space Projection

Redundant manipulators possess more degrees of freedom than it is needed to execute a given task. In such a case, the inverse mapping (joints angular velocities that give a particular velocity of EEF) involves infinite number of solutions, those solutions are given by the equation:

$$\dot{\boldsymbol{q}} = \mathbf{J}^{\dagger}\dot{\boldsymbol{x}} + \mathbf{N}\dot{\boldsymbol{q}}_c \tag{2.9}$$

Where $\mathbf{N}$ is the null space projection matrix and $\dot{\boldsymbol{q}}_c$ is an angular velocity command. $\dot{\boldsymbol{q}}_c$ is commonly used to optimize some criteria. From [37], the basic formula of $\mathbf{N}$ is given by:

$$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{\mathrm{T}})^{-1}\mathbf{J} \tag{2.10}$$

Where $\mathbf{I}$ is the identity matrix $(n \times n)$. From (2.9) it can be noticed that redundant manipulators allow internal motion where the pose (position/orientation) of EEF is kept fixed, in such a case $\dot{\boldsymbol{x}} = 0$, and (2.9) becomes:

$$\dot{\boldsymbol{q}}_{null} = \mathbf{N}\dot{\boldsymbol{q}}_c \tag{2.11}$$

Where $\dot{\boldsymbol{q}}_{null}$ is the vector of the angular velocities in the null space of the robot.

## 2.4 Artificial Potential Field

The Artificial Potential Field (APF) method introduced by the pioneering work of Khatib [38] is one of the most popular methods for performing real-time reactive collision avoidance (according to the number of citations). This method was applied successfully for both manipulators and mobile robots. In this method the robot is subjected to two types of potential fields. One represents the attraction forces that pull the robot toward the goal position, while the other represents the repulsion forces that push the robot away from obstacles in the environment. Those forces are defined by the gradient of the potential field. As a result the robot moves toward the goal while avoiding collisions with obstacles. Owing to its additive property, the total potential field is the sum of the attraction and the repulsion potentials:

$$U_{total} = U_{att} + U_{rep} \tag{2.12}$$

Figure 2.1 demonstrates the artificial potential fields concept and illustrates their additive property. The total field (Figure 2.1 right) is the sum of the attraction field (Figure 2.1 left), with forces pulling towards the goal, and the repulsion field (Figure 2.1 middle), with forces pointing away from obstacles. Owing to its computational efficiency the method can be implemented in real-time, and can be deployed easily into the low-level control system of the robot. The use of this method had been extended successfully to offline global path planning problems. However, in the context of this study we limit the discussion to real-time collision avoidance implementation.

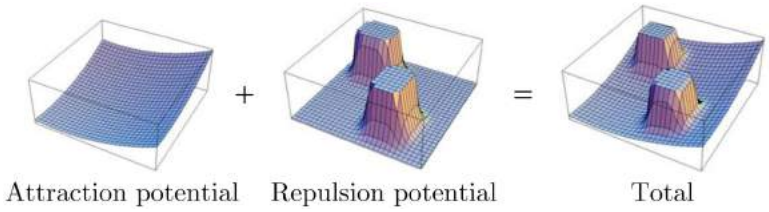


Figure 2.1: Artificial potential field concept.

# Chapter 3

# Collision Avoidance

Modern factories are becoming more automated. Greater number of robots are being installed on production lines for performing various industrial operations including welding, manipulation and assembly. This is particularly advantageous in repetitive tasks that require precision. However, the presence of humans on the factory floor is still essential for performing complex and cognitive tasks. For safety concerns, robots are separated from humans, where they work behind fences. This physical separation limits the productivity and the flexibility of the production lines. Currently, lots of attention and efforts are being directed for removing the barriers between humans and robots on the factory floor. This will bring better productivity by combining the precision and repetitiveness of robots with the cognitive ability of humans. In the framework of Industry 4.0, collision avoidance plays an important role for achieving safety criteria while having humans and machines working side by side. Consequently, there is a requirement for adapting the research results on robot collision avoidance into the factory floor. This study introduces the subject of manipulator's on-line collision avoidance into a real industrial application implementing typical sensors and a commonly used collaborative industrial manipulator (KUKA iiwa). In the proposed method, the human coworker and the robot are represented by capsules, the configuration of those capsules are acquired using external sensors, the minimum distance between the capsules is calculated, when human/obstacles are nearby the concept of hypothetical repulsion and attraction vectors is used. By coupling this concept with a mathematical representation of robot's kinematics we achieved a task level control with collision avoidance capability, where the off-line generated nominal path of the industrial task is modified on–the-fly so the robot is able to avoid collision with the coworker safely while being able to fulfil the industrial operation. Tests were carried out successfully on an industrial robotic cell performing typical assembly operations in automotive industry. Results show that the robot moves smoothly and avoids collisions successfully by adjusting the off-line generated nominal paths.

## Chapter's breakdown

This chapter is organised into various sections:

- Section 3.1: lists the various methods applied in the literature for representing

the geometry of humans and the robots. Due to its computational efficiency we opted to represent geometries by capsules, where we present an efficient algorithm for calculating the minimum distance between capsules using QR factorization.

- Section 3.2: describes the method used for capturing the configuration of the capsules representing the human using data acquired from laser scanner and a group of Inertial Measurement Unit sensors (IMUs).
- Section 3.3: describes an implementation of a custom-made collision avoidance algorithm based on the potential field method. The algorithm is applied on a robotic cell containing a collaborative industrial manipulator (KUKA iiwa), where the robot has to perform an industrial task while avoiding collisions with a human coworker around it, sharing the same work space and tasks.
- Section 3.4: describes the application of Newton's method on the potential fields for collision avoidance, and discusses the advantages of such implementation.
- Section 3.5: describes a software package developed by the author which allows controlling KUKA iiwa robots from external computer using MATLAB.

Given that the chapter discusses different subjects, each in its own section, then for the sake of clarity, the author has opted to list the related work (state of the art) of each subject in the beginning of its respective section. In such a case, each section (of the previously listed) starts by a small introduction, followed by a review of the related work, afterwards the developed methodology is presented. Each of the proposed methods is validated by testing or simulation (sometimes both), the acquired results are reported in the experiments subsection. Finally each section is concluded with a summary.

## 3.1 Geometrical Representation

To implement a collision avoidance algorithm it is of vital importance to have a geometrical representation of the robot and the human/obstacles. Several methods had been proposed in literature for this purpose. In [39] convex polyhedrons are utilized to represent two PUMA 560 manipulators. In [40] the authors opted for ellipses to represent the links of the robot, while obstacles were represented by spheres. Another technique that is computationally efficient represents the robot and obstacles by primitive shapes, as segments of lines with spheres swept onto them, [41] and [42]. A similar convention was implemented in [43], where obstacles and robot are represented by spheres and cylinders. Also, in [44] the authors represented a humanoid robot by cylindrical shapes. While in [45] the authors opted to represent the robot and the coworker by a collection of spheres for efficient calculation of the minimum distance.

Another technique, which is more precise for representing the robot and the environment around it, utilizes mesh representation [4]. This type of representation has the disadvantage of being extremely costly in terms of computations while performing minimum distance calculations between the robot and the obstacle. To speed up the computation, researchers have utilized the power of parallel processing, using GPU, to carry out the calculations [4]. Nevertheless, programming a GPU is not an easy task, it requires considerable amount of time and special skills, so it had been avoided for the sake of this study. Thus, the primitive geometry method was our choice, the coworker and the robot are represented by hemisphere capped cylinders (capsules).

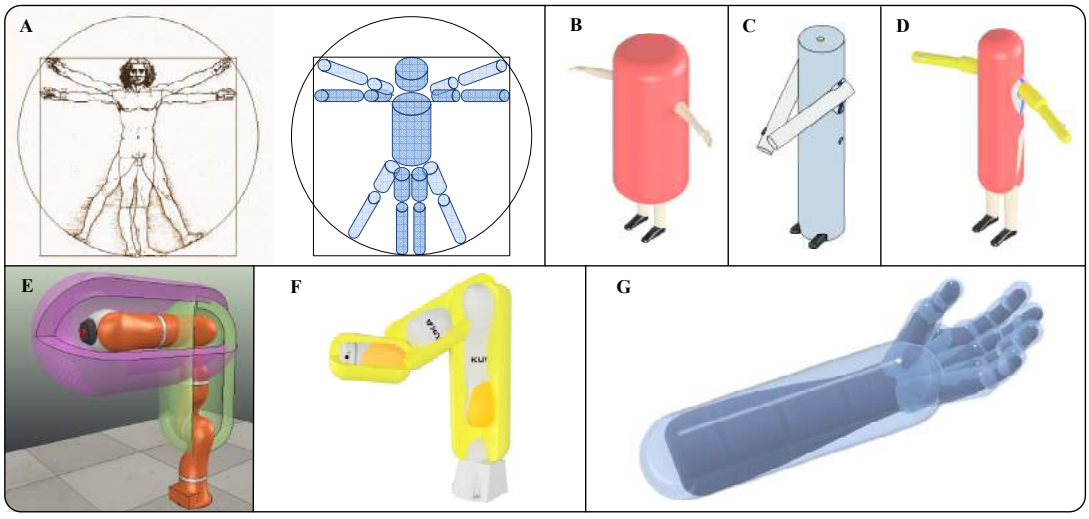


Figure 3.1: (A) L'Uomo Vitruviano from Leonardo da Vinci and its representation by 10 capsules, (B) human represented by 1 capsule, (C) human represented by 3 capsules, (D) human represented by 5 capsules, (E) robot represented by 2 capsules, (F) robot represented by 3 capsules and (G) human hand and forearm are represented by 21 capsules.

### 3.1.1 Minimum Distance Calculation

The subject of minimum distance calculation is important in many areas, for example in CAD design, computer graphics/games and in simulation. In such situations, minimum distance calculations are used to detect any overlap or collision between elements. This subject is also important in robotics for the problem of path planning and safety in human-robot interaction, where the minimum distance is used as a measure of collision imminence. In this scenario, calculating the minimum distance online is required for time critical applications such as human-robot collision avoidance and the path planning of robots navigating obstacles towards a goal. A novel approach to approximate the minimum distance between robot links and obstacles is proposed in [46]. Obstacles are represented by a bounding box, modelled as cylinders and boxes. Each part of the robot arm is subdivided into an optimal number of spheres that encompass its volume. The minimum distance between the robot and the objects is approximated by the minimum distance between the obstacle bounding box and the spheres. In [47] a method is proposed to determine the minimum distance between multiple known objects (geometry, position, orientation and configuration for example a robot) and multiple unknown objects within a camera image. The distance is estimated by searching for the largest expansion radius where the projected model does not intersect the object areas classified as unknown. An algorithm for computing the minimum translational distance based on the Gilbert-Johnson-Keerthi algorithm between two spherically extended polytopes is introduced in [48].

In our study, the implemented collision avoidance algorithm requires the calculation of the minimum distance between robot and surrounding environment (including humans), which are represented by capsules. By using higher number of geometric primitives the accuracy of representation increases. The human body can be represented roughly by a single capsule, Figure 3.1 (B). In this scenario the arms can extend out of the capsule volume for some configurations. Figure 3.1 (C) shows a human represented by 3 capsules. In Figure 3.1 (D) the human is represented by 5 capsules, 2 capsules in each arm and 1 single capsule for the torso and head. A relatively precise representation of the human hand and forearm by 21 capsules is shown in Figure 3.1 (G). A robot can roughly be represented by 2 capsules, Figure 3.1 (E), or by 3 capsules representing the main robot links (KUKA iiwa with 7 DoF), Figure 3.1 (F). The position and orientation of the capsules covering the robot are obtained from forward kinematics calculation at the measured robot joint angles. In literature, various methods were proposed for calculating the minimum distance between capsules/line-segments. In [49], it is proposed an algebraic method for minimum distance computation between two capsules. Other methods were proposed in [50, 51]. A solution for computing the minimum distance between cylinders with flat ends was proposed in [52]. A well known methodology for computing the segment to segment (capsules) distance which is considered the most efficient method in literature (concerning computation time) is detailed in [5], pages 417-418.

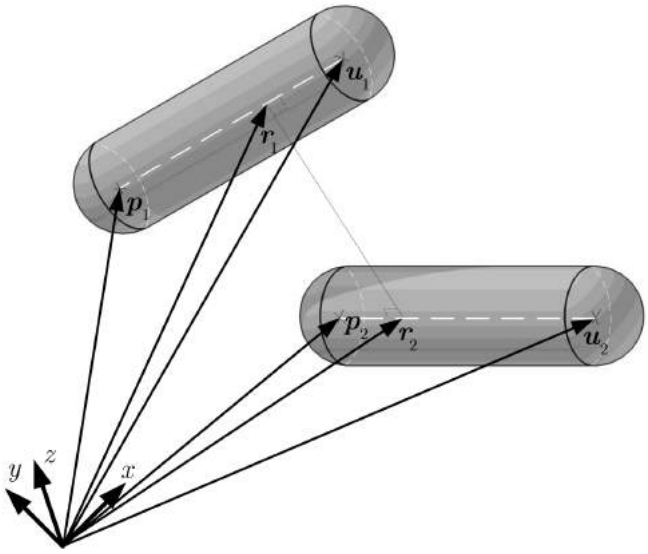


Figure 3.2: Minimum distance between two capsules.

### 3.1.2 Mathematical Formulation and QR Factorization

In this section the problem of calculating the minimum distance between two capsules is expressed mathematically as an optimization problem. Then the optimization function is reformulated using QR factorization and a geometrical solution is proposed.

#### 3.1.2.1 Formulation

Owing to their geometry, calculating the minimum distance between two capsules can be reduced to the calculation of the minimum distance between two line-segments at the capsules axes. In Figure 3.2 it is shown two line-segments representing the axes of two capsules and their associated common normal. Each capsule can be defined by two vectors (at the beginning and end of the capsule's axis-segment) and a radius. Let's designate the position vectors defining the end points of the axis-segment of a capsule by $p_1$ and $u_1$ (capsule 1), and $p_2$ and $u_2$ (capsule 2). Then, we can define two vectors $s_1 = u_1 - p_1$ and $s_2 = u_2 - p_2$. Two points of interest, one in each axis segment of a capsule, are considered. Those points are represented relative to the base frame by two vectors, $r_1$ and $r_2$:

$$\boldsymbol{r}_1 = \boldsymbol{p}_1 + \boldsymbol{s}_1\lambda_1 \tag{3.1}$$

$$\boldsymbol{r}_2 = \boldsymbol{p}_2 + \boldsymbol{s}_2\lambda_2 \tag{3.2}$$

Where $\lambda_1$ and $\lambda_2$ are scalar parameters. Each parameter has a value in the range from zero to one when the point it represents is confined in between the two ends of the axis segment of the capsule. The problem of calculating the minimum distance between the two capsules renders to a minimization problem:

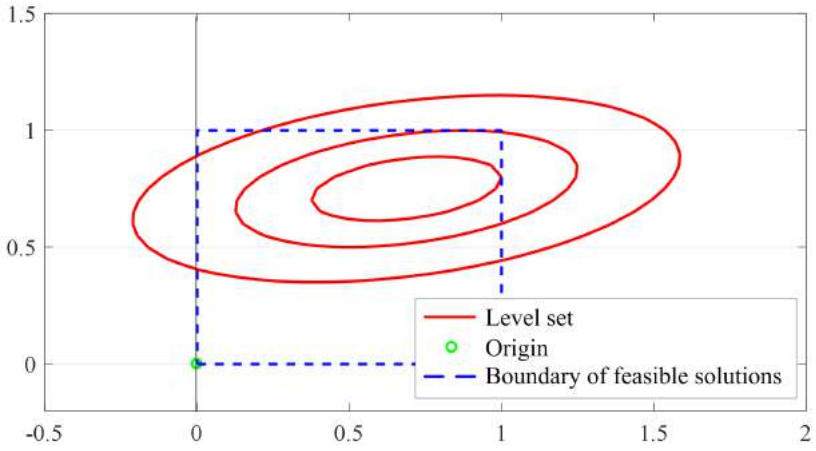


Figure 3.3: Region of feasible solutions and level sets of optimization problem (3.5).

$$min(\phi) = min(\mid \boldsymbol{p}_2 + \boldsymbol{s}_2\lambda_2 - (\boldsymbol{p}_1 + \boldsymbol{s}_1\lambda_1) \mid -\rho_1 - \rho_2) \tag{3.3}$$

Where $\rho_1$ and $\rho_2$ are the radii of the capsules. Giving that $\rho_1$ and $\rho_2$ are constants, then the minimization problem can be reformulated:

$$min(\mid \Delta\boldsymbol{r} \mid) = min(\mid \boldsymbol{p}_2 + \boldsymbol{s}_2\lambda_2 - (\boldsymbol{p}_1 + \boldsymbol{s}_1\lambda_1) \mid) \tag{3.4}$$

Where $\Delta\boldsymbol{r} = \boldsymbol{r}_2 - \boldsymbol{r}_1$. We can rewrite the optimization function in the following equivalent quadratic form:

$$min(f) = min((\mathbf{A}\boldsymbol{x} + \boldsymbol{y})^{\mathrm{T}}(\mathbf{A}\boldsymbol{x} + \boldsymbol{y})) \tag{3.5}$$

Where matrix $\mathbf{A} = \begin{bmatrix} \boldsymbol{s_2} & -\boldsymbol{s_1} \end{bmatrix}$ and vector $\boldsymbol{y} = \boldsymbol{p_2} - \boldsymbol{p_1}$. The problem can be viewed as minimizing (3.5), subject to the constrains $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$ ($x_1$ and $x_2$ are the components of the vector $\boldsymbol{x}$ - also they are equal to parameters $\lambda_1$ and $\lambda_2$). Figure 3.3 shows the level sets and the region of feasible solutions of the optimization problem (3.5).

#### 3.1.2.2 QR Factorization

The function $f$ can be reformulated by performing QR factorization on matrix $\mathbf{A}$ and fixing. Then, the optimization problem in (3.5) is equivalent to:

$$min(f_1) = (\mathbf{R}\boldsymbol{x} + \mathbf{Q^T}\boldsymbol{y})^{\mathrm{T}}(\mathbf{R}\boldsymbol{x} + \mathbf{Q^T}\boldsymbol{y}) \tag{3.6}$$

Where Q is a 3x2 matrix whose column vectors are of unit length and mutually orthogonal, and matrix R is a 2x2 upper triangular. By performing a variable change the optimization problem becomes:

$$min(f_1) = min(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}) \tag{3.7}$$

Where $\boldsymbol{u}$ is given by:

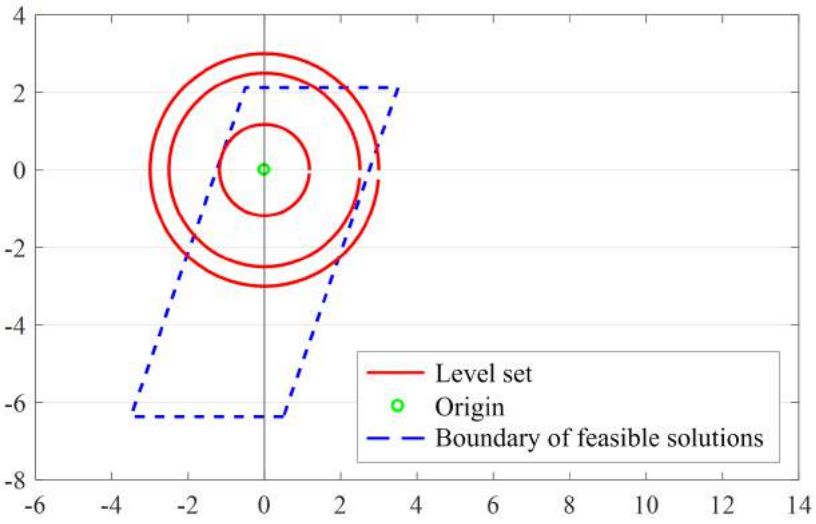


Figure 3.4: Region of feasible solutions and level sets of modified optimization problem (3.7).

$$\boldsymbol{u}=\mathbf{R}\boldsymbol{x}+\mathbf{Q}^{\mathbf{T}}\boldsymbol{y} \tag{3.8}$$

We want to mention here that the optimization problems, original (before applying QR factorization) and modified (after performing the QR factorization) are equivalent (both problems result in the same minimizing solution $\boldsymbol{x}_{min}$) but the functions $f$ and $f_1$ are not equal (they have different values at the minima). The modified optimization problem (3.7) is represented in Figure 3.4. We notice that after performing the transformation described in (3.8) the elliptical level sets of the cost function are transformed into circles and the rectangular region of feasible solutions is transformed into a parallelogram. The solution to the modified optimization problem (3.7) is reduced to finding the point of the parallelogram region closest to the origin, $\boldsymbol{u}_{min}$, efficiently calculated in 2D space (Algorithm 3.1). Finally, the minimum distance between two capsules is calculated from:

$$d_{min}=\sqrt{\boldsymbol{u}^{\mathrm{T}}{}_{min}\boldsymbol{u}_{min}+\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{y}^{\mathrm{T}}\mathbf{Q}\mathbf{Q}^{\mathbf{T}}\boldsymbol{y}}-\rho_1-\rho_2 \tag{3.9}$$

For a set of $k$ line-segments/capsules it can be noticed that:

1. Minimum distance calculations shall be performed mutually between any two capsules of the set, resulting in $O(k^2)$ complexity.

2. QR factorization of matrix $\mathbf{A_i}$ associated with a subset $i$ of two capsules can be enhanced for efficiency since different $\mathbf{A_i}$ have shared columns in their structure.

3. The vector $\boldsymbol{u}_{min}$ is being calculated in two dimensional space, while other algorithms calculate $\boldsymbol{x}_{min}$ in the three dimensional space.

4. Vectors operations in two dimensional space are less costly than operations in three dimensional space.

**Algorithm 3.1** Minimum distance calculation.

Input : $\boldsymbol{\mathcal{R}}$ region of feasible solutions
$(\mathbf{Q}, \mathbf{R})$ facorization matrices of $\mathbf{A}$
Output : $\boldsymbol{u_{min}}$ vector of coefficients

01 : Transform $\boldsymbol{\mathcal{R}}$ using the function $\boldsymbol{f(x)} = (\mathbf{R}\boldsymbol{x} + \mathbf{Q}^{\mathrm{T}}\boldsymbol{y})$
02 : **If** Origin is inside $\boldsymbol{\mathcal{R}}$ **then**
03 : $\boldsymbol{u_{min}} \leftarrow [0, 0]^{\mathrm{T}}$
04 : **else**
05 : **for** each boundary segment of $\boldsymbol{\mathcal{R}}$ **do**
06 : $\boldsymbol{c} \leftarrow$ point of segment closest to origin
07 : **If** first iteration **then**
08 : $\boldsymbol{u_{min}} \leftarrow \boldsymbol{c}$
09 : **else**
10 : **If** norm$(\boldsymbol{u_{min}})$ > norm$(\boldsymbol{c})$ **then**
11 : $\boldsymbol{u_{min}} \leftarrow \boldsymbol{c}$
12 : **end if**
13 : **end if**
14 : **end for**
15 : **end if**

5. We propose calculating $\boldsymbol{u}_{min}$ (in $\Re^2$) while taking advantage of the fact that the area of feasible solutions is a parallelogram.

By taking the aforementioned observations into account, the described algorithm can be applied efficiently for calculating minimum distance for a set of capsules[1].

### 3.1.3 Experiments

The performance was evaluated by comparing the proposed QR method with the method in [5] to compute the segment to segment minimum distance. Three comparison criteria were considered: (1) computational complexity, (2) execution time using C++ and (3) numerical precision. The results for the computational complexity of the algorithms (number of floating point operations – addition, multiplication, division and square root for QR) are in Table 3.1. For the second comparison criteria, a set of line-segments was randomly generated and the minimum distance (squared) between each two line segments of the set was calculated using the proposed QR method and the method in [5]. By implementing the algorithms in C++, results indicate that in terms of execution time the proposed QR method performed up to 25% faster than the method in [5], Figure 3.5 and Figure 3.6.

We noticed that method [5] can be modified by promoting some of the operations from $O(n^2)$ to $O(n)$. The results of those operations can be stored in memory and used later in the $O(n^2)$ part of the algorithm. In such a case, the $O(n^2)$ computational complexity of method [5] is reduced as shown in Table 3.1. Nevertheless, the proposed QR method is still more efficient in terms of execution time as shown in Figure 3.5 and Figure 3.6. The modified method in [5] is detailed in the Media materials (attachments) of [12], which includes C++ code in Appendix I (page 22) and a detailed breakdown

[1] A complete elaboration (including C++ implementation) on applying the proposed method efficiently for a set of capsules is available in the file attached to the Media materials of [12].

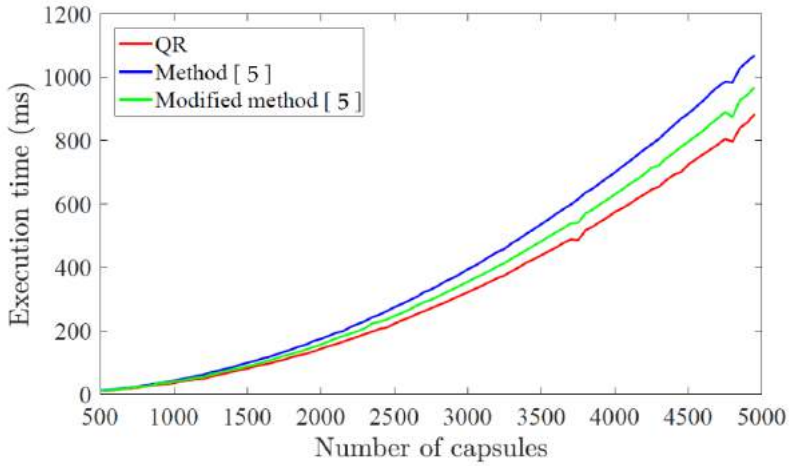


Figure 3.5: Execution time comparison for the proposed QR method, the method in [5] and the modified method in [5] as a function of number of line-segments/capsules of the set. Algorithms implemented in C++.

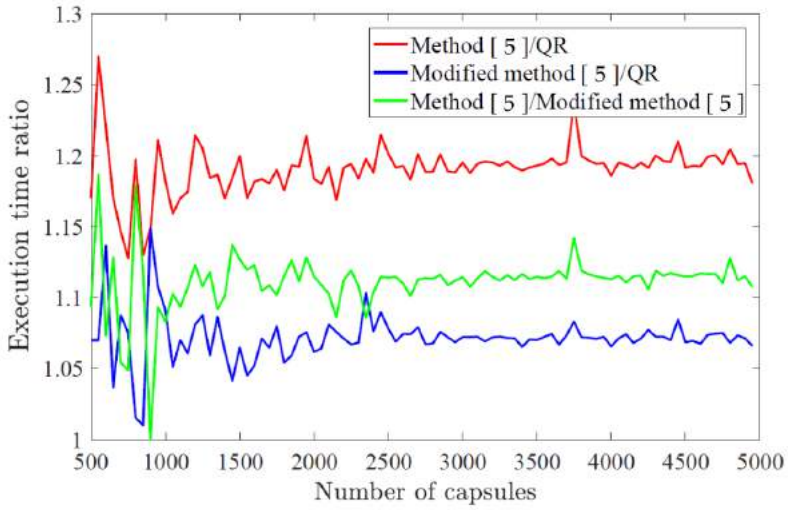


Figure 3.6: Execution time ratio (Method [5]/QR in red), (Modified method [5]/QR in blue) and (Method [5]/Modified method [5] in green) as a function of number of line-segments/capsules of the set. Algorithms implemented in C++.

Table 3.1: Computational complexity for the method in [5], the modified method in [5] and the proposed QR method.

| Method | Route | Computational complexity |
|---|---|---|
| Method in [5] | Least expensive | $\frac{56}{2}n^2 - \frac{56}{2}n$ |
| Method in [5] | Most expensive | $\frac{66}{2}n^2 - \frac{66}{2}n$ |
| Proposed QR method | Least expensive | $\frac{29}{2}n^2 - \frac{5}{2}n$ |
| Proposed QR method | Most expensive | $\frac{52}{2}n^2 - \frac{28}{2}n$ |
| Method in [5] modified | Least expensive | $\frac{40}{2}n^2 - \frac{24}{2}n$ |
| Method in [5] modified | Most expensive | $\frac{50}{2}n^2 - \frac{34}{2}n$ |

of the operation count in Appendix II (page 36). Considering the third comparison criteria, numerical precision, a group of 5000 line-segments/capsules has been generated randomly in 3D space. The $(x, y, z)$ coordinates of the start and end point of each segment are in the range $[-100, 100]$. The proposed QR method and the method in [5] are used to calculate the minimum distance between each couple of segments from the set. For each couple of segments, the relative error is defined as the ratio of the absolute value of difference between calculations using the two methods:

$$e = \frac{2|d^{qr}_{min} - d^{s}_{min}|}{d^{qr}_{min} + d^{s}_{min}} \tag{3.10}$$

Where $d^{qr}_{min}$ is the minimum distance calculated using the proposed QR method and $d^{s}_{min}$ is the minimum distance calculated using the method in [5]. Experimental tests resulted in a maximum value of the relative error of $1.059e^{-8}$, a minimum value of the relative error of $1.11e^{-16}$ and an average error of $4.87e^{-10}$. These values demonstrate that the error is negligible. The same test has been repeated for different groups of line-segments/capsules with different number of elements, Figure 3.7.

### 3.1.4 Summary

In this section, a novel method based on QR factorization for performing minimum distance calculations for a set of line-segments/capsules is proposed. Capsules demonstrated to be good solution to represent humans and objects in real environment having data from real sensors as input. Experimental results indicate that the proposed solution is more efficient than the existing most efficient method in literature. Such efficiency was measured in computational complexity (reduced number of floating point operations), execution time (up to 25% better) and numerical precision (the error is negligible).

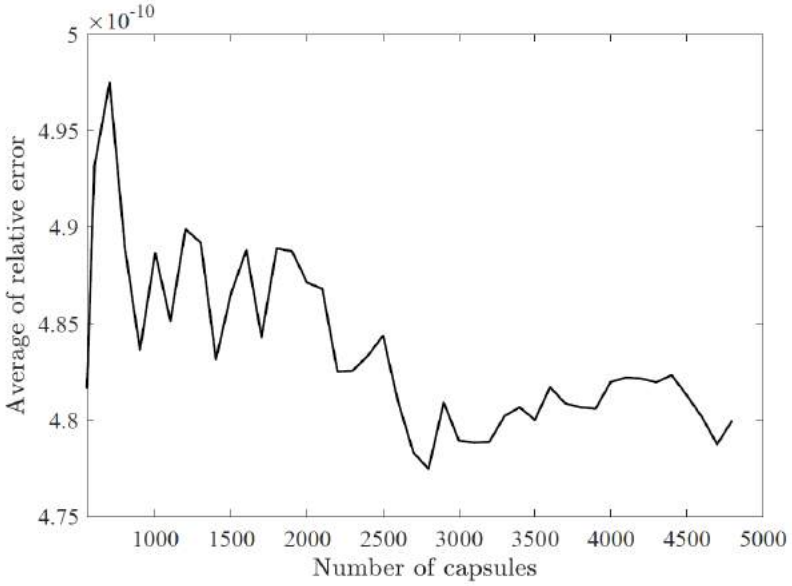


Figure 3.7: Average relative error between the method in [5] and the proposed QR method as a function of the number of line-segments/capsules of the set.

## 3.2 Human Pose Estimation

The avoidance motion carried out by a manipulator to prevent collisions with a human nearby relies heavily on his/her pose estimates. In this study, we opted to use Inertial Measurement Units (IMUs), shown in Figure 3.8 (a), coupled with a Sick laser scanner, shown in Figure 3.8 (b), for locating the human, and his/her configuration, around the robot. A total of five IMUs are used. One is attached to the human's chest, the other four are attached to the arms. This is suitable for our choice of representing the upper part of the human body by five capsules, one covering the torso and four covering the arms, Figure 3.8 (b). Given that in our robotic cell the robot is mounted on a table, as shown in Figure 3.9, with its motion restricted to a one meter level above the factory floor, there is no danger of collision between the robotic arm and the lower part of the human body, hence our negligence of representing the legs is justified. On the other hand, the robot (KUKA iiwa with 7 DoF) is described by three capsules representing the main links of the robot, Figure 3.8 (c). The configuration of the capsules covering the robot is obtained after calculating forward kinematics applied on the measured joint angles (acquired from the controller of the robot). After capturing the configuration of the capsules covering the human and the robot, the minimum distance is calculated using the method introduced in the previous section. In the following subsections, the methodology used to capture the configuration of the capsules covering the human body is described.

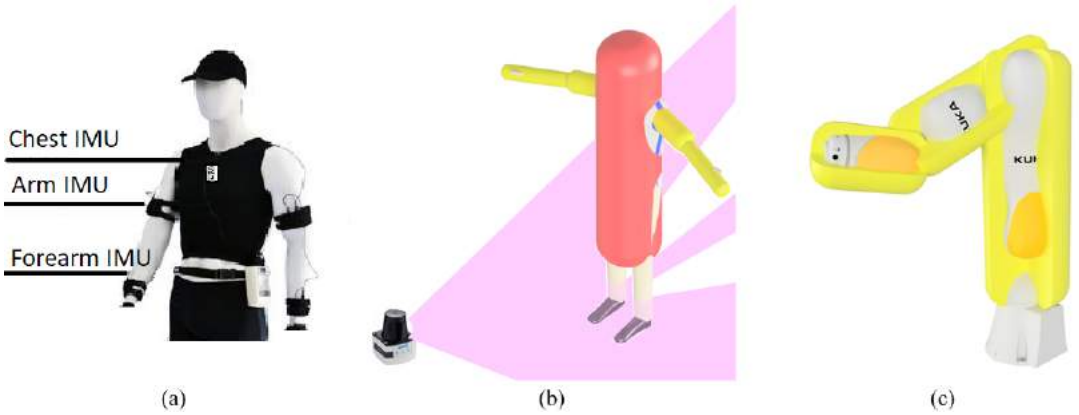


Figure 3.8: (a) IMUs attached to the upper body, (b) a human (represented by 5 capsules) standing in the scanning field of a laser scanner and (c) robot represented by 3 capsules.

### 3.2.1 Laser Scanner (Torso Position)

Data from the laser scanner mounted at the level of the coworker's legs, Figure 3.9, are utilized to define the relative position of the capsule representing the torso. Through TCP/IP connection, the sensor (SICK TiM5xx) provides the radius measurements along a scan-angle spanning 270 degrees, an angular precision of one degree and a maximum measurement radius of 8 meters. The methodology behind the proposed algorithm for calculating the torso position is divided into two stages:

Figure 3.9: Laser scanner mounted at the level of the coworker's legs in the base of the table holding the robot. If the robotic arm is installed on a mobile platform the solution is similar.

1. Data acquisition and filtering;
2. Calculating the minima and the position of the torso.

#### 3.2.1.1 Data Acquisition and Filtering

A TCP/IP server is implemented in MATLAB. It acquires the measurements from the laser scanner by decoding TCP/IP messages coming from the sensor. The result is stored in a two dimensional array, which relates the radius measurements with the corresponding scan-angle. Figure 3.10 shows the radius measurements along the scan-angle as acquired from the laser sensor corresponding to a scene where a human is standing in the scan field of the sensor. The radius measurements are clipped to 1400 mm away from the sensor. The contour of the legs appears in the plot as two minimas. It is noticed that the raw data (represented by a blue continuous line in Figure 3.10) are not smooth. To smooth out the curves two filtering methods are proposed: (1) temporal filtering and (2) spatial filtering (along the scan-angle).

#### 3.2.1.2 Temporal Filtering

To filter out the noise in radius measurements from the laser scanner a low-pass filter (LPF) is utilized:

$$r_{(\theta,t)} = \alpha r^{m}_{(\theta,t)} + (1-\alpha) r_{(\theta,t-dt)} \tag{3.11}$$

Where $r_{(\theta,t)}$ is the filtered value of the radius at angle $\theta$ and time $t$, and $r^{m}_{(\theta,t)}$ is the measurement value of the radius at angle $\theta$ and time $t$, $\alpha$ is a scalar from zero to one (adjusted during experiment for better response), and $dt$ is an update time interval

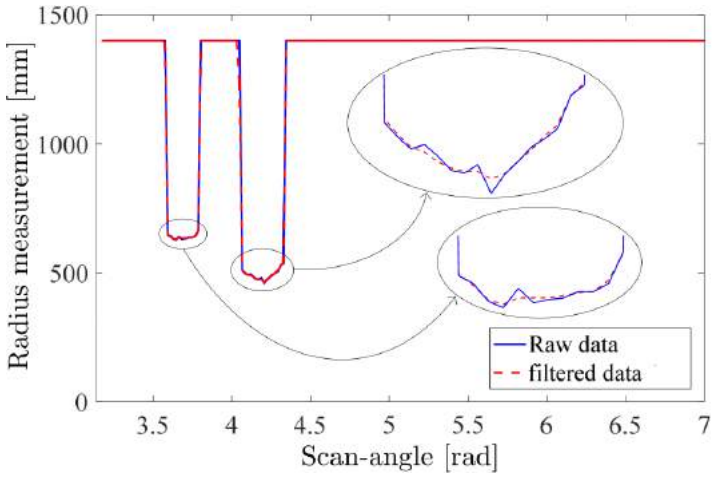


Figure 3.10: Raw data and LPF filtered of radius measurement with scan-angle, close ups show filtered data are smoother in the critical parts of the curves.

between two consecutive scans. The results of the application of the LPF filter are shown in dashed red line in Figure 3.10.

#### 3.2.1.3 Spatial Filtering

To smooth out short term fluctuations of the radius measurements along the scan-angle, a moving average (MVA) filter is used:

$$r^f_{(\theta,t)} = r^f_{(\theta-d\theta,t)} - \frac{r_{(\theta-nd\theta,t)}}{n} + \frac{r_{(\theta,t)}}{n} \tag{3.12}$$

Where $r^f_{(\theta,t)}$ is the value of the MVA at angle $\theta$ and time $t$, $d\theta$ is the angular resolution of the scanner, $r^f_{(\theta-d\theta,t)}$ the value of the MVA at angle $\theta - d\theta$ and time $t$, and $n$ is the number of averaging steps.

#### 3.2.1.4 Minimas and the Position of the Torso

Considering the plane of the floor, the xy position of the capsule representing the torso is calculated based on the position of the legs. The polar coordinates of the legs correspond to the minimas in the plot, as shown in Figure 3.11. To calculate the minimas, first we perform a mirror transform, on plot in Figure 3.11, with respect to the axis of the scan-angle $\theta$. In such a case, the minimas become peaks, which can be calculated by performing a peak analysis on the resulting curve. Consequently, the angle and the radius associated with the first leg $(\theta_1, r_1)$ and the second leg $(\theta_2, r_2)$ are acquired. Afterwards, the position vector of the first leg $\boldsymbol{x}_1$ in the Cartesian space is calculated:

$$\boldsymbol{x}_1 = \begin{bmatrix} cos(\theta_1) \\ sin(\theta_1) \end{bmatrix} (r_1 + \rho_l) \tag{3.13}$$

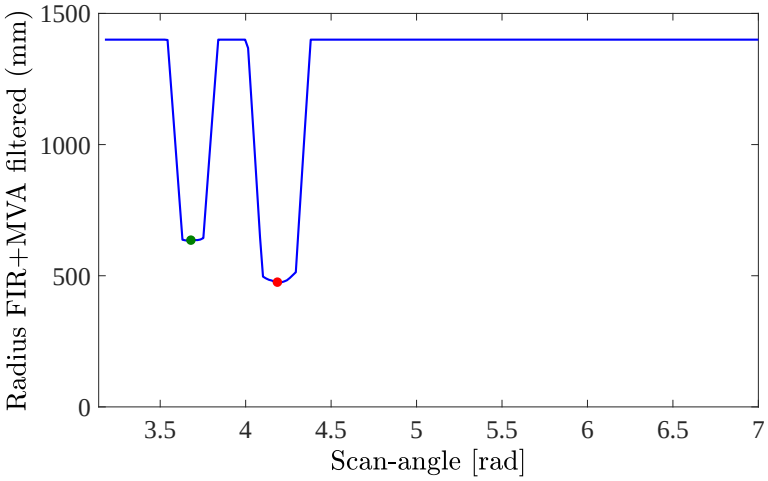


Figure 3.11: Filtered radius measurement with scan-angle. Minimas are marked with green and red dots representing the human legs.

Where $\rho_l$ is the radius of the leg. The vector of the Cartesian position for the second leg $\boldsymbol{x}_2$ is calculated in the same manner. The position of the torso capsule $\boldsymbol{x}_t$ can be approximately considered at the middle distance between the two legs:

$$\boldsymbol{x}_t = \frac{(\boldsymbol{x}_1 + \boldsymbol{x}_2)}{2} \tag{3.14}$$

Figure 3.12 shows a laser scan with a human in the scan field of the sensor. The scan field span of 270 degrees is satisfactory given that the sensor is mounted at the corner of the table holding the robot. Using the proposed algorithm, the position of the legs is detected, red and green dots. Using the legs coordinates the position of the capsule covering the torso is approximated, black circle in Figure 3.12.

### 3.2.2 IMU Sensors (Configuration of the Upper Body)

To capture the configuration of the capsules representing the human upper body five IMU sensors are used. One is attached to the chest (IMU 1) and the other four sensors are attached to the arms and the forearms (IMU 2, IMU 3, IMU 4 and IMU 5), Figure 3.13. Each capsule is described by two vectors and a radius. The vectors represent the position of the beginning and end of each capsule.

The quaternion measurements provided by the IMU sensors coupled with the geometric information from the human coworker's body (its dimensions) are used to estimate the position of the capsules covering the coworker's limbs in relation to robot base. The procedure for performing the calculations is divided into the following steps:

1. Calibration phase;
2. Calculating rotations of limbs with respect to base frame of the robot;
3. Calculating the position of the limbs' capsules with respect to the base frame of the robot.

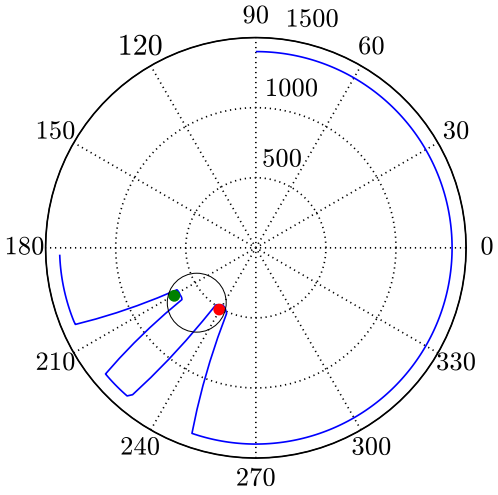


Figure 3.12: Human legs detected in the laser scan field. Red and green dots represent the legs and the black circle represents the projection of the capsule covering the torso.

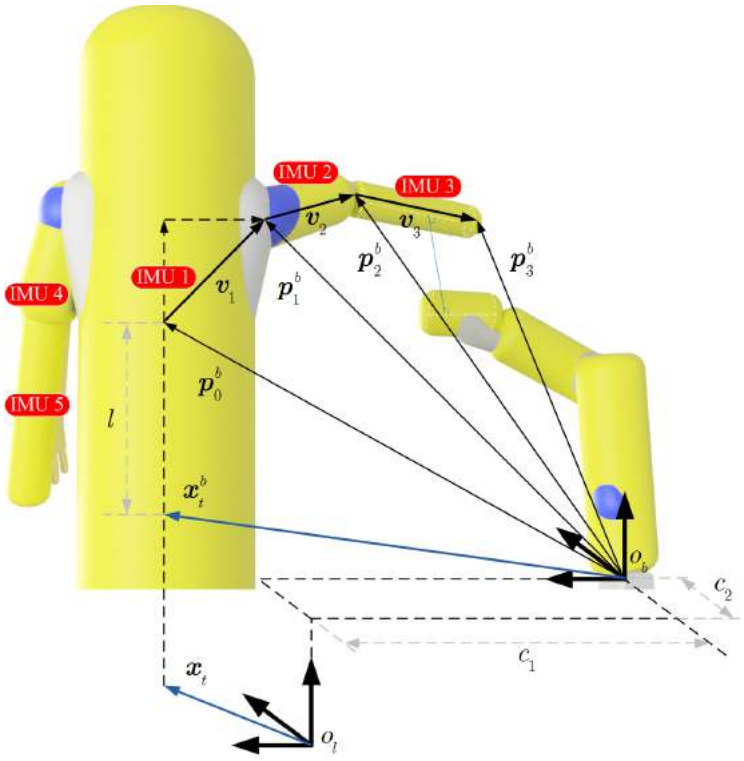


Figure 3.13: Minimum distance between two capsules.

#### 3.2.2.1 Calibration

Each IMU measurement gives its orientation, in quaternion $\boldsymbol{w}_{imu}^{ref}$, with respect to a pre-defined reference frame[2]. For the collision avoidance algorithm the rotations shall be described in relation to the robot base frame using the quaternion $\boldsymbol{w}_{imu}^{b}$. To calculate $\boldsymbol{w}_{imu}^{b}$, the rotation quaternion from the reference frame to the robot base frame $\boldsymbol{w}_{ref}^{b}$ shall be calculated. This is achieved by performing an initial calibration phase. Consequently, the IMUs are placed in a predefined orientation with respect to the robot base before the system is initiated. In such a case, the initial rotation quaternion $\boldsymbol{w}_{b}^{imu,init}$ of the IMU frame with respect to robot base is already known. By reading the initial IMU measurement $\boldsymbol{w}_{imu,init}^{ref}$, the quaternion $\boldsymbol{w}_{b}^{ref}$ is calculated:

$$\boldsymbol{w}_{b}^{ref} = \boldsymbol{w}_{imu,init}^{ref} \boldsymbol{w}_{b}^{imu,init} \tag{3.15}$$

As a result of the calibration phase, the rotation quaternion from the reference frame of the IMU to the base of the robot $\boldsymbol{w}_{ref}^{b}$ is calculated as the inverse of the quaternion $\boldsymbol{w}_{b}^{ref}$.

#### 3.2.2.2 Orientation of the Limbs

After calculating the orientation of the reference frame with respect to robot base $\boldsymbol{w}_{ref}^{b}$, the quaternion measurements describing the IMU orientation with respect to base frame of the robot $\boldsymbol{w}_{imu}^{b}$ is calculated from the orientation measurement of the IMU $\boldsymbol{w}_{imu}^{ref}$, as the following:

$$\boldsymbol{w}_{imu}^{b} = \boldsymbol{w}_{ref}^{b} \boldsymbol{w}_{imu}^{ref} \tag{3.16}$$

#### 3.2.2.3 Position of the Limbs

Five vectors on the coworker's body are considered, Figure 3.13. Due to the symmetry of the human body, the following elaboration and equations are given for the left half of the coworker's body. For the other half, an identical methodology is applied. For the left half of the body three vectors are considered:

1. Vector $\boldsymbol{v}_1$: a vector that spans from the chest up to the left shoulder;
2. Vector $\boldsymbol{v}_2$: a vector that spans the left arm of the coworker, from the shoulder up to the elbow;
3. Vector $\boldsymbol{v}_3$: a vector that spans the length of the left forearm, from the elbow up to the wrist.

The IMUs are mounted firmly on the body of the coworker according to:

- The chest's IMU (IMU 1 in Figure 3.13) is mounted such that the x axis of the IMU is pointing vertically down when the coworker is standing straight up. The

[2]In $\boldsymbol{w}_{imu}^{ref}$ the superscript $ref$ stands for reference frame, and the subscript $imu$ stands for the frame of the IMU.

z axis of the sensor is in the Sagittal plane, coming out of the chest. The y axis of the sensor is horizontally positioned in the Coronal (Frontal) plane. In such case the coordinates of the vector $\boldsymbol{v}_1^{imu_1}$ as described in upper chest's IMU frame are $[\ -d_1 \quad d_2 \quad 0\ ]$ for the left shoulder and $[\ -d_1 \quad -d_2 \quad 0\ ]$ for the right shoulder. Where $d_1$ is the length on coworker's body taken vertically from the chest up to the shoulder and $d_2$ is the width of the coworker's shoulders divided by two.

- The upper arm's IMU is mounted such that the x axis of the IMU is aligned with the upper arm's length, in such case the coordinates of the vector $\boldsymbol{v}_2^{imu_2}$ as described in upper arm's IMU frame are $[\ d_3 \quad 0 \quad 0\ ]$. $d_3$ is the length of the upper arm measured from the shoulder to the elbow.

- The forearm's IMU is mounted such that the x axis of the IMU is aligned with the forearm's length. In such case the coordinates of the vector $\boldsymbol{v}_3^{imu_3}$ as described in forearm's IMU frame are $[\ d_4 \quad 0 \quad 0\ ]$. $d_4$ is the length of the forearm measured from the elbow to the wrist.

The previous vectors are rotated back to the base frame of the robot using:

$$\boldsymbol{v}_i^b = \boldsymbol{w}_{imu_i}^b \boldsymbol{v}_i^{imu_i} \left(\boldsymbol{w}_{imu_i}^b\right)^{-1} \tag{3.17}$$

Where $\boldsymbol{v}_i^{imu_i}$, $i = 1, 2, ...5$, is the vector of human part described in the $i^{th} - imu$ frame. $\boldsymbol{w}_{imu_i}^b$ is the quaternion describing the rotation of $i^{th} - imu$ in relation to base frame of the robot Eq (3.16) and $\left(\boldsymbol{w}_{imu_i}^b\right)^{-1}$ is the complex conjugate of $\boldsymbol{w}_{imu_i}^b$.

The position vector of the shoulder point in the base frame of the robot $\boldsymbol{p}_1^b$ is calculated:

$$\boldsymbol{p}_1^b = \boldsymbol{v}_1^b + \boldsymbol{p}_0^b \tag{3.18}$$

Where $\boldsymbol{p}_0^b$ is the position of the chest point with respect to the base frame of the robot, given that the coworker is standing up all the time the xy position of the coworker's chest point is the same as the position of the torso acquired from the laser scanner. If we denoted $l$ to the height of the coworker's chest point from the xy plane of the base frame of the robot, then $\boldsymbol{p}_0^b$ is calculated:

$$\boldsymbol{p}_0^b = \begin{bmatrix} \boldsymbol{x}_t^b \\ l \end{bmatrix} \tag{3.19}$$

Where $\boldsymbol{x}_t^b$ is the xy position of the torso of the coworker with respect to the base frame of the robot, calculated by transforming the estimation of the torso position (acquired from laser scanner measurement) $\boldsymbol{x}_t$ into the base frame of the robot. Given that the frame of the laser scanner is parallel to the base frame of the robot then the vector $\boldsymbol{x}_t^b$ is given by:

$$\boldsymbol{x}_t^b = \boldsymbol{x}_t + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{3.20}$$

Where $c_1$ is the x coordinate of the origin $O_l$ of the scanner's-measurement-frame in the base frame of the robot, and $c_2$ is the y coordinate of the origin $O_l$ of the scanner's-measurement-frame in the base frame of the robot. The origin $O_l$ and the dimensions $(c_1, c_2)$ are shown in Figure 3.13.

The position vector of the elbow point in the base frame of the robot $\boldsymbol{p}_2^b$ is given by:

$$\boldsymbol{p}_2^b = \boldsymbol{v}_2^b + \boldsymbol{p}_1^b \tag{3.21}$$

Accordingly, the position vector of the wrist point in the base frame of the robot $p_3^b$ is:

$$\boldsymbol{p}_3^b = \boldsymbol{v}_3^b + \boldsymbol{p}_2^b \tag{3.22}$$

### 3.2.3 Summary

In this section a method for capturing the configuration of the human upper body is presented. Laser scanner and IMU sensors are used for this purpose. The position of the human torso (approximated by an upright capsule) is acquired by processing measurements from the laser scanner. The configuration of the capsules covering the human arms are acquired by processing the measurements from the IMU sensors. In addition, a method for calibrating the IMU reference frame with respect to robot base frame is presented. The method proposed in this section is important for the application of our collision avoidance algorithm (in the next section).

## 3.3 Collision Avoidance Algorithm

This section introduces the subject of manipulator's on-line collision avoidance into a real industrial application implementing typical sensors and a commonly used collaborative industrial manipulator, KUKA iiwa. In the proposed methodology, the minimum distance and relative velocity between the human and the robot is calculated, when human/obstacles are nearby the concept of hypothetical repulsion and attraction vectors is used. By coupling this concept with a mathematical representation of robot's kinematics, a task level control with collision avoidance capability is achieved. Consequently, the off-line generated nominal path of the industrial task is modified on–the-fly so the robot is able to avoid collision with the coworker safely while being able to fulfill the industrial operation. To guarantee motion continuity when switching between different tasks, the notion of repulsion-vector-reshaping is introduced. Tests on an assembly robotic cell in automotive industry show that the robot moves smoothly and avoids collisions successfully by adjusting the off-line generated nominal paths.

### 3.3.1 A Review on Collision Avoidance Algorithms

Collision avoidance is a major topic in robotics research, especially in the new context of collaborative robotics. Owing to its importance, numerous studies approaching collision avoidance had been presented. In the following we tried to list the major studies about the subject. In potential field (PF)-based methods the robot is in a hypothetical vector field influenced by two types of forces. Forces of attraction that guide the robot towards the goal, and repulsion forces that repel it away from obstacles. Subjected to these forces the robot finds its way to the goal while avoiding collisions. Unlike its predecessors the artificial potential field [38] was the first of its kind that treated the problem of collision avoidance at the low-level control, which is best suited for achieving real-time response. Another method that is based on a similar principle is the 3 Dimensional (3D) force field method [53]. In this method each link of the robot is surrounded by a virtual elliptical volume. When the obstacle penetrates into the ellipsoid, a hypothetical repulsive force is generated, repelling the robot away from it and avoiding collision.

Based on the PF principle, an approach to collision avoidance and trajectory planning was proposed in [54]. This method requires as an input a preliminary trajectory, generated with another method, repulsion poles are used for representing obstacles and an attraction pole is utilized for guiding the robot toward the goal. The attraction pole moves along the preliminary trajectory and the robot will follow it while being repelled away from obstacles. This method is suitable for generating collision free paths that are as close as possible to a predefined trajectory. Recently, in [55], a depth space approach for collision avoidance between a robot and coworker was presented. The study describes an improved implementation of the artificial potential field method in which an estimation of obstacle's velocity was taken into consideration when computing the repulsion vector. This is an advancement to the original artificial potential field that utilizes only information about minimum distances for calculating the repulsion vectors. Also, the paper proposed a novel approach for estimating obstacle's velocity.

In [50] the authors presented the skeleton algorithm, a framework for self-collision

avoidance for humanoid robots. In this algorithm a skeletal representation of the robot is proposed, based on this representation the points of the robot closest to collision are obtained through minimum distance calculations between different segments of the skeletal structure, then repulsion forces are calculated and converted to control commands used to control the robot.

In [56] the authors proposed a control architecture for a collaborative robotic cell that performs assembly works, the cell is provided with collision avoidance module, the robot and the workspace are represented in a virtual environment where the minimum distance is computed and control commands are generated.

While the artificial potential field is inspired by electric field phenomena, other approaches, inspired by other types of fields, were proposed. The circular fields method, inspired by electromagnetism, was investigated in [57, 58]. One of the advantages offered by this method over the artificial potential field is that it is immune to getting stuck in local minima, a drawback that might appear in the artificial potential field.

Other researchers drew inspiration from fluid mechanics. In this context, [59] utilized the stream lines of potential flow for attaining collision free paths. The Circular Theorem from fluid dynamics was implemented for computing the stream lines of the potential flow. Similarly, in [60] collision free paths are attained by superimposing elementary flows. Doublets are used to model cylindrical obstacles inserted in a uniform flow. In such a way, collision free paths are attained after calculating the stream lines of the flow. Analogous to circular fields, the methods based on fluid dynamics are also immune to dead-lock (getting stuck in local minima). Nevertheless, almost all studies based on fluid mechanics techniques are dedicated to applications of collision avoidance for mobile robots.

In [61] the authors proposed the application of biharmonic potential functions for collision free path calculation, inspired by the phenomena of stress distribution in materials. In this method, the navigation zone is considered to be a continuous solid with elasticity properties, obstacles are voids inside the solid, and the goal is modelled by a pressurized cavity. The distribution of stresses inside the material is calculated and from this distribution a collision free path is established.

Other researchers have approached robot collision avoidance as an optimization problem. In [41] the authors utilized the square of a norm of an error vector as objective function. This error was defined from the difference between the end-effector's velocity and the desired velocity. The constraints are formulated from limits on (1) joints angles, (2) joints velocities and (3) the likelihood of collision. The optimization problem is solved in real-time using the logarithmic barrier method, and the computed velocity is used to command the robot.

A motion planner that takes into consideration obstacle avoidance is presented in [62]. This planner employs both the potential field and a genetic optimization algorithm. In this method the planning process is divided into two procedures. In the first the artificial potential fields method was employed to find a collision free path for the end-effector. In the second procedure, collision free configurations were calculated using Genetic Optimization techniques. The method was validated by performing simulations on a virtual robot with twelve DoF.

In [63] the authors described an algorithm for collision avoidance for redundant manipulators. The algorithm operates at the kinematics (joint-velocities) level. At

first, the minimum distance between the robot and the obstacle is calculated, if closer than a predefined safety distance, a motion component is assigned to the part of the robot closest to the obstacle, repelling the robot away from collision. The novelty in their approach was in the way the repulsion motion was defined, or what they call "one dimensional operational space" approach. In such approach, the repulsion component is projected on the direction aligned with the closest distance between the robot and the obstacle. This way of defining the repulsion motion gave their algorithm better immunity against singularities. Nonetheless, the proposed method was built around the fact that the end-effector's task is the primary task, or the task with the highest-level priority, while collision avoidance was treated in the null space of the Jacobian associated with the end-effector. This will cause the manipulator to fail in avoiding collision with the obstacle in some situations. For example, when there is a contradiction between the end-effector's task, assigned the highest-level priority, and the collision avoidance task, assigned a lower-level of priority. For solving this drawback the authors proposed to re-invoke a high-level path planning algorithm, so that a new collision free path can be calculated. Another solution for this problem, more efficient and not requiring re-planning, is proposed in [64]. In this solution a coefficient is introduced into the control equation, and by changing the value of this coefficient the priority level between tasks can be switched smoothly.

In [65] the authors proposed an inverse kinematics solver that takes into consideration joint limits and collision avoidance. The inverse kinematics problem was formulated as a minimization of the square of the error between the goal position and the end-effector position. The minimization is subjected to (1) constraints due to joints limits, and (2) constraints due to restrictions on minimum distance between the obstacle and the robot. Fiacco and McCormick algorithm was used to solve the optimization problem. The method was tested on a 5 DoF robotic manipulator.

Probabilistic RoadMaps (PRM), [66] is a powerful method for generating collision free paths for robots with high degrees of freedom in non-dynamic environments. This method is composed of two phases. The first phase is called the learning phase, it is performed offline, and used for stochastic generation of a *roadmap* of the scene. The second phase is called the query phase. In this phase, paths are queried, and the *roadmap* is searched for a feasible path. While PRM is a powerful method, it suffers two major problems. The first of which is that it is unsuitable for real-time implementation. This is due to the high computational cost of the method, particularly for robots with high degrees of freedom. The second drawback comes from the fact that the resulting trajectories generated by this planner are not dynamically optimized for direct implementation on the robot. So much so, PRM is more suited for the offline global planning over real-time collision avoidance implementation. Nevertheless, due to the advancement in processing power of contemporary computers, and despite the high computational cost of PRM, a recent study [67] reported success in implementing PRM for real-time collision free path planning in dynamic environments. Based on PRM, in [68] the authors proposed the dynamic roadmaps, a high level algorithm for planning dynamic paths in changing environment, and more suitable for real-time collision avoidance applications. This method was used successfully in [69], where the authors presented a collaborative human-robot system with integrated collision avoidance capability.

In [45] the authors presented a collision avoidance system in which the control strategy proposed searches for a motion-direction of the end-effector that guarantees a collision free path between the robot and the coworker. In case this direction exists, the end-effector is commanded to move in that direction. Otherwise the robot is stopped, while waiting for the coworker to move from its way.

Other researchers took alternate approach towards solving collision avoidance, this approach requires a complete knowledge of obstacle's trajectory. Though restrictive, these methods are appealing when performing collision avoidance between different robotic manipulators. For example, in multi-robot cells or for dual-arm manipulators, where the trajectories of the manipulators are known a priori. A study that utilizes this knowledge for achieving collision avoidance between two end-effectors is presented in [70]. In this study the end-effectors are modelled as spheres, using their pre-calculated trajectories, a collision map is constructed. Afterwards, an iterative time shifting algorithm is applied and the time delay required to achieve collision avoidance is calculated. In other words, collision free operation is achieved by performing time-rescheduling on motion commands before sending them to controllers. This approach was developed further in [71], where an Advanced Collision Map is introduced, the method can be applied for collision avoidance between two manipulators rather than their end-effectors only. In [72] the method was generalized to perform collision avoidance between any number of manipulators.

A study dedicated to collision avoidance between two manipulators for industrial applications is in [73]. The problem was addressed by dividing the work space of the manipulators into a shared work area, accessible to both manipulators, and an external work area accessible to only one manipulator. The authors added a processing layer into the control structure, in which point-to-point control commands are processed before being sent to the controllers. As consequence, the manipulators are allowed to operate in their own external work area at any time. However, the presence of one of the manipulators inside the shared work area will deny access to the other manipulator, causing it to wait until the shared work area is free from the other manipulator.

For redundant manipulators self induced collisions could occur. This problem is treated in [74], where RoBE (Representation of Body by Elastic Elements) is presented as a method for preventing robot self collisions. In this method each link is covered by a fictitious elastic element, whenever the elements touch, a force is generated and collision avoidance is achieved.

### 3.3.2 Collision Avoidance for Industrial Manipulators

Industrial robots are traditionally working inside fences, isolated from humans. The ability to have robots sharing the workspace and working side-by-side with human coworkers is a key factor for the materialization of the Industry 4.0 concept. The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario where robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining coordination, dexterity and cognitive capabilities of humans with the robots' accuracy, agility and ability to produce repetitive work. For example, robots can help humans in carrying and manipulating sensitive/heavy objects safely [75] and

positioning them precisely by hand-guiding [16]. In this scenario the robot can play the role of a force magnifier while moving compliantly according to the haptic feedback from the human.

Reaching the goal of developing/creating safe collaborative robots will allow a greater presence of robots in our society, with a positive impact in several domains, including industry [22]. Nowadays, industrial collaborative robots, which are not operating inside fences, do not have autonomy to perceive its unstructured and time-varying surrounding environment, nor the ability to avoid collisions with human coworkers in real-time while keeping the task target defined by the off-line generated paths. On the contrary, they stop when a predefined minimum separation distance is reached. Due to this issue, the full potentialities of collaborative robots in industrial environment are not totally explored. The increasing demand by industry for collaborative robot-based solutions makes the need for advanced collision avoidance strategies more visible. To have them working safely alongside humans, robots need to be provided with biological-like reflexes, allowing them to circumvent obstacles and avoid collisions. This is extremely important in order to give robots more autonomy and minimum need for human intervention, especially when robots are operating in dynamic environment and interacting/collaborating with human coworkers [76]. The requirements for safe collaborative robots, including physical human-robot interaction (HRI), are detailed in [77], where collision avoidance is listed as a factor, among others, which is important for human-robot safety. The new standard ISO 10218 and the technical specification TS 15066 define the safety requirements for collaborative robots [78]. Apart from industrial domain and human-robot collaboration, collision avoidance is also being investigated for aerospace applications, including robotic arms mounted on space maneuverable platforms [79] and aerial manipulators mounted on drones [80].

The subject of collision avoidance for robotic manipulators has captured the interest of researchers for decades. In the pioneering work of Khatib [38], a real-time obstacle avoidance approach based on the classical artificial potential field (PF) concept is introduced. In PF-based methods, the robot is in a hypothetical vector field, and its motion is influenced by forces of attraction that guide the robot towards the goal and forces of repulsion that repel it away from obstacles. Subjected to these forces the robot finds its way to the goal while avoiding collisions. An improved implementation of the PF method in which an estimation of obstacle's velocity is taken into consideration when computing the repulsion vector is proposed in [81]. PF-based robot self-collision avoidance has been studied, as well as the development of collision avoidance techniques for redundant robots [82]. A distributed real-time approach to collision avoidance considering not only the robot tool centre point over the objects in the cell, but also the body of the tool mounted on the robot flange is in [83]. A passivity-based control scheme for human-robot safe cooperation is proposed in [84]. A collision free trajectory generating method for a robot operating in a shared workspace in which a neural network is applied to create the way points required for dynamic obstacles avoidance is proposed in [85]. In [86], the authors presented a method for calculating collision free optimal trajectories for robotic manipulators with static obstacles. The proposed algorithm takes into consideration the maximum limits of jerk, torques, and power for each actuator. Tests have been carried out in simulation in a PUMA 560 robot. In [87], it is presented a collision avoidance algorithm between robotic manipulators and mobile

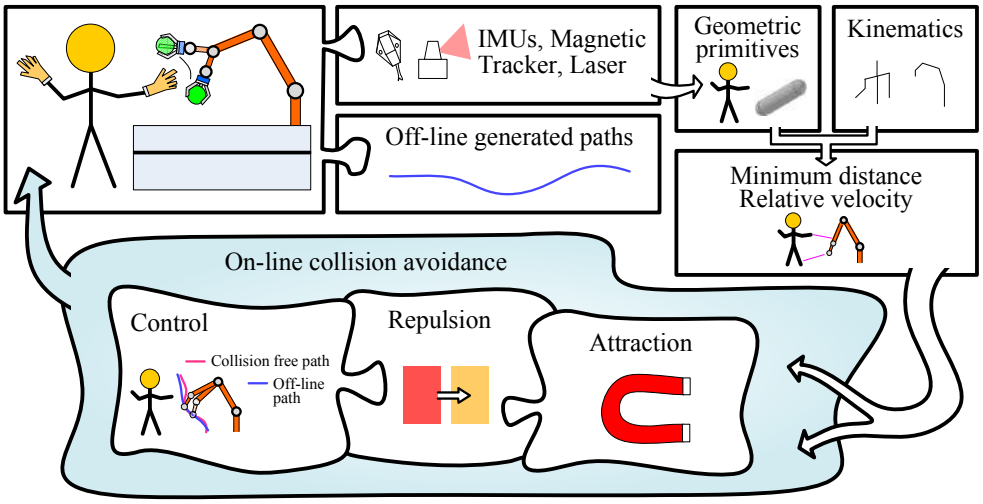


Figure 3.14: Proposed framework for on-line human-robot collision avoidance.

obstacles validated in simulation environment. Using the variation principle, it is proposed a path planner for serial manipulators with high degrees of freedom operating in constrained work spaces where the planner produces monotonically optimal collision free paths [88]. Based on fuzzy rules [89], the authors presented a method for resolving internal joint angles in redundant manipulators. The method allows the EEF to follow the desired path, while the internal motion manifold is used to perform other objectives including collision avoidance with surrounding obstacles.

However, the number of existing studies dedicated to on-line human-robot collision avoidance for manipulators in industrial setups is very limited, and when existing, results are presented in simulation environment. Some of these studies, especially the ones with more direct industrial application, approach collision avoidance by stopping the robot or reducing its velocity when a human reaches a given distance threshold [90]. An interesting work defines four safety strategies for workspace monitoring and collision detection: the system alerts the operator, stops the robot, moves the robot away, or modifies the robot's trajectory from an approaching operator [91].

### 3.3.3 Proposed Approach

In this thesis, an industrial task is defined by the off-line generated robot paths (task primitives), which can be programmed by a Computer-Aided Design (CAD) software, or by using more sophisticated methods including Programming by Demonstration (PbD) [92]. Consequently, a safe human robot interaction is achieved through real-time modification of the off-line generated paths, Figure 3.14. In this scenario of shared workspace, the human coworker focuses on the collaborative task he/she is performing rather than the potential danger from the robot. Using external sensors (IMUs, a magnetic tracker and a laser scanner) the pose of the human body is captured and approximated by capsules. The robot is represented by three capsules. The analytical minimum distance between capsules representing the robot and the human(s) is calculated using the method in section 3.1. The human-robot minimum distance and relative

velocity are used as inputs for the proposed collision avoidance controller, where hypothetical attraction and repulsion vectors attract the robot towards the goal/target while repelling it away from obstacles. We also introduce the notion of repulsion-vector-reshaping to avoid control discontinuity. By coupling these concepts with a mathematical representation of robot's kinematics we can achieve a task level control with smooth collision avoidance capability. The proposed framework is tested in three different experiments, including a real use case for assembly in automotive industry using real sensors and a collaborative industrial manipulator. Results indicate that the robot reacts smoothly by modifying its off-line generated paths to avoid collision with the human coworker. Consequently, our study is the first of its kind (according to our knowledge) that satisfies all the following points combined:

1. In our study a real industrial robot (not experimental) is used for performing a typical industrial task;

2. The proposed method for performing the collision avoidance is tailored for industrial use, by combining an off-line path of the EEF (important for industrial applications) with an on-line reactive collision avoidance (required for dynamic collision avoidance);

3. In our method a human coworker is moving freely around the robot, the whole configuration of his/her upper body is captured using real sensors. In this regard, our study differs from the previously presented studies, which mainly based on simulations, while the remaining few (that approached human-robot collision avoidance) utilized an experimental robot, and mostly vision systems which suffer from occlusion, in addition other studies focused on the collision avoidance itself, without showing results in a real industrial operation;

4. Unlike other studies, we realized that in a real industrial scenario, the control shall switch between different operation modes (as shown later in Figure 3.22 and Algorithm 3.4), this can lead to repulsion action discontinuity, we resolved this issue by proposing the repulsion-vector-reshaping (described in Algorithm 3.2).

### 3.3.4 Problem Specifications

Two major problems in on-line human-robot collision avoidance are related with the reliable acquisition of the human pose in unstructured environments and the difficulty in achieving smooth continuous robot motion while generating collision avoidance paths. For capturing the configuration of the human, the method in the previous (Human Pose Estimation) section is implemented. Concerning the difficulty in achieving collision free and smooth continuous robot motion, this problem is particularly visible in an industrial setup where the control algorithm switches between different controllers depending on the task-in-hand. In summary, several challenges can be pointed out:

1. Accurate definition of humans/obstacles and robot pose in space using geometric primitives, and calculation of the minimum distance between them;

2. Achieving reliable autonomous human-robot collision avoidance in which the robot adapts the off-line generated nominal paths while keeping the task goal/target.

In such a case, instead of stopping or reducing robot's velocity when humans are nearby, the robot has to continue its motion while avoiding the humans/obstacles;

3. The control strategy shall produce continuous motion of robot's reaction when it adjusts the path to avoid collision. This continuity shall be guaranteed even when switching between different controllers;

4. Industrial applications require high-performance control in terms of motion accuracy and agility;

5. Collision free robot motion should be possible and reliable in the entire working volume of the robot.

Experiments demonstrated the ability of the proposed solution to achieve on-line human-robot collision avoidance materialized in the following contributions:

1. Reliable and smooth human-robot collision avoidance in which the robot adapts the off-line generated nominal paths (defined in the initial robot program) while keeping the task target. The robot finds a way to get around the human(s)/obstacles when they are nearby. Human-robot minimum distance and relative velocity are used as inputs to the implemented collision avoidance algorithm;

2. Successfully applying on-line collision avoidance on a real industrial collaborative robot performing industrial assembly tasks in collaboration with a human coworker.

### 3.3.5 Collision Avoidance Strategy

Hypothetical attraction and repulsion vectors attract the robot towards the goal/target (defined by the off-line generated nominal paths) while repelling it away from human(s)/obstacles. By coupling this concept with a mathematical representation of robot's kinematics we can achieve a task level control with collision avoidance capability.

#### 3.3.5.1 Repulsion

A vector $\boldsymbol{v_{cp.rep}}$ acts on the point of the robot closest to the obstacle (CP) repelling it away from collision. This vector is defined considering a magnitude $v_{rep.mod}$ (calculated from a base repulsion amplitude $v_{rep}$) and a direction $\boldsymbol{s}$:

$$\boldsymbol{v_{cp.rep}} = v_{rep.mod}\boldsymbol{s} \tag{3.23}$$

The direction of the repulsion vector $\boldsymbol{s}$ is taken to be aligned with the line segment associated with the minimum distance:

$$\boldsymbol{s} = \frac{\boldsymbol{r_1} - \boldsymbol{r_2}}{|\boldsymbol{r_1} - \boldsymbol{r_2}|} \tag{3.24}$$

Where $\boldsymbol{r_1}$ is the position vector of the point of the robot closest to the obstacle and $\boldsymbol{r_2}$ is the position vector of the point of the obstacle closest to the robot. For calculating

the base repulsion amplitude $v_{rep}$ we propose to superimpose the repulsion due to the minimum distance ($v_{rep1}$) and the repulsion due to the relative velocity between human and robot ($v_{rep2}$), so that $v_{rep} = v_{rep1} + v_{rep2}$. Here, $v_{rep1}$ is calculated from the minimum distance $d_{min}$:

$$v_{rep1} = \begin{cases} k_1 \left( \frac{d_0}{d_{min} - d_{cr}} - 1 \right), & if\ d_{min} - d_{cr} < d_0 \\ 0 & if\ d_{min} - d_{cr} \geq d_0 \end{cases} \tag{3.25}$$

Where $k_1$ is a constant, $d_0$ is an offset distance around the obstacle's capsule, it specifies the area around the obstacle where the repulsion vector is activated, and $d_{cr}$ is a critical distance below which the robot is not allowed to be near the human. To enhance the responsiveness of the robot, we propose a dynamical reshaping of the size of the area of influence around the obstacle $d_0$, such that the value of $d_0$ increases when the relative velocity between the robot and the obstacle increases:

$$d_0 = \begin{cases} d_1 - c_v v_{rel} & v_{rel} < 0 \\ d_1 & v_{rel} \geq 0 \end{cases} \tag{3.26}$$

Where $v_{rel}$ is the human-robot relative velocity, $c_v$ is a constant and $d_1$ is the minimum value of the area of influence around the obstacle. For $v_{rep2}$ we have:

$$v_{rep2} = \begin{cases} -c\, k_2\, v_{rel} & v_{rel} < 0 \\ 0 & v_{rel} \geq 0 \end{cases} \tag{3.27}$$

Where $k_2$ is a damping constant and $c$ is a coefficient that takes into consideration the proximity of the obstacle from the robot:

$$c = \begin{cases} 1 & d_{min} \leq l_1 \\ \frac{1 + cos(\pi \frac{d_{min} - l_1}{l_2 - l_1})}{2} & l_1 < d_{min} < l_2 \\ 0 & l_2 \leq d_{min} \end{cases} \tag{3.28}$$

Where $l_1$ and $l_2$ are constant distances that define the range around the robot where the damping force is activated. The intuition of using $c$ is that obstacles far away from the robot shall not affect robot's motion since that they do not pose any risk of collision.

The modified repulsion magnitude $v_{rep.mod}$ is calculated from $v_{rep}$ according to Algorithm 3.2. For complex industrial collaborative operations, the collision avoidance controller is typically embedded in a state machine, where the collision avoidance functionality is activated/deactivated according to the tasks being performed. In such a case, discontinuity could appear when calculating the repulsion action. As an example, if the controller is switched (from the collision avoidance deactivation to the collision avoidance activation) while the coworker is near the robot, discontinuity appears. In such a case, a high magnitude of the repulsion vector will act on the robot suddenly. To solve this problem, the repulsion action is proposed to be time dependent, by introducing the concept of repulsion-vector-reshaping coefficient $\gamma$, such that when the control scheme is switched the repulsion magnitude is allowed to increase monotonically starting from zero up to its stable value. In the Algorithm, *now* is a function returning

**Algorithm 3.2** Modified repulsion vector (magnitude).

1: **for each** time step $\triangle t$ **do**
2: **if** controller switched **then**
3: $t_0 = now$
4: **end if**
5: $t = now - t_0$
6: $\gamma = 1 - exp(-t/\tau)$
7: $v_{rep.mod} = \gamma v_{rep}$
8: **end for**

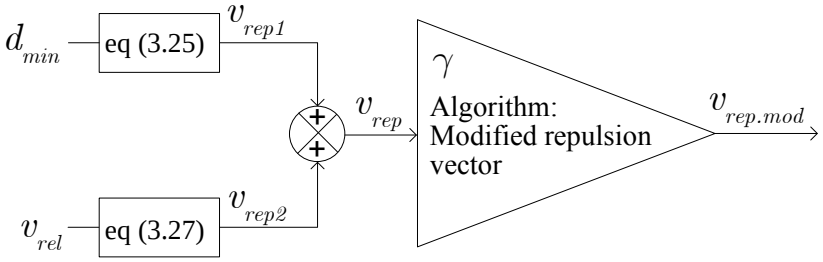


Figure 3.15: Block diagram showing the proposed method for calculating the magnitude of the modified repulsion vector.

the current time, $\tau$ is a time constant that can be calculated from $v_{max}/(5a_{max})$, where $v_{max}$ and $a_{max}$ are the maximum curvilinear velocity and acceleration of the EEF used during collision avoidance, respectively.

Figure 3.15 shows a block diagram illustrating the proposed method for calculating the modified repulsion $v_{rep.mod}$ and its relationship to $v_{rep.1}$ and $v_{rep.2}$.

#### 3.3.5.2 Attraction

An attraction velocity vector $\boldsymbol{v_{e.att}}$ attached to the EEF guides the robot towards the goal/target, Figure 3.16. This vector is a function of the error $\boldsymbol{e}$ between EEF's position $\boldsymbol{p_e}$ and the goal position $\boldsymbol{p_g}$ (defined in the off-line generated nominal paths):

$$\boldsymbol{e} = \boldsymbol{p_e} - \boldsymbol{p_g} \tag{3.29}$$

The attraction velocity is calculated from a proportional term ($\boldsymbol{\psi}_p$) and a quasi-integral term ($\boldsymbol{\psi}_i$):

$$\boldsymbol{v_{e.att}} = \beta(\boldsymbol{\psi}_p + \boldsymbol{\psi}_i) \tag{3.30}$$

Where $\boldsymbol{\psi}_p$ is a pure proportional term:

$$\boldsymbol{\psi}_p = -\mathbf{K_p}\boldsymbol{e} \tag{3.31}$$

In which $\mathbf{K_p}$ is the proportional coefficient. The quasi-integral term $\boldsymbol{\psi}_i$, Algorithm 3.3, prevents the quasi-integral from accumulating when the human-robot distance is less than a predefined safety distance $d_0$. In the Algorithm, $d_{min}$ is the human-robot minimum distance and the integral term is calculated numerically using a simple Euler

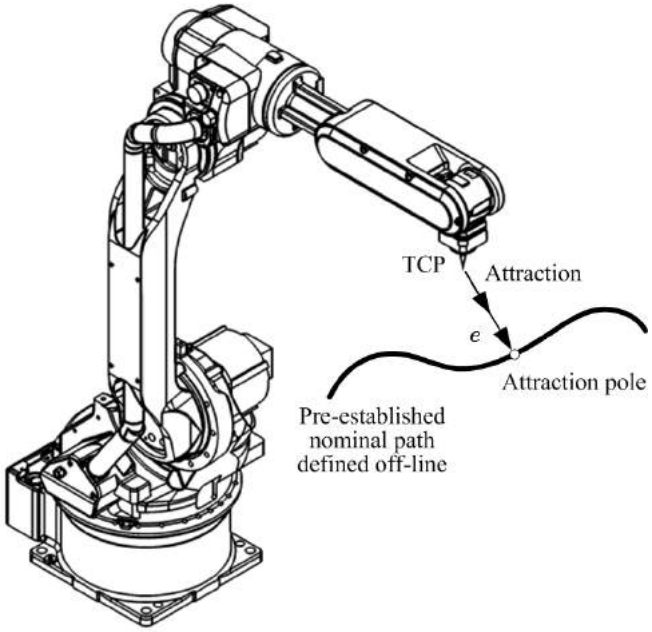


Figure 3.16: The nominal path curve defined off-line, the attraction pole, and the error vector.

**Algorithm 3.3** Integral term of the attraction vector.

1: **for each** time step $\triangle t$ **do**
2: **if** $d_{min}$-$d_{cr}$>$d_0$ **then**
3: $\boldsymbol{\psi}_i = \boldsymbol{\psi}_i - \mathbf{K}_i \int_t^{t+\triangle t} \boldsymbol{e} dt$
4: **else**
5: $\boldsymbol{\psi}_i = \boldsymbol{\psi}_i$
6: **end if**
7: **end for**

scheme (more sophisticated Runge-Kutta methods could also be used). The term $\beta$ is used to reduce the magnitude of the attraction vector. This term has the effect of detaching the EEF gradually from the goal when the human coworker is closer to the robot:

$$\beta = \left( \frac{2}{1 + e^{-\left(\frac{d_{min} - d_{cr}}{d_0}\right)^2}} - 1 \right) \tag{3.32}$$

Figure 3.17 shows a block diagram illustrating the proposed method for calculating the attraction vector.

#### 3.3.5.3 Controller

The robot is controlled at the joint velocity level. The repulsion and attraction vectors are considered velocity vectors in which the repulsion velocity is calculated from (3.23), and the attraction velocity at the EEF $\boldsymbol{v_{e.att}}$ is calculated from (3.30). Calculating

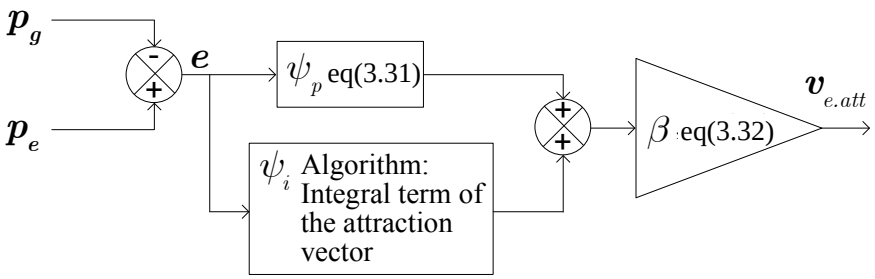


Figure 3.17: Block diagram showing the proposed method for calculating the attraction vector. Term $\psi_i$ is calculated using the Algorithm 3.3, which is used to avoid windup problem.

the overall angular velocities of the joints requires superimposing the angular velocities due to repulsion and attraction. Thus, the angular velocities due to $\boldsymbol{v_{cp.rep}}$ that acts at CP is calculated using the Damped Least Squares (DLS) [35]:

$$\dot{\boldsymbol{q}}_{\boldsymbol{rep}} = \mathbf{J}_{\mathbf{cp}}^{\mathbf{T}}(\mathbf{J}_{\mathbf{cp}}\mathbf{J}_{\mathbf{cp}}^{\mathbf{T}} + \lambda^2\mathbf{I})^{-1}\boldsymbol{v}_{\boldsymbol{cp.rep}} \tag{3.33}$$

Where $\dot{\boldsymbol{q}}_{\boldsymbol{rep}}$ is the joint velocities vector due to the repulsion action, $\mathbf{J_{cp}}$ is the partial Jacobian associated with CP on the robot, $\lambda$ is a damping constant, and $\mathbf{I}$ is the identity matrix.
The angular velocities due to $\boldsymbol{v_{e.att}}$ that act at the EEF are calculated from:

$$\dot{\boldsymbol{q}}_{\boldsymbol{att}} = \mathbf{J}_{\mathbf{e}}^{\mathbf{T}}(\mathbf{J}_{\mathbf{e}}\mathbf{J}_{\mathbf{e}}^{\mathbf{T}} + \lambda^2\mathbf{I})^{-1}\boldsymbol{v}_{\boldsymbol{e.att}} \tag{3.34}$$

Where $\dot{\boldsymbol{q}}_{\boldsymbol{att}}$ is the joint velocities vector due to the attraction action and $\mathbf{J_e}$ is the Jacobian associated with the EEF. Thus, the total angular velocities of the joints sent to the robot:

$$\dot{\boldsymbol{q}}_{\boldsymbol{total}} = \dot{\boldsymbol{q}}_{\boldsymbol{att}} + \dot{\boldsymbol{q}}_{\boldsymbol{rep}} \tag{3.35}$$

### 3.3.6 Experiments

Experiments are conducted in three main tests:

1. Test 1: Detailed in 3.3.6.1, in this test the human arm acts as an obstacle for the robot that is performing a straight line path (off-line generated nominal path);
2. Test 2: Detailed in 3.3.6.2, in this test the human approaches the robot from the side while the robot is stopped at a predefined home position;
3. Test 3: Detailed in 3.3.6.3, representing an industrial collaborative assembly operation for automotive industry in which the human coworker approaches the robot to place a sticker in a car door card while the robot is inserting trim clips in the same door card.

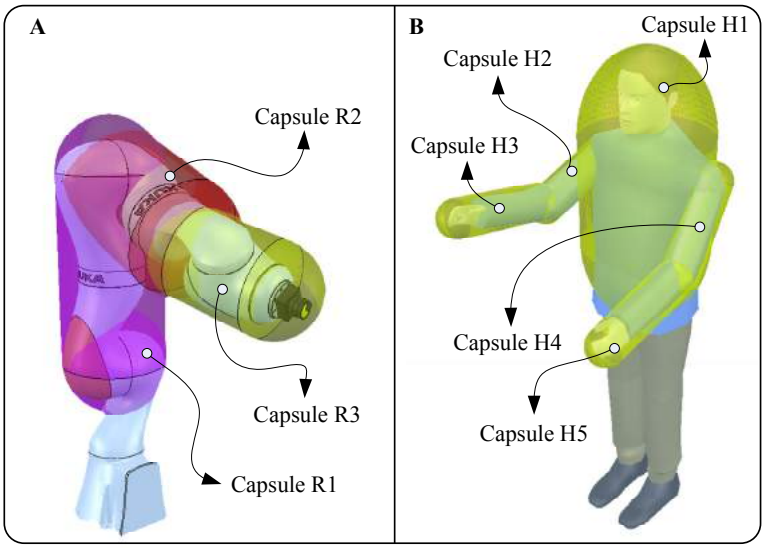


Figure 3.18: Robot (A) and human (B) represented by capsules.

### Setup and Data Acquisition

The three experimental tests were performed using different sensors for capturing the human pose in space. In test 1 and test 2, the proposed solution was performed with Polhemus Liberty magnetic tracking sensors attached to the human upper body (arm, forearm and chest) to acquire 6 DoF pose (position and orientation) of each body part in space. In test 3, the method proposed in the previous section (Human Pose Estimation) is used for capturing the human body pose from five IMUs (Technaid MCS) attached to the arms/forearms and the chest, and a laser scanner (SICK TiM5xx) at the level of the legs. An external computer Intel Core i7 with 32 GB of RAM running MATLAB® was used for performing the required computations: sensor data acquisition, capsules configuration calculation, minimum-distance and relative-velocity calculation, collision avoidance algorithms, and robot control using the KUKA Sunrise Toolbox (KST) detailed in section 3.5.

### Human and Robot Representation

The human is represented by five capsules, Figure 3.18, four capsules used to cover the right/left upper arm and forearm, while the fifth capsule is used to cover the torso up to the head. The robot (KUKA iiwa with 7 DoF) is represented by three capsules, Figure 3.18. Capsule R3 also incorporates the tool attached to the robot. The pose of the capsules are defined by applying the forward kinematics on the joint angles acquired from the controller of the robot.

#### 3.3.6.1 Test 1

Figure 3.19 shows the results for test 1. The human forearm, represented by a capsule, is extended and acts as an obstacle. The robot is moving on a straight line path

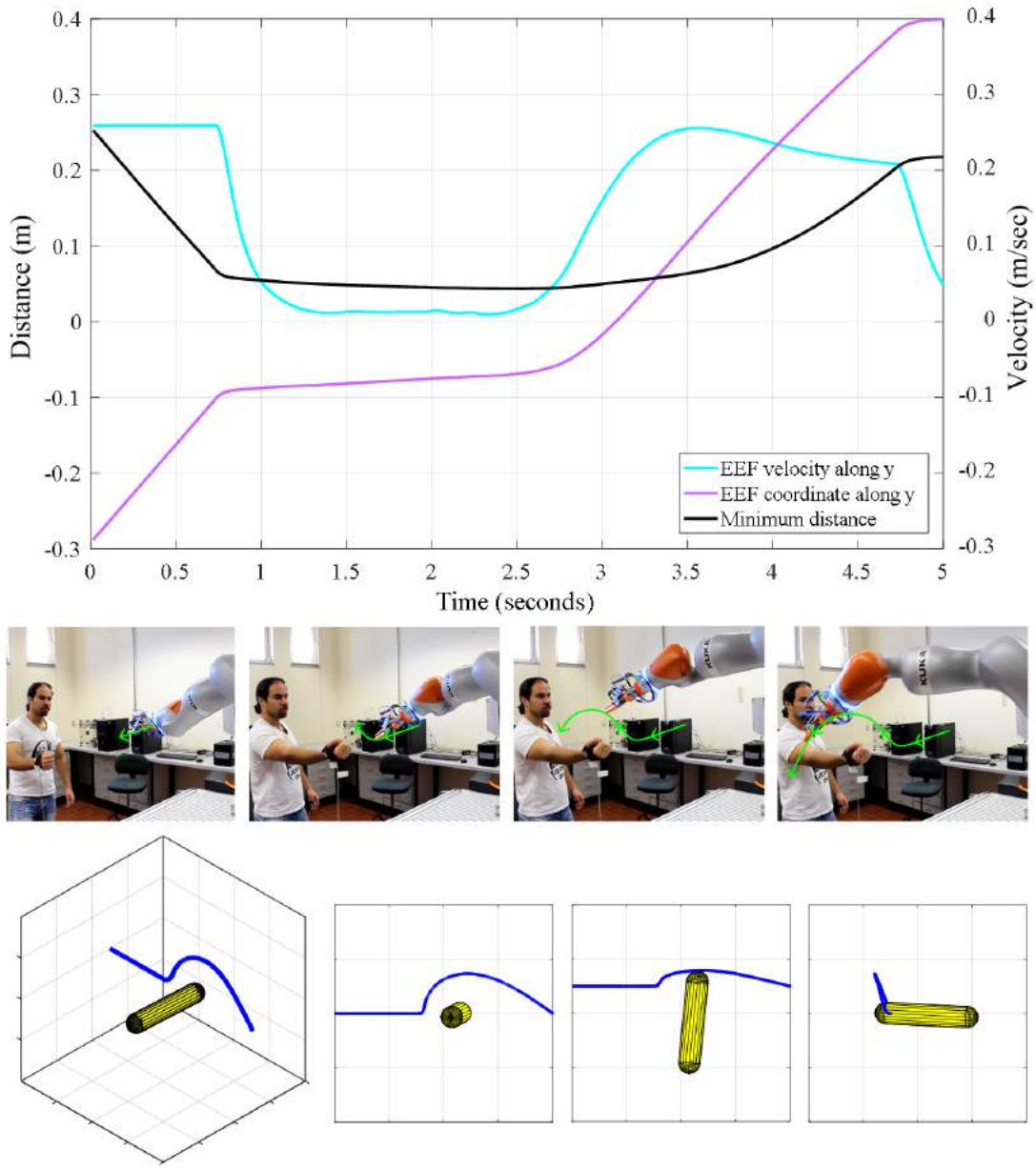


Figure 3.19: Test 1 results. Minimum distance, EEF velocity and position along y axis (top). Snapshot of collision avoidance testing and collision avoidance path in 3D and 2D space (middle and bottom).

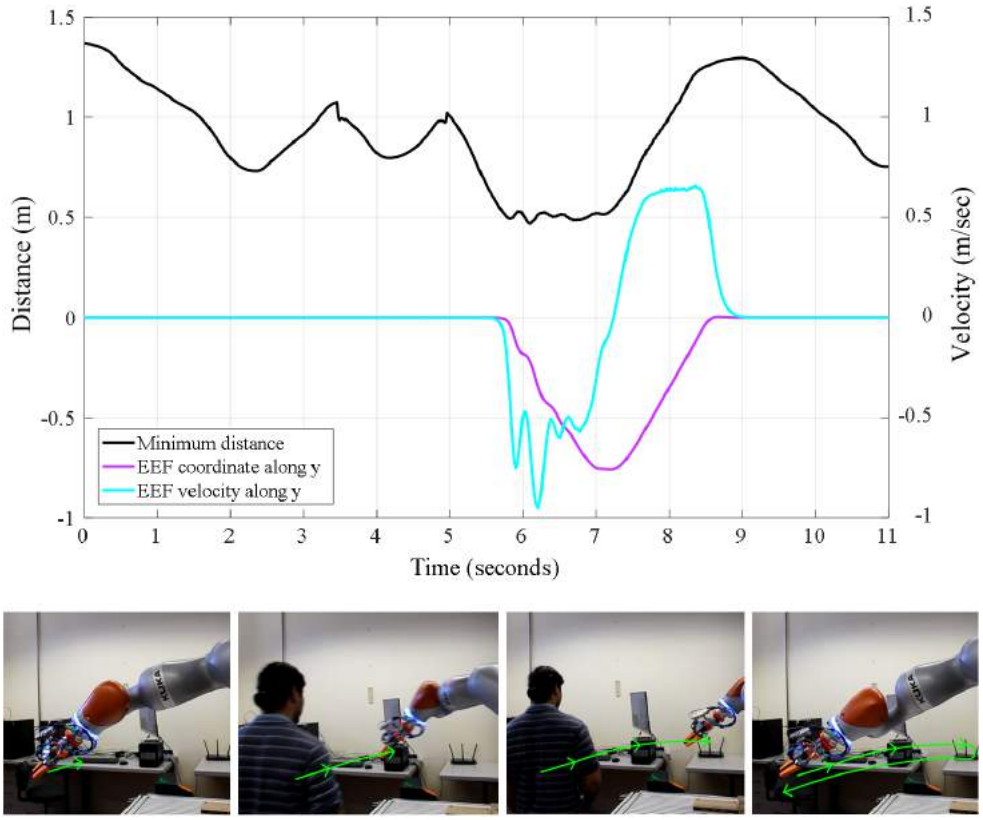


Figure 3.20: Test 2 results. Minimum distance, EEF velocity and position along y axis (top). Snapshot of collision avoidance testing (bottom).

(nominal path defined off-line) along the y direction of its base frame. While moving along the straight line, the minimum distance between the human-arm and the robot decreases. Consequently, the robot adapts the nominal path smoothly circumventing the human arm. At the top of Figure 3.19, the graph shows the minimum distance, the velocity of the EEF and its position in Cartesian coordinates along y axis. These quantities are reported as function of time. We can notice from the plot that at the beginning the human arm is in a resting position and the robot is moving with a constant velocity of about 0.26 m/sec, along y direction, towards the human arm. When the robot EEF approaches the human arm the minimum distance decreases to a minimum of about 5 centimeters. The EEF velocity is constant until a threshold minimum distance is reached. In such scenario the EEF velocity decreases to start circumventing the obstacle and then accelerates to reach a velocity close to the nominal velocity of 0.26 m/s. It can be concluded that the robot manages to avoid collision with the coworker successfully and the collision avoidance controller smoothly reacts to avoid collision while reaching the task target.

#### 3.3.6.2 Test 2

In test 2 the human approaches the robot from the side while the robot is stopped at a pre-defined home position. As the human approaches the robot the human-robot minimum distance decreases and the robot reacts in an agile-smooth behaviour to avoid collision. At the top of Figure 3.20, the same quantities presented for test 1 show that at the beginning the human starts walking towards the robot, when the minimum distance reaches 0.5 meters, the robot reacts to avoid collision. When the human goes away the robot returns to the initial home position. The robot successfully avoided collision as in the snapshots at the bottom of Figure 3.20 and in the video segment in [93].

#### 3.3.6.3 Test 3

Flexible manufacturing, and industrial assembly processes in particular, present several challenges due to the unstructured nature of an industrial environment. Some tasks are more suited to be executed by humans, others by robots, and others by the collaborative work between human and robot. The ability to have humans and robots working side-by-side will bring enormous efficiency benefits to flexible manufacturing. However, this scenario is challenging, due to the requirement of having the robot avoiding collisions with the coworker in real-time, allowing him/her to focus on the manufacturing tasks and not on the potential danger from the robot side. In this context, we tested the proposed system, in test 3, in a collaborative robotic cell for automotive industry where a human coworker approaches the robot to place a sticker in a car door card while the robot is inserting trim clips in the same door card, Figure 3.21 (a video segment showing the experiment is available in [6]). This flexible collaborative task allows the coworker to manage his/her working time and sequencing of operations since he/she is free to place the sticker in the door card at any time and devotes attention to other tasks that he/she has to take care of. Meanwhile the robot continues inserting the trim clips in the door card by using its force feedback to compensate for deviations in the door card positioning. When the human coworker approaches the robot it adapts the nominal path to avoid collisions in a smooth way while keeping its task. For this test, the pre-established nominal path is divided in 3 sub-path segments, Figure 3.22 (A). In segment 1 and 3 (green line) the collision avoidance control is activated (collision avoidance (CA) paths) while in segment 2 (red and blue line) is deactivated. This is because this path is defined to be the working path where the robot is inserting the trim clips at relative reduced velocity. Starting from a given home position coincident with the beginning of segment 1 and the tool centre point (TCP) the system behaves as in Algorithm 3.4. In Figure 3.22 (B) the robot and goal point move along segment 1 so that an error vector is established. If the human is detected in the safety zone the collision avoidance is activated and the goal point stops moving, Figure 3.22 (C). When the human is not in the safety zone the robot returns back to the goal point that starts moving with the robot, Figure 3.22 (D). When the robot reaches segment 2 the collision avoidance is deactivated, Figure 3.22 (E). All the users indicate that the system does not appear to be dangerous virtue to the collision avoidance motion which is perceived as smooth and natural.

**Algorithm 3.4** Collision avoidance - collaborative robotic cell in automotive industry (reads together with Figure 3.22)

**for each** time step **do**
  **if** human is not detected in the safety zone **then**
    **if** CA path **then**
      Goal point moves along path segment
      Error vector generated between TCP and the goal point, sub-figure (B)
      Attraction velocity generated from the error vector
      **if** goal point reaches the segment end **then**
        Goal point stops moving
        The robot TCP reaches the goal point
      **end if**
    **else**
      Collision avoidance controller is deactivated
      Controller to insert trim clip is activated
    **end if**
  **else** human is detected in the safety zone
    **if** CA path **then**
      Goal point stops moving
      Integral term stops accumulating
      A repulsion velocity $v_{rep}$ acts in the robot, sub-figure (C)
    **else**
      Collision avoidance controller is deactivated
      Controller to insert trim clip is activated, sub-figure (E)
    **end if**
  **end if**
**end for**


Figure 3.21: Test 3. Collaborative robotic cell for car door card assembly (video segment showing the experiment is in [6]).

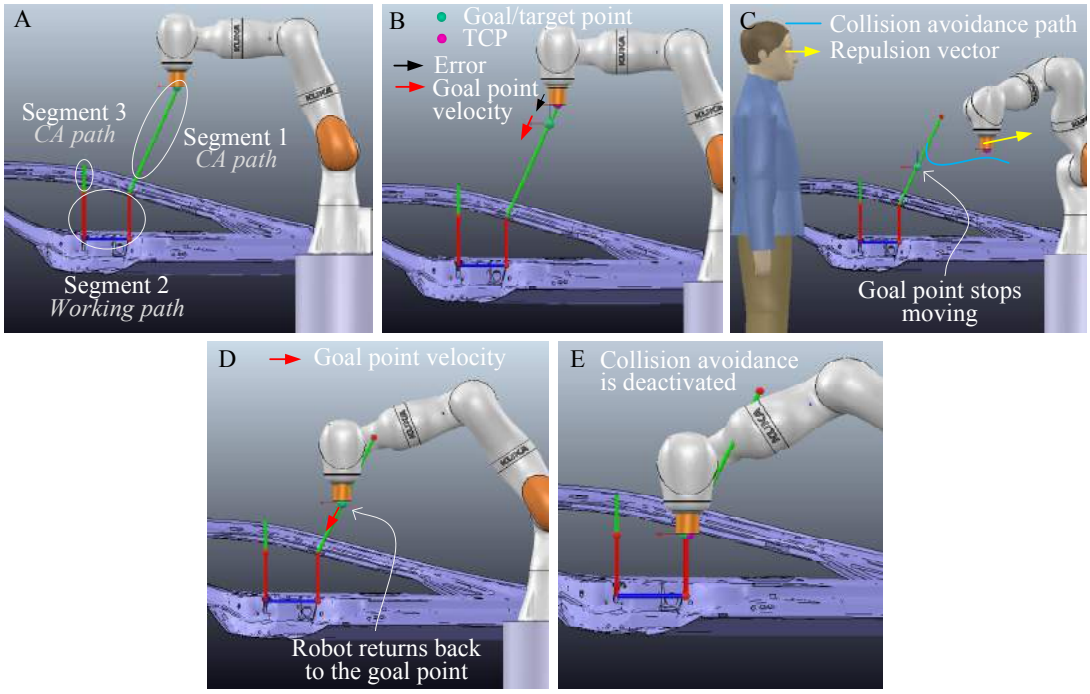


Figure 3.22: Pre-established sub-path segments for test 3. This process is detailed in the algorithm 3.4.

### 3.3.7 Summary

This section presented a novel method for human-robot collision avoidance for collaborative robotics tailored for industrial applications. The collision avoidance controller demonstrated on-line capabilities to avoid collisions while the robot continues working by keeping the task target. The concept of repulsion-vector-reshaping was introduced to guarantee the continuity of the generated motion when switching between controllers. Experiments indicated that the robot reaction to avoid collisions is well perceived by the coworker, smooth, natural and effective.

## 3.4 Collision Avoidance Algorithm Using Newton's Method

In this section the application of Newton method for the problems of collision avoidance and path planning for robotic manipulators is investigated, especially for robots with high Degrees of Freedom (DoF). The proposed algorithm applies on the potential fields method, where the Newton technique is used for performing the optimization. Furthermore, a computationally efficient symbolic formula for calculating the Hessian with respect to joint angles is deduced, which is essential for achieving real-time performance of the algorithm in high DoF configuration spaces. As compared to classical gradient descent method (with repulsion and attraction vectors) this implementation of Newton's method offers various advantages: it is mathematically elegant, enhances the performance of motion generation, eliminates oscillations, does not require tedious gains tuning, and gives faster convergence to the solution. Finally the method is validated successfully in simulation for 15 DoF planar manipulator.

### 3.4.1 Introduction

Redundancy is a term used for referring to robots that have more degrees of freedom than necessary to complete a task. Extra degrees of freedom offer better flexibility and can be used to achieve multiple objectives besides to the required task [94], as such several studies have been dedicated to the problem of controlling redundant robots [95, 82]. In cluttered environments with obstacles, redundancy plays an important role, allowing the robot to reach the goal while avoiding obstacles. Yet, for achieving real-time path planning capacity for high DoF manipulators in unstructured environments, it is required to merge the redundancy control frameworks with some notion of attraction/repulsion vector field [55, 96]. In such a case, task-level priorities are achieved through projections on the null space of the Jacobian [64], or by using more sophisticated projection criteria [97]. This method for formulating collision avoidance is proved to be computationally efficient. Nevertheless, in many cases this implementation requires a predefined trajectory of the end-effector (EEF) and the repulsion/attraction vectors are custom engineered with many parameters to tune, as shown in the previous section. Moreover, this method suffers from oscillation problems, which become more evident for increasing number of obstacles and for higher DoF robots, this was noticed in our implementation in the previous section, yet a simple (but not very elegant) solution is implemented by filtering. A method with better immunity to local minima problem is the Probabilistic RoadMaps (PRM) proposed in [66]. This method is composed of two phases. The first phase is called the learning phase, it is performed for stochastic generation of a roadmap of the scene (robot and its surroundings). The second phase is called the query phase. In this phase, paths are queried, and the roadmap is searched for a feasible path. While PRM is a powerful method, it suffers two major problems. The first of which is due to computational cost, where according to the times reported in [66] it can be concluded that for high DoF robots PRM is unsuitable for implementation in control loops with fast update rates. The second drawback comes from the fact that the resulting trajectories generated by this planner are not smooth and require processing before implementation on the robot [98].

In this section, the application of Newton method on the problem of collision avoidance for hyper redundant robotic manipulators is investigate. In such a case, the problem is reformulated as a second order optimization problem. The Newton method is utilized for performing the optimization, as a result generating collision-free and smooth paths. The application of Newton method in robotics offers various advantages. As reported in the optimization literature [99] using the Newton method for optimization offers: (1) better numerical stability, (2) faster convergence and (3) eliminates numerical oscillations that might be induced during first order optimization. In robotics this translates into generating smoother paths and faster execution, as reported in [100, 101], where the modified Newton method is applied for the problem of mobile robot navigation, where the robot is approximated by a point in a plane and the potential field is an explicit function of the configuration variables, in such a case calculating the Hessian analytically is possible as noted in [102]. While the Newton method has been described in literature for the problem of mobile robot navigation, this method has not been investigated yet for the problem of collision avoidance in robotic manipulators. This comes from the fact that the resulting potential fields, which are functions of Cartesian space variables, are implicit functions of configuration space variables, with high non-linearity due to the nature of the complex transform function between the spaces [103]. This makes finding a closed form solution for the Hessian in joint space a challenging task. In this section, a systematic method for applying Newton technique for robotic manipulators, at kinematic level, is addressed. We present a formula for efficient calculation of the Hessian matrix, by deducing a relationship between the Hessian in Cartesian space and the resulting Hessian in joint space. This allows to speed up the algorithm and makes it applicable for real-time implementation. In addition to the aforementioned advantages the proposed solution contributes in the following:

- Bigger optimization steps: In Newton method bigger optimization steps can be used, resulting in faster execution rates, as compared with the gradient method;
- Faster convergence: Newton method offers better performance in terms of convergence rate than the gradient descent [104];
- Elegant formulation: The proposed method is mathematically elegant, eliminates the need for tedious tuning of control parameters, and can be applied on hyper redundant manipulators. The method does not require pre-planned trajectory and keeps the number of control parameters to a minimum.

The proposed method can be used either for real-time control of the manipulator to avoid collisions with obstacles, or it can be used for off-line path planning. The application of the presented method extends easily to computer animations and to motion generation for kinematic chains in virtual environments with obstacles.

### 3.4.2 Mathematical Formulation

For achieving a robot motion that drives the EEF towards the goal, while avoiding collision with obstacles, the potential fields method by Khatib [38] is considered. In this method an attraction potential field $u^{att}$ attracts the EEF towards the goal. This

potential is expressed in Cartesian space as a function of EEF position. In addition, repulsion potential fields repel the robot away from obstacles. As a result the robot is subjected to a total potential field $u^{total}$, given by:

$$u^{total} = u^{att} + \sum u^{rep}_{(i,j)} \tag{3.36}$$

Where $u^{rep}_{(i,j)}$ is a repulsion potential field that repels link $i$ from colliding with obstacle $j$. The potential $u^{rep}_{(i,j)}$ is expressed in Cartesian space as a function of the minimum distance between link $i$ of the robot and obstacle $j$. Given that the control algorithm of the robot is running in a loop, each iteration the robot shall move in a way to minimize the total potential function, as a result the robot moves towards the goal while avoiding collisions with obstacles. Traditionally, the notion of attraction and repulsion vectors is used for performing the collision avoidance. This traditional method can be interpreted as minimizing the potential function using the gradient descent technique, where the gradients represent the repulsion and attraction vectors. In contrast, we propose to use Newton method for minimizing the potential function. As shown in Appendix A, taking the second order approximation of the potential function yields the update equation for the joint angles:

$$\boldsymbol{q}^{m+1} = \boldsymbol{q}^{m} - \alpha(\mathbf{H} + \lambda\mathbf{I})^{-1}\boldsymbol{g} \tag{3.37}$$

Where $\boldsymbol{q}^m, \boldsymbol{q}^{m+1}$ are the resulting vectors of the angular positions calculated at time-steps $m, m+1$. In real world scenario $\boldsymbol{q}^m$ represents the angular positions of the joint angles as acquired from the encoder measurements, $\alpha$ is a scalar representing the minimization step, $\mathbf{H}$ is the Hessian of the potential field with respect to joint angles, $\boldsymbol{g}$ is the gradient of the potential field with respect to joint angles, $\mathbf{I}$ is the identity matrix, and $\lambda$ is a damping factor. By adopting a similar approach to the already developed techniques in differential inverse kinematics, the damping factor can be calculated as a function of the minimum singular value [105], but unlike inverse kinematics methods, in our method the minimum singular value pertains to the singular value decomposition of the Hessian matrix $\mathbf{H}$. For fast execution of the proposed algorithm the Hessian shall be calculated analytically. By taking a second order approximation of the potential function, equation (3.36), it is proven (in Appendix A) that for robotic manipulator the Hessian is given by:

$$\mathbf{H} = \sum \mathbf{J}_k^{\mathrm{T}} \nabla^2 u_k \mathbf{J}_k \tag{3.38}$$

Where $\nabla^2 u_k$ is the Hessian matrix of $u_k$ with respect to Cartesian coordinates. Given that $u_k$ is an explicit function of Cartesian coordinates for both the attraction and repulsion fields, then it is possible to derive a closed form solution for the matrix $\nabla^2 u_k$, either by hand or using symbolic math software. $\mathbf{J}_k$ is the partial Jacobian associated with the potential field $u_k$. In case of the attraction potential $u^{att}$, then $\mathbf{J}_{att}$ is the Jacobian associated with the EEF. In case of a repulsion potential $u^{rep}_{(i,j)}$ due to the effect of obstacle $j$ on link $i$, then $\mathbf{J}_{(i,j)}$ is the partial Jacobian associated with control point $cp_{(i,j)}$, where $cp_{(i,j)}$ is the point of link $i$ closest to obstacle $j$. The gradient (as shown in Appendix A) is given by:

$$\boldsymbol{g} = \sum \mathbf{J}_k^{\mathrm{T}} \nabla u_k \tag{3.39}$$

Where $\nabla u_k$ is the gradient of potential function $u_k$ in Cartesian space. For achieving better numerical robustness and by analogy to differential kinematics approaches, the alternate Jacobian described in [106] can be utilized for calculating the partial Jacobians. As a result, it can be concluded that the presented mathematical formulation forms a systematic way to integrate several potential fields:

- Attraction potential field attracting the EEF to the goal position;
- Repulsion potential field repelling the robot away from obstacles;
- Repulsion potential field repelling the robot away from self-collision and joint limits.

### 3.4.3 Experiments

To evaluate the performance of the proposed method, real-time simulations using MATLAB are implemented, where the attraction potential field $u^{att}$ is chosen to be a quadratic function with a minima at the goal position. In such a case this potential field drives the EEF towards the goal:

$$u^{att} = \frac{1}{2} k_{att} (\boldsymbol{x}_{ef} - \boldsymbol{x}_g)^{\mathrm{T}} (\boldsymbol{x}_{ef} - \boldsymbol{x}_g) \tag{3.40}$$

Where $k_{att}$ is the attraction constant, $\boldsymbol{x}_{ef}$ is the position of the EEF and $\boldsymbol{x}_g$ is the position of the goal point. On the other hand, the repulsion potential field chosen for the simulations is identical to the one in [38], where according to the proximity between obstacle $j$ and link $i$ the repulsion potential $u^{rep}_{(i,j)}$ is given by:

$$u^{rep}_{(i,j)} = \begin{cases} 0 & \rho_{(i,j)} > \rho_1 \\ \frac{k_{rep}}{\rho_{(i,j)} - \rho_0} - \frac{k_{rep}}{\rho_1 - \rho_0} & \rho_{(i,j)} \leq \rho_1 \end{cases} \tag{3.41}$$

Where $k_{rep}$ is a repulsion constant, $\rho_{(i,j)}$ is the minimum distance between obstacle $j$ and link $i$ of the robot, $\rho_1$ is the distance at which the repulsion potential starts acting on the robot, $\rho_0$ defines the forbidden area around the obstacle where the robot can not evolve. Similar repulsion fields could be added for joints limits avoidance and for self collision avoidance.

Under the previous potential functions a planar hyper redundant manipulator, with 15 DoF, is navigating through 10 obstacles towards the goal. The goal is marked by a green circle and obstacles are marked by red circles, as shown in Figure 3.23, where the Newton method is applied to perform the collision avoidance and the motion generation. The figure shows a sequence of configurations taken by the robot during its motion towards the goal. In the beginning of the simulation, the robot is in an initial configuration where it is vertically straight, links are fully stretched. Afterwards the robot starts bending and reaching towards the goal while navigating through the obstacles. In the simulation, the Hessian matrix was calculated efficiently using equation (3.38), as such an update time lower than 5 milliseconds was achieved on dual core laptop without code optimization.

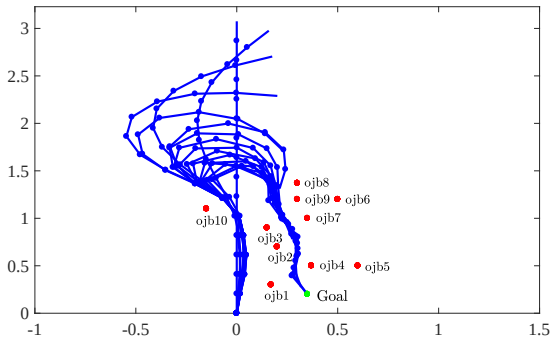


Figure 3.23: Sequence in time of collision avoidance simulation for hyper redundant planar manipulator using Newton method.

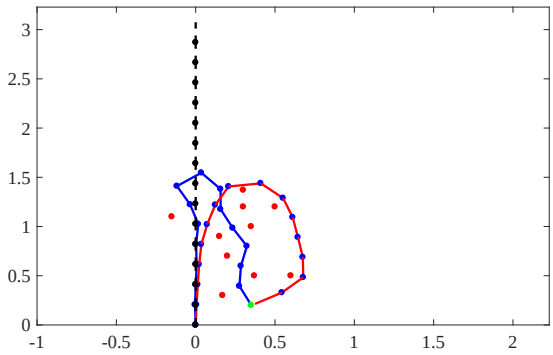

Figure 3.24: Test 1: comparison between configurations of the two manipulators after a period time from the beginning of the simulation.

For comparing the proposed implementation of the Newton method as opposed to the traditional gradient descent based method, two different tests are performed:

- Test 1: In this test a comparison between Newton method and gradient descent is performed. The comparison criteria is in terms of the generated path quality, and the minimum reach distance between last link of the manipulator and obstacles position, a video segment showing the simulation of Test 1 is in [107];
- Test 2: In this test a comparison between three different methods for performing the optimization is performed. Those methods are (1) Newton method, (2) fixed step gradient descent and (3) gradient descent with momentum. The comparison criteria is in terms of convergence rate, a video segment showing the simulation of Test 2 is in [108].

#### 3.4.3.1 Test 1

In this test the Newton method applied to the problem of collision avoidance and path planning for high DoF manipulators is compared with the gradient descent method. For this purpose the following MATLAB simulation is proposed. Two identical planar robots with 15 DoF are navigating towards the same goal point, while avoiding

collision with 10 circle-shaped obstacles. In the beginning of the simulation, the two manipulators are coincident with each other, dashed line in Figure 3.24. The blue manipulator is controlled using the Newton method and the red manipulator is controlled using the gradient descent method. For performing the comparison the same goal, obstacles, potential fields are used for both manipulators. The only difference is in the optimization method used, where for the red robot the gradient method was used to calculate the optimization direction, while for the blue robot the Newton method was used to calculate the optimization direction. For both robots the magnitude of the optimization step $\alpha$ is identical, in other words the update equation for both robots is:

$$\boldsymbol{q}^{m+1} = \boldsymbol{q}^{m} - \alpha \boldsymbol{s} \tag{3.42}$$

Where $\boldsymbol{s}$ is the optimization direction. In the case of the Newton method $\boldsymbol{s}$ is given by:

$$\boldsymbol{s} = \frac{(\mathbf{H} + \lambda \mathbf{I})^{-1} \boldsymbol{g}}{\|(\mathbf{H} + \lambda \mathbf{I})^{-1} \boldsymbol{g}\|} \tag{3.43}$$

In case of the gradient method:

$$\boldsymbol{s} = \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \tag{3.44}$$

Figure 3.24 shows the final configurations taken by the two manipulators. Figure 3.25 shows joints angles for this simulation as recorded in the Newton method (left) and the gradient descent method (right). After performing the simulation the two methods are compared in terms of:

- Quality of the path: The Newton method generated an oscillation free path. On the other hand, the gradient method suffered intermittent oscillations. This is evident in Figure 3.26, where the angular position of the first joint is plotted during the Newton method (left) and during the gradient method (right). It is worth mentioning that the oscillations, or the zig-zag phenomena as known in the optimization community, happens in the gradient method when one component of the gradient keeps flipping direction during the minimization, as such the minimization point keeps overshooting in the vicinity of a cleavage on the optimization surface [99], this phenomena does not appear in the Newton method.

- Distance between robot and obstacles: In Newton method the robot keeps bigger minimum distance away from obstacles than in the gradient method where the robot reaches closer to obstacles, despite the fact that identical potential fields, geometries, and optimization steps are used. This is demonstrated in Figure 3.27, where the distances $\rho_{(15,j)} - \rho_0$ between the last link of the robot (link 15) and the obstacles, $j = 1, 2..10$, are plotted for Newton method (left), and for the gradient method (right). Numerical values of the minimum distance reached during the simulations are listed in Table 3.2.

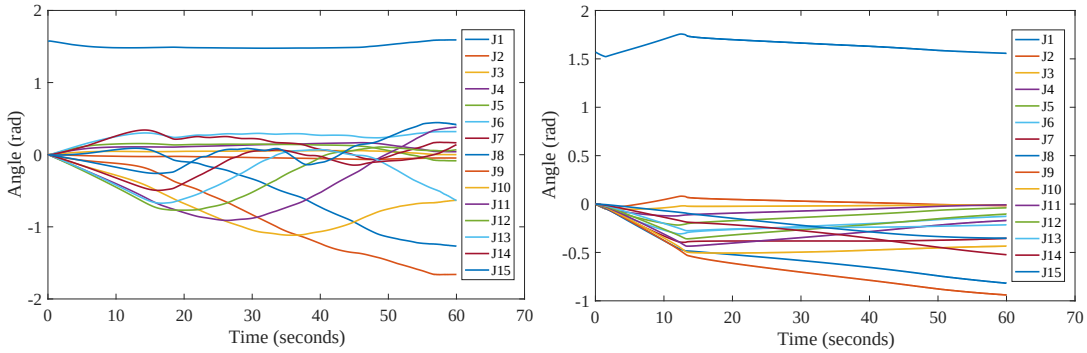


Figure 3.25: Test 1: joint angles, Newton method (left), gradient method (right).

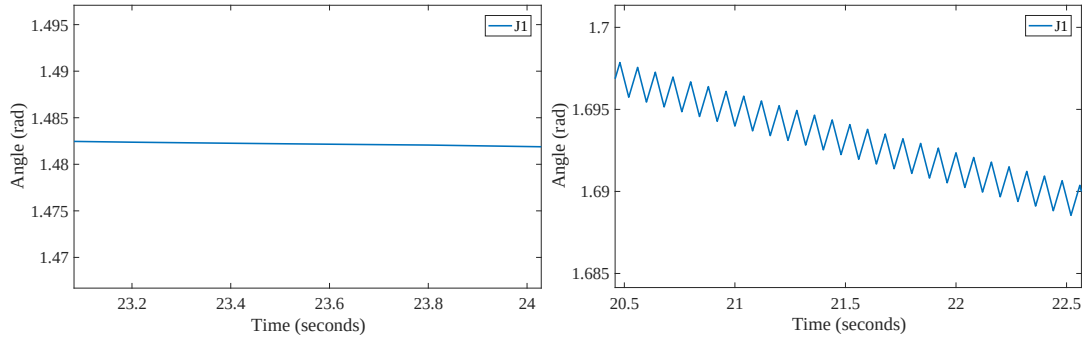


Figure 3.26: Test 1: first joint angle, Newton method (left), gradient method (right).

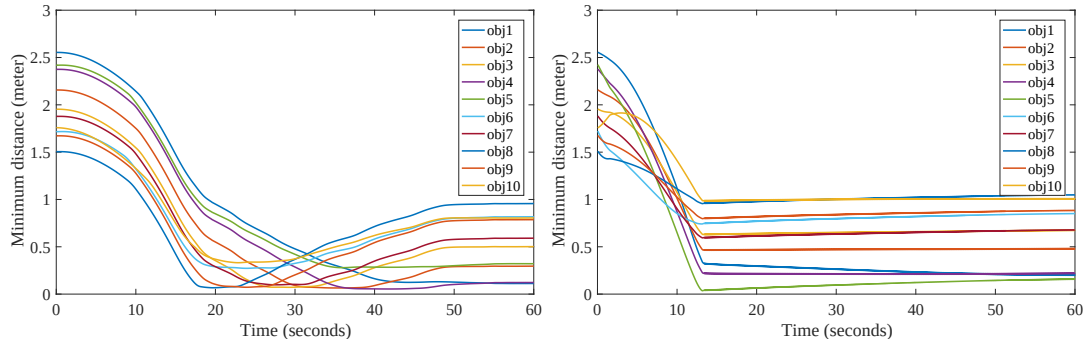


Figure 3.27: Test 1: minimum distance between last link and obstacles, Newton method (left), gradient method (right).

Table 3.2: Test 1: Minimum reach-distance between last link of the manipulator and obstacles.

| Method | Minimum distance (m) |
|---|---|
| Gradient descent | 0.0102 |
| Newton | 0.0480 |

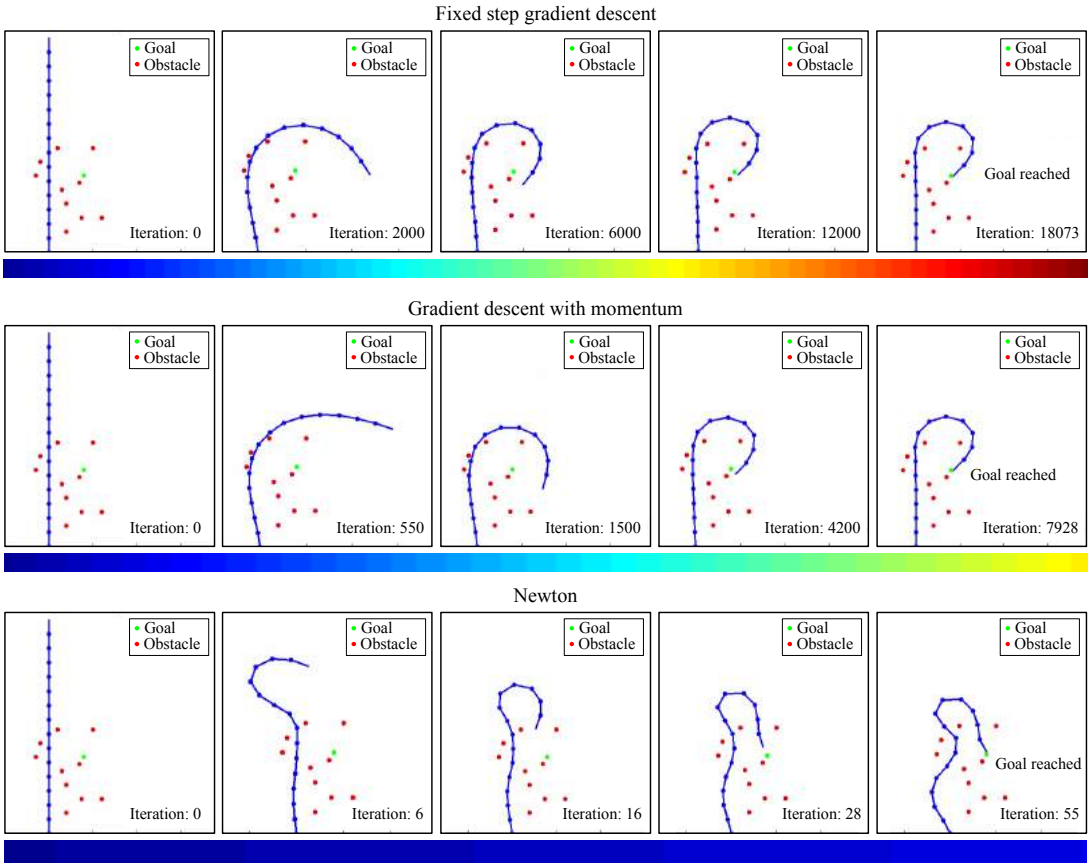


Figure 3.28: Test 2: time line during simulations of 15 DoF planar manipulator for three different optimization methods.

#### 3.4.3.2 Test 2

Test 2 is performed to asses the convergence rate for Newton method against the fixed step gradient descent and the gradient descent with momentum [109] as applied to the problem of collision avoidance and path planning. The three methods are applied in simulation environment, in all cases the simulation scenes are identical. The distance between the EEF and the goal position is used as the convergence criteria, when the EEF is closer than 0.02 meter from the goal position, the algorithm is terminated and the iterations number is reported. In all three methods the step size $\alpha$ is taken as big as possible, given that the algorithm does not cause the robot to overshoot through any of the obstacles. From the tests the Newton method achieved the best convergence per iteration, followed by the momentum and lastly the fixed step gradient method. The convergence rates achieved are as shown in Table 3.3. The differences in the resulting configuration are also evident as demonstrated in Figure 3.28.

Table 3.3: Test 2: Convergence rate for three different optimization methods as applied to collision avoidance and path planning problems for robotic manipulators.

| Method | Iterations |
|---|---|
| Newton | 55 |
| Fixed step gradient descent | 18073 |
| gradient descent with momentum | 7928 |

### 3.4.4 Summary

This section addressed the application of Newton method to collision avoidance and path planning problems for high DoF manipulators. The standard artificial potential fields is used. The collision avoidance problem is viewed mathematically as an optimization problem. The Newton method is utilized for performing the optimization. As such, filling a gap in the robotics literature, where the Newton method has been described for application on collision avoidance of mobile robots only owing to the simplicity in calculating its Hessian matrix. For robotic manipulators the Hessian matrix is more complex and computationally expensive, consequently, an efficient formula for speeding up its calculation is presented. To compare the performance of the Newton method against gradient based methods, two tests using simulation were performed. In test 1 the Newton method and the gradient method are compared in terms of quality of the path generated, oscillations, and the minimum distance to obstacles. In test 2 three different methods (1) Newton (2) fixed step gradient descent (3) gradient descent with momentum are compared in terms of the convergence rate. Qualitatively, the Newton method generates oscillation free paths and resulted in an interesting, snake like, behavior of the robot. Quantitatively the Newton method is superior in terms of convergence rate, 55 iteration for the Newton method, 18073 iterations for the fixed step gradient descent and 7928 iterations for the gradient descent with momentum method. Future work will be dedicated to applying the proposed technique for controlling industrial robotic manipulator in human centered environments to achieve real-time collision avoidance with dynamic obstacles.

## 3.5 KUKA Sunrise Toolbox

During the author's work on the collision avoidance system, a software-library (Toolbox) for controlling KUKA iiwa collaborative robots from an external computer using MATLAB is created, this was motivated by various reasons (discussed in the following subsection).

For the purpose of readability, this section is organised into various subsections. After the Motivation, the Introduction subsection gives a survey on the most popular robotic software packages available in the research community. The Architecture subsection gives details on the building blocks of the proposed Toolbox. Finally, the Modus Operandi subsection lists with examples the various methods offered by the Toolbox and their use.

### 3.5.1 Motivation

Given that one of the objectives of this thesis is to implement the collision avoidance algorithm on a real collaborative robot (KUKA iiwa), extensive time and effort was dedicated for studying the Sunrise.OS[3]. Consequently, various operation modes were tested. During these tests, the author faced various challanges, including:

1. Sensors integration: while the sensors used in the collision avoidance system are easily integrated into personal computers, they are not easily integrated into the robot's controller.

2. Simulation integration: to write programs for KUKA iiwa robots, the Sunrise.Workbench IDE (Integrated Development Environment) is used. Unfortunately, the Sunrise.Workbench does not include 3D simulation capability, which is very important during the development process of the control algorithm (as explained later).

3. Implementing advanced mathematical operations: the Sunrise.Workbench supports Java programming language, which is a powerful language. But the collision avoidance algorithm includes lots of advanced mathematical operations (matrix operations, filtering, integration among others). Consequently, developing the algorithm in Java will require considerable amount of time.

4. Hardware limitations: due to the limitation of the performance of the robot controller, the complexity of the developed algorithms and their performance is limited by the robot's hardware capacity.

Faced with the previous limitations, the author opted to develop the collision avoidance control algorithm on an external computer using MATLAB, which offers the following advantages:

1. MATLAB is suitable for developing the collision avoidance algorithm (mathematically demanding), given its support of advanced mathematics and the existence of powerful toolboxes including the control toolbox and the signal processing toolbox.

[3]The operating system of the KUKA iiwa.

2. MATLAB supports various debugging and data visualisation tools, which facilitate the process of debugging advanced control algorithms including the collision avoidance algorithm.

3. MATLAB allows interfacing with Vrep (3D simulation environment). This allows simulating and experimenting the collision avoidance algorithm in a virtual environment before implementation on the real robot. This is extremely important for verifying the control algorithm, and offers two main advantages, (1) reducing testing and validation time, (2) reducing the risk of equipment loss and injuries.

4. The used sensors are easily integrated into external computers. Hardware manufacturers offer drivers and Application Programming Interfaces (APIs) for using their sensors on personal computers.

5. MATLAB compatible APIs already exist for interfacing with the used senors.

6. Unlike the robot hardware, upgrading an external computer with better hardware is easily done. Thus allowing the implementation of more complex algorithms without compromising the real-time performance.

7. Using an external computer allows the experimentation with various types of sensors easily.

Given the powerful advantages of using MATLAB, it was decided to bundle the developed interfaces, mostly during the development of the collision avoidance system, in a MATLAB toolbox that allows users to control the robot more easily. The developed toolbox was named the KUKA Sunrise Toolbox (KST). It was published in the IEEE Robotics & Automation Magazine in [11] including proper documentation and tutorials.

### 3.5.2 Introduction

Robotics is a multidisciplinary subject. It involves various engineering specialities, including: electrical engineering, mechanical engineering, control engineering, computer science and software engineering. Consequently, to develop a robust robotic application, all previous disciplines work hand in hand. To avoid reinventing the wheel and to facilitate the way robots are programmed, various researches have developed robotics libraries and made them available for use. The robotics toolbox for MATLAB [110, 111] is a MATLAB library that implements various robotics algorithms important for calculating the forward/inverse kinematics, Jacobian, robot dynamics and others. The Machine Vision Toolbox [112] is another MATLAB toolbox that is used to integrate vision into robotics applications. Those packages are used for robot simulation and for developing control algorithms, yet they do not interface directly with the robot. Due to the advantages of using an external computer for controlling industrial robots, various software packages have been proposed. Those packages differ in the type of robots they support, the programming languages they implement, in addition to the different functionalities they offer. Using the KUKA Control Toolbox (KCT) [113] the user is able to control KUKA industrial robots equipped with the KUKA Robot Controller (KRC)

from an external computer using MATLAB. JOpenShowVar [114, 115] is another library that allows interfacing KRC controllers from external PC using Java. For KUKA iiwa collaborative robots (equipped with the Sunrise Controller) various packages also exist. From the Robot Operating System (ROS) [116] the user is able to interface with KUKA iiwa using the iiwa_stack [117] and KUKA-IIWA-API [118]. The iiwa_stack is a library that allows controlling iiwa robots from ROS. For communication purposes it implements the ROSJava nodes (third party libraries). In addition, its motion functions are built on top of the Smart.Servo library (from KUKA) and does not implement the robot's native point-to-point motion functions. The KUKA-IIWA-API provides a ROS interface for controlling KUKA iiwa from an external PC, similar to iiwa_stack the interface provides various topics, the user can subscribe or publish to these topics, but in the case of KUKA-IIWA-API the update rate for data feedback is low (10 Hz). To control the robot using iiwa_stack or KUKA-IIWA-API a familiarity with ROS is required. On the other hand, the KUKA Sunrise Toolbox (KST) allows controlling KUKA iiwa robots from an external computer using MATLAB, such that any person with basic knowledge on MATLAB can start using KUKA collaborative robots without requiring advanced knowledge on how to program industrial manipulators. With more than a hundred different functions, KST supports various operation modes including soft-real time control, point-to-point motion calls and non-blocking motion calls, those functions are divided into seven categories detailed in subsection 3.5.6.

The toolbox was designed to be user friendly, so that anyone with basic knowledge about MATLAB can start using KUKA iiwa collaborative manipulators in a minimal amount of time. KST is provided under MIT license, and is freely available in the GitHub repository [26]. The repository also includes a detailed documentation, video tutorials, code snippets and tutorial MATLAB scripts.

### 3.5.3 Architecture

The KST has a client server architecture, Figure 3.29. For achieving a robust data transmission the toolbox utilises the Transmission Control Protocol/Internet Protocol (TCP/IP). The server, represented by KUKA Sunrise Cabinet in Figure 3.29, is a multi-threaded Java application that runs inside the robot controller. The main thread of the Java application, KST Main in Figure 3.29, is mainly used to run the motion commands. The second thread, KST Server in Figure 3.29, is mainly used for communication layer and also data acquisition using robot's own sensors including: the integrated joints torque sensors, the encoders and the inputs integrated in the flange. Before starting to utilize the toolbox the user shall synchronize the server application into the robot controller, after being synchronized the user shall run the server, once is running, the toolbox can be used to connect to the robot from MATLAB.

On the other hand, the client part is written using MATLAB scripting language, the various functions offered by the toolbox are wrapped inside the KST.m class. Therefore the user is able to control the robot from MATLAB through this class.

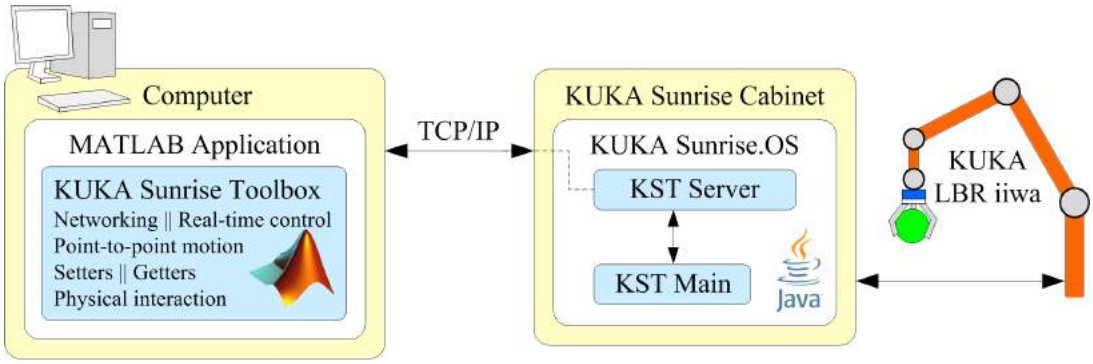


Figure 3.29: Architecture of the toolbox.

### 3.5.4 Modus Operandi

The KUKA Sunrise Toolbox implements object oriented programming. The main programming class of the toolbox is KST, a wrapper class which includes the various properties and functions (methods) required to control the manipulator from MATLAB. To get an access to the controller of the robot, the user shall first instantiate an object from the KST class by calling its constructor:

```
>> iiwa = KST(ip,rT,fT,Tef_fl);
```

Where `iiwa` is an instance of the KST class, `ip` is a string variable which contains the IP of the robot, `rT` is a constant referring to robot type 7R800 or 14R820, `fT` is a constant referring to type of the flange attached to the robot, and `Tef_fl` is 4x4 matrix representing the transformation matrix from the end-effector to the frame of the flange (`Tef_fl` is an optional argument if omitted the KST uses the identity matrix by default).

In the following example an instance of the KST class associated with LBR7R800 robot is created.

```
>> ip='172.31.1.147';
>> rT=KST.LBR7R800;
>> fT=KST.Medien_Flansch_elektrisch;
>> Tef_fl=eye(4);
>> iiwa = KST(ip,rT,fT,Tef_fl);
```

### 3.5.5 Properties of the KST class

The KST class implements several properties. Those properties are used to store various variables and values that are important for the proper operation of the toolbox. Some of those properties are used for communication purposes, for example to store the IP of the robot or to store the TCP/IP communication object used by the class. Others are used to store the various parameters defining the robot, like the type of the robot, the type of the flange attached to the robot, the Denavit-Hartenberg (DH)

parameters of the robot and the inertial data of the robot. These inertial parameters include: the mass of each link, the inertial tensor of each link and the position vector of center of mass of each link. As an example, to access the link-twist angles [32] of DH parameters, the user can access the `dh_data` property as the following:

```
>> alfa=iiwa.dh_data.alfa

alfa=
{[0]} {[-1.57]} {[1.57]} {[1.57]} {[-1.57]} {[-1.57]} {[1.57]}
```

### 3.5.6 Methods of the KST class

The KST class implements various methods (Appendix B), according to their functionality those methods are grouped into seven different categories:

1. Networking;
2. Soft real-time control;
3. Point-to-point (PTP) motion;
4. Setters;
5. Getters;
6. General purpose;
7. Physical interaction.

In the following subsections each category is presented along with illustrative snippet code examples.

#### Networking

The KST provides methods to establish and terminate the connection with the robot controller (Table 6.1 of Appendix B). The TCP/IP connection is initialized by typing:

```
>> iiwa.net_establishConnection();
```

The TCP/IP connection is terminated by typing:

```
>> iiwa.net_turnOffServer();
```

#### Soft real-time control

The soft real-time motion control is required when performing online motion generation, where the trajectory is generated/adjusted online, for example from external sensor data (see Test 3 in subsection 3.3.6.3). KST allows soft real-time control of the robot. The control modes supported are joints velocity mode and joints/end-effector

position control mode. The soft real-time motion in joints position mode is activated by typing the command:

```
>> iiwa.realTime_startDirectServoJoints()
```

Once started, the target angular positions of the joints can be sent to the robot using:

```
>> jPos={pi/3,0,0,-pi/2,0,pi/6,pi/2};
>> iiwa.sendJointsPositions(jPos);
```

Where `jPos` is a cell array of dimension 1x7 containing the values of the target joints angles in radians (KUKA LBR iiwa is a 7 DoF manipulator). A loop can be implemented for sending a stream of joint angles to the robot, so that the robot will perform the motion between the points on-the-fly.

To stop the soft real-time motion control the following method is used:

```
>> iiwa.realTime_stopDirectServoJoints();
```

The impedance control is activated by:

```
>> iiwa.realTime_startImpedanceJoints(
        massOfTool,cOMx,cOMy,cOMz,
        cStifness,rStifness,nStifness);
```

Where `massOfTool` is the mass of the tool (in kg), `cOMx`, `cOMy`, `cOMz` are the coordinates of the centre of mass of the tool in the reference frame of the flange (in mm), and `cStifness`, `rStifness`, `nStifness` are the stiffness values for Cartesian translation, Cartesian orientation and null space, respectively. The robot joint positions are updated online using:

```
>> iiwa.sendJointsPositions(jPos);
```

To stop the impedance functionality:

```
>> iiwa.realTime_stopImpedanceJoints();
```

The KST simultaneous control-feedback methods allow to control the robot while simultaneously receiving feedback about the robot's internal state, namely torques, joint positions and end-effector position. These functions are possible due to the multithreaded nature of the KST server.

### Point-to-point motion

The KST point-to-point (PTP) motion methods allow the robot to move from the current configuration/end-effector pose (position and orientation) to a defined target configuration/pose. The user is not required to implement the path planning algorithms

since they are implemented directly by the software. The robot can be controlled in Cartesian space and also in joint space. For example, to move the robot in joint space:

```
>> jPos={pi/3,0,0,-pi/2,0,pi/6,pi/2};
>> relVel=0.25;
>> iiwa.movePTPJointSpace(jPos,relVel);
```

Where `relVel` represents the override velocity assuming a value from zero to one. The robot can also be controlled in Cartesian space. For example, to move the end-effector along a straight line:

```
>> pos={400,0,580,-pi,0,-pi};
>> vel=50;
>> iiwa.movePTPLineEEF(pos,vel);
```

Where `pos` is the target end-effector pose in Cartesian space, a 1x6 cell array in which the first three cells are the end-effector x,y and z coordinates (relative to robot base) and the last three cells are the X-Y-Z fixed rotation angles representing the orientation of the end-effector in the space. The variable `vel` represents the linear velocity of the end-effector in mm/sec. The KST also includes methods for arc and ellipse motion.

The aforementioned code snippet utilizes a blocking call. However, the KST also supports non-blocking PTP motion methods (listed in Table 6.4 of Appendix B). This gives the possibility to perform other computations, like path planning or input sampling, while the robot is moving. Examples of these methods are the `nonBlocking_isGoalReached` and the `nonBlocking_movePTPArcXY_AC`.

In some applications the ability to interrupt robot motion upon the occurrence of some condition is required, for instance if a given torque threshold is reached. The KST allows such operations using the conditional motion functions. An example on using those functions is in the tutorial script `KSTclass_Tutorial_ptpConditionalTorques.m`.

### Setters

KST provides methods for updating robot parameters, including the state of the outputs and the LED light of the flange. For example, the LED light is useful to alert the human about the state of the robot and can be associated to the beginning and end of a given robot task. To turn on the blue LED the following method is used:

```
>> iiwa.setBlueOn();
```

To turn off the blue led:

```
>> iiwa.setBlueOff();
```

### Getters

The KST provides functionalities to acquire various internal parameters of the robot such as joint angles, end-effector pose, force/moment acting on the end-effector, joints torques and the values of the IO connector inputs. For example, to acquire the joint angles of the robot, the following method is utilized:

```
>> jPos = iiwa.getJointsPos()

jPos =
{[1.017]} {[1.919]} {[0.513]} {[0.621]} {[0.698]} {[0.341]} {[0.175]}
```

The variable `jPos` is a 1x7 cell array with the robot joint angles in radians.

### General purpose

The KST integrates several methods to calculate forward kinematics, inverse kinematics, partial Jacobian, forward dynamics, inverse dynamics, among others. As an example, to calculate the transform matrix of the end-effector and the Jacobian associated with the end-effector:

```
>> [eef_transform,J] = iiwa.gen_DirectKinematics(q)

eef_Transform =
-0.8162 -0.2795  0.5056  0.6229
 0.5210  0.0220  0.8532  0.8155
-0.2496  0.9599  0.1277  0.3434
 0.0000  0.0000  0.0000  1.0000

J =
-0.815  0.001  0.281  0.148 -0.039 -0.211  0.00
 0.622  0.002 -0.214 -0.297  0.010  0.142  0.00
 0.000 -1.017 -0.123  0.578  0.086 -0.115  0.00
 0.000 -0.866  0.469  0.664  0.733 -0.417  0.50
 0.000  0.500  0.813 -0.581  0.627  0.112  0.85
 1.000  0.000 -0.342 -0.469  0.261  0.902  0.12
```

Where `eef_transform` is the end-effector transform matrix, `J` is the Jacobian associated with the end-effecor, and the variable `q` is a 7x1 vector of the robot joint angles in radians.

To calculate the partial Jacobian associated to a given point in a given robot link:

```
>> J = iiwa.gen_partialJacobean(q,ln,Pos);
```

Where `ln` is the number of the link for which the partial Jacobian is calculated (an integer number from 1 to 7), `Pos` is a 3x1 vector representing the position of the point at which the partial Jacobian is calculated (this vector is described in the reference frame of the link specified by `ln`), and `J` is the partial Jacobian at the described point.

#### Physical interaction

KST provides several physical interaction functionalities, some of which are the hand-guiding mode and the touch detection. The hand-guiding is activated using:

```
>> iiwa.startHandGuiding();
```

Once called the hand-guiding functionality is initiated. To perform hand-guiding operation, the user shall press the flange's white button to deactivate the brakes and move the robot. When the white button is released the robot stops in its current configuration. To terminate the hand-guiding mode, the green button should be pressed continuously for more than 1.5 seconds (after 1.5 seconds of pressing the green button the blue LED light starts to flicker), release the green button and the hand-guiding mode is terminated.

### 3.5.7 Summary

According to users' feedback (students, researchers, and industry engineers), the proposed toolbox is a useful and intuitive tool to interface with KUKA Sunrise.OS and, in particular, to speed up the development and implementation of robot applications. KST functionalities are advantageous for the implementation of advanced robot applications. KST also facilitates integration of external hardware, data processing, and implementation of complex algorithms using existing toolboxes.

# Chapter 4

# Hand-guiding

The hand-guiding functionality represents an intuitive human-robot interface which allows human coworkers to move and teach the robot easily, without requiring advanced technical skills on how to program industrial manipulators. This functionality also allows to use the robot as an assistive third hand, which is appealing for tedious assembly tasks. Consequently, a robot can be hand-guided to lift and hold parts in place, while giving the human coworker an opportunity to apply the fixtures (perform the assembly task). Such functionality reduces the risk to workers and properties (falling components), provides precision, allows lifting heavier parts, and increases productivity by keeping less workers occupied in manual tasks on the factory floor.

## Chapter's breakdown

This chapter of the document is organised in two main sections:

- Section 4.1 describes the proposed algorithm for performing precise hand-guiding at the EEF level. This method is evaluated on 7 DoF KUKA iiwa robot.
- Section 4.2 builds upon the ideas presented in section 4.1, where a method is developed that allows hand-guiding sensitive redundant manipulators (at EEF level) in cluttered environments with obstacles. In such a case, the redundancy of the robot is used to navigate obstacles during the contact while preserving the precision of the motion at the EEF in Cartesian space.

## 4.1 Precision Hand-Guiding

In the hand-guiding mode the robot moves compliantly according to the force applied by the user. While it is a good tool for rough positioning, end-effector (EEF) precision positioning is still difficult to achieve by hand-guiding. This section presents a novel method for precisely hand-guiding a robot at the EEF level. Using the EEF force/torque measurements the hand-guiding force/moment is calculated after compensating for the weight/inertia of the EEF. The proposed control solution is inspired by the motion properties of a passive mechanical system subjected to Coulomb/viscous friction. As a result, the human can control the linear/angular motion of the decoupled EEF. To ensure that the robot does not exceed its joints limits during the hand-guiding process, a solution to scale down the linear/angular velocity of the EEF near the joints limits is incorporated into the control strategy. Several experimental tests were conducted in an assembly task using a KUKA iiwa manipulator. Performance was evaluated considering the precision in positioning, ability to avoid joint limits and vibrations during hand-guiding. A qualitative and quantitative comparison of the proposed control strategy against the off-the-shelf KUKA iiwa hand-guiding is also presented.

### 4.1.1 Introduction

Physical human-robot interaction has been studied extensively in the last decade. Research in this field focused on the ability to interact with a robot physically while preserving safety. This includes hand-guiding, which is a representative functionality of collaborative robots that allows the unskilled user to interact and to program robots in a more intuitive way than using the traditional teach pendant. Existing collaborative robots include hand-guiding functionality with limitations in terms of the precision and accuracy required for many tasks. When precision positioning (position and orientation of the EEF) is required, the teach pendant is still widely used, where it remains the main option for most of robots in operation [119]. However, the use of the teach pendant limits the intuitiveness of the teaching process, also it is time consuming (an important parameter on the factory floor) [120]. In addition, the use of the teach pendant requires a mental visualisation of the different reference axes of the robot, which lacks intuitiveness and becomes cumbersome as the human moves around the robot [121]. In some scenarios, using the teach pendant for robot positioning can lead to undesirable collisions, which may cause damage to sensitive equipment.

Owing to its importance, hand-guiding has been addressed in several studies, where different terminologies have been used by researchers for describing the hand-guiding process: manual-guidance [122], force guidance [123], lead-through programming [124], or walk-through programming [125]. Due to its advantages, the hand-guiding functionality is gaining popularity in the robotics community, allowing to program a robot in an intuitive manner by moving it compliantly with the applied force.

In this section it is presented a novel method for hand-guiding while preserving the precision at EEF level, an important requirement for plenty of robotized manufacturing operations. By analogy to the motion properties of a passive mechanical system, mass subjected to Coulomb/viscous friction, a control scheme is proposed, which governs

the linear/angular motion of the decoupled EEF. To avoid violating the physical limits of the joints, a scaling factor is introduced, it is used to scale down the linear/angular velocity of the EEF near the joints limits. To test the proposed approach, experiments were conducted in robot assisted assembly tasks requiring precision, where a KUKA iiwa robot is used. Performance was evaluated considering the precision in positioning, ability to avoid joints limits and the smoothness of the motion during hand-guiding. In addition, a qualitative and quantitative comparison of the proposed control strategy against the off-the-shelf KUKA iiwa hand-guiding is presented. Results indicate that the proposed solution allows smooth Cartesian hand-guiding motion, with low level of vibrations, while having the ability to react for avoiding the joints limits.

### 4.1.2 State of the art

The importance of hand-guiding as a functionality to teach collaborative robots is well known in the robotics community. At the same time, it is still a challenge to have robots and humans sharing the same space and performing collaborative tasks with the required safety levels to minimise the risk of injuries [77]. Humans can interact physically with a robotic arm by hand-guiding the robot into the desired grasping poses while the robot's configuration is recorded [126]. Robot assistance through hand-guiding is also used to assist humans in the transport of heavy parts [75]. In [127] the integration of a human-robot cooperative system (implementing hand-guiding) in a production line was studied including safety, operability and the level of assistance to humans. The assisted gravity compensation method is presented in [128]. This method facilitates the hand-guiding process, making it more intuitive for unskilled users. A virtual tool method for kinesthetic teaching of robotic manipulators is proposed in [129]. The method builds on an admittance controller that utilizes the feedback from a FT sensor attached at the EEF. In a recent study an operator guides a collaborative robot along a predefined geometric path while taking required joint constraints into account and thus implicitly considering the actuator dynamics [130]. Kinesthetic teaching and learning have been combined to enable non-experts to configure and program a redundant robot in the presence of constraints such as confined spaces [131].

Concerning the use of conventional industrial robots, hand-guiding is normally sensor based. However, sensorless hand-guiding methods have been proposed. In [132], the dynamic model of a robot along with the motor current and friction model is used to determine the user's intention to move the EEF of a robot instead of directly sensing the external force by the user. By gaining access to motors currents measurements, hand-guiding can be achieved without the need to install an external FT sensor on the manipulator [133]. In [124] the authors presented a sensorless method for hand-guiding the EEF in a structured surrounding.

The problem of Cartesian impedance control of a redundant robot executing a co-operative task with a human has been addressed in [134]. Redundancy was used to keep robot's natural behaviour as close as possible to the desired impedance behaviour, by decoupling the EEF equivalent inertia. The authors claim that this allows to easily find a region in the impedance parameter space where stability is preserved. An important study in the field uses impedance control of multiple robots to aid a human moving an object [135]. Experiments were conducted considering motion along 1 DoF.

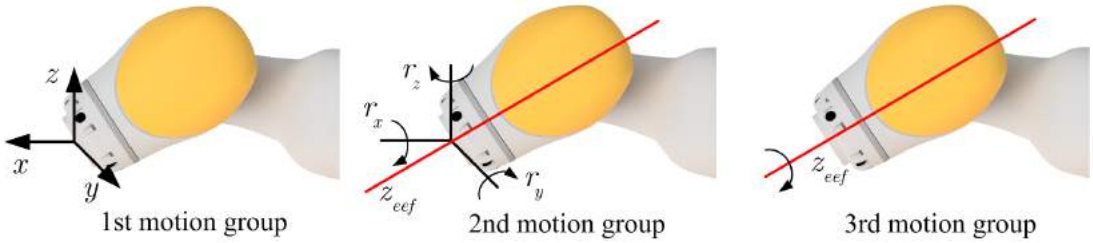


Figure 4.1: Robot motion groups for precision hand-guiding.

In medical domain, robot assisted surgery demands high precision and consequently hand-guiding accuracy. In this scenario, an approach that enables an operator to guide a robot along a predefined geometric path while considering joint constraints and implicitly the actuator dynamics is proposed in [130].

However, the subject of precision in hand-guiding for robot assisted assembly operations was not fully covered in the literature. In this section it is presented a solution for precisely hand-guiding a robot at the EEF level applicable for precise assembly operations.

### 4.1.3 Work principle

Assuming force feedback at the robot EEF (the robot may have joint torque sensors or a force/torque sensor attached to the flange of the robot), three groups of robot hand-guiding motion are considered, Figure 4.1:

1. The first motion-group used to control the positioning (linear displacements) of the EEF along the x, y and z axes of the robot base;
2. The second motion-group used to control the orientation of the EEF axis in Cartesian space;
3. The third motion group represents the rotation of the EEF around its axis.

Those motion groups are selected because they cover the 6-DoF translation and rotation in Cartesian space, also because they are very intuitive for workers (just imagine how humans insert a key into a keyhole). A video showing the proposed concept is in [136].

The robot moves in one of the three motion groups when a force/moment is applied at the EEF. Two terms will be introduced to specify the input force/moment. The first term is the hand-guiding force for linear positioning, defined as the force applied by the operator at the EEF. The second term is the hand-guiding moment for angular positioning, defined as the moment applied by the operator at the EEF. The hand-guiding force/moment shall be calculated from the force/moment measurements (directly acquired from FT sensor, or indirectly calculated from integrated joint torque sensors). Those measurements represent the forces/moments due to (1) the EEF weight, (2) the inertial force/moment due to the acceleration of the EEF, and (3) the external hand-guiding force/moment applied by the operator. To simplify the calculations, we omit the inertial force/moment due to the acceleration of the EEF because in precise hand-guiding they are relatively small compared to EEF's weight and to the external

force/moment. In this context, the force/moment measurements are approximated to be due to the EEF weight and the external hand-guiding force/moment applied by the operator. The components of the hand-guiding force described relative to the robot base frame are used as inputs for the proposed linear motion controller, i.e., for positioning the EEF according to the robot base frame (first motion group). The components of the hand-guiding moment are used as inputs for the orientation controller (second and third motion groups).

Throughout this section, the following notation is used to describe a vector in a reference frame:

- The superscript $b$ is used for vectors described in base frame of the robot;
- The superscript $e$ is used for vectors described in end-effector frame. It is considered that the origin of the EEF frame is coincident with the point where the hand-guiding force/moment acts. Consequently, the human operator applies the hand-guiding force/moment at the handle, which is coincident with the origin of the EEF reference frame;
- The superscript $s$ is used for vectors described in sensor measurement frame. This frame is defined by the manufacturer of the FT sensor (in case an external sensor is used). However, if a sensitive robot is used, then the sensor measurement frame is virtual, at which the equivalent values of the external forces/moments are calculated from the torque readings at the integrated torque sensors. In our experiment with KUKA iiwa robot, the sensor measurement frame is considered at the frame of the flange;
- The little hat above a vector ($\hat{}$) is used to denote the skew symmetric matrix of that vector.

### 4.1.4 Hand guiding force

The measured hand-guiding force described in robot base frame is $\boldsymbol{f}^b = (f_x, f_y, f_z)$. This force $\boldsymbol{f}^b$ serves as the input for calculating the 1st motion group command. The maximum of the components $(f_x, f_y, f_z)$ is used as the control command to move the end-effector along one axis at a time. The hand-guiding force $(f_x^e, f_y^e, f_z^e)$ described in the EEF reference frame is calculated from:

$$\begin{bmatrix} f_x^e \\ f_y^e \\ f_z^e \end{bmatrix} = \mathbf{R}_s^e \left( \begin{bmatrix} u_x^s \\ u_y^s \\ u_z^s \end{bmatrix} - \mathbf{R}_b^s \boldsymbol{w} \right) \tag{4.1}$$

Where $(u_x^s, u_y^s, u_z^s)$ are components of force measurements in sensor frame, $\mathbf{R}_s^e$ is a constant rotation matrix that describes the orientation of the sensor measurement frame in relation to the EEF frame, and the vector $\boldsymbol{w}$ is defined by $\boldsymbol{w} = [\ 0 \quad 0 \quad -w\ ]^{\mathrm{T}}$ for base mounted robots in the upright position, in which $w$ is the weight of the EEF. $\mathbf{R}_b^s$ is the rotation matrix describing the orientation of robot base frame in relation to sensor frame:

Table 4.1: Values and conditions to obtain $(a, b, c)$.

| $a$ | $b$ | $c$ | *Condition* |
|---|---|---|---|
| 1 | 0 | 0 | $\lvert f_x\rvert > \lvert f_y\rvert \,\&\&\, \lvert f_x\rvert > \lvert f_z\rvert$ |
| 0 | 1 | 0 | $\lvert f_y\rvert > \lvert f_x\rvert \,\&\&\, \lvert f_y\rvert > \lvert f_z\rvert$ |
| 0 | 0 | 1 | $\lvert f_z\rvert > \lvert f_x\rvert \,\&\&\, \lvert f_z\rvert > \lvert f_y\rvert$ |

$$\mathbf{R}_b^s = \mathbf{R}_e^s \left(\mathbf{R}_e^b\right)^{\mathrm{T}} \tag{4.2}$$

Where $\mathbf{R}_e^b$ is the rotation matrix of EEF frame in relation to robot's base frame, obtained from the forward kinematics.

The hand-guiding force $\boldsymbol{f}^b$ described in base frame is calculated from:

$$\boldsymbol{f}^b = \mathbf{R}_e^b \begin{bmatrix} f_x^e \\ f_y^e \\ f_z^e \end{bmatrix} \tag{4.3}$$

The force command $\boldsymbol{f}_{cmd}$ sent to the control algorithm is:

$$\boldsymbol{f}_{cmd} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \boldsymbol{f}^b \tag{4.4}$$

Where $(a, b, c)$ values are defined according to the force conditions detailed in Table 4.1. In addition, to achieve smooth transitioning between different directions/conditions, switching from one motion direction to another is only allowed when the velocity of the EEF is zero.

### 4.1.5 Hand guiding moment

The end-effector orientation (2nd and 3rd motion groups) is calculated from the hand-guiding moment. In this scenario, the hand-guiding moment $\boldsymbol{m}^e$ in the EEF reference frame is calculated:

$$\boldsymbol{m}^e = \mathbf{R}_s^e \left( \begin{bmatrix} \tau_x^s \\ \tau_y^s \\ \tau_z^s \end{bmatrix} - \hat{\boldsymbol{p}}_e^s \mathbf{R}_e^s \boldsymbol{f}^e - \hat{\boldsymbol{p}}_c^s \mathbf{R}_b^s \boldsymbol{w} \right) \tag{4.5}$$

Where $(\tau_x^s, \tau_y^s, \tau_z^s)$ are the three torques measurements from the force/torque sensor, $\boldsymbol{p}_e^s$ is the vector describing the position of the EEF in sensor's reference frame, $\boldsymbol{p}_c^s$ is the vector describing the position of the center-of-mass (COM) of the EEF in the sensor's reference frame.

In the 2nd motion group, the end-effector axis $\boldsymbol{z}_{eef}$ is oriented in space, Figure 4.1. For this type of motion the input to the controller, vector $\boldsymbol{m}_{xy}$, is calculated from the hand-guiding moment $\boldsymbol{m}^e$. The component of the hand-guiding moment in the xy plane of the end-effector frame, $\boldsymbol{m}_{xy}^e$, is shown in Figure 4.2 where it is calculated:

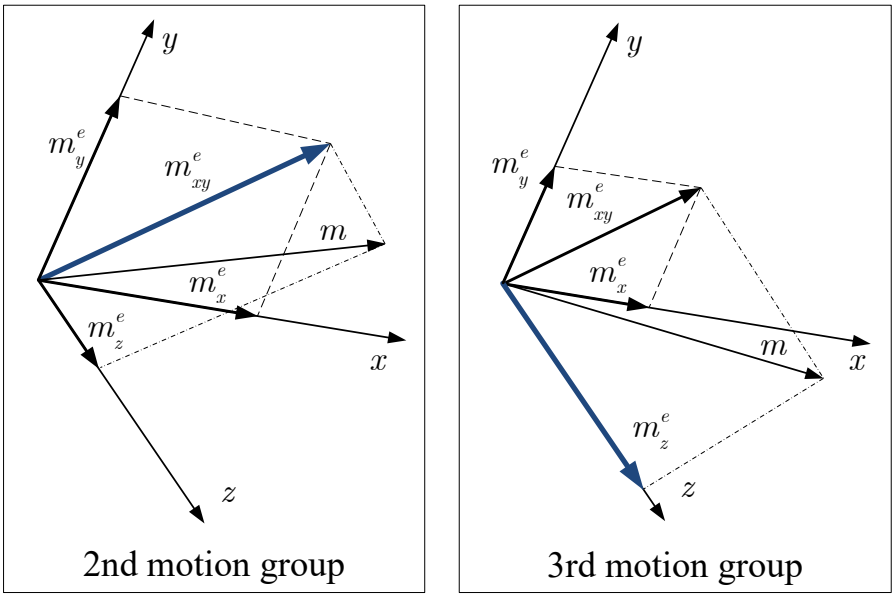


Figure 4.2: Hand-guiding moment in EEF reference frame for 2nd motion group (left) and for the 3rd motion group (right).

$$\boldsymbol{m}_{xy}^e = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{m}^e \tag{4.6}$$

The controller command for 2nd motion group is calculated by transforming $\boldsymbol{m}_{xy}^e$ to the robot base frame:

$$\boldsymbol{m}_{xy} = \mathbf{R}_e^b \boldsymbol{m}_{xy}^e \tag{4.7}$$

For the 3rd motion group, the end-effector is allowed to rotate around its axis $z_{eef}$. The input command to the controller is the vector $\boldsymbol{m}_z$. To calculate this vector, the vector $\boldsymbol{m}_z^e$, Figure 4.2, shall be calculated first:

$$\boldsymbol{m}_z^e = \boldsymbol{m}^e - \boldsymbol{m}_{xy}^e \tag{4.8}$$

The controller command for the 3rd motion group is calculated by transforming $\boldsymbol{m}_z^e$ to the robot base frame:

$$\boldsymbol{m}_z = \mathbf{R}_e^b \boldsymbol{m}_z^e \tag{4.9}$$

Only one motion group (2nd or 3rd motion group) is allowed to be performed once at a time. Depending on the magnitude of $\boldsymbol{m}_{xy}$ and $\boldsymbol{m}_z$, the moment command $\boldsymbol{m}_{cmd}$ sent to the control algorithm is:

$$\boldsymbol{m}_{cmd} = \begin{cases} \boldsymbol{m}_{xy} & if \quad \|\boldsymbol{m}_{xy}\| > \|\boldsymbol{m}_z\| \\ \boldsymbol{m}_z & if \quad \|\boldsymbol{m}_z\| \geq \|\boldsymbol{m}_{xy}\| \end{cases} \tag{4.10}$$

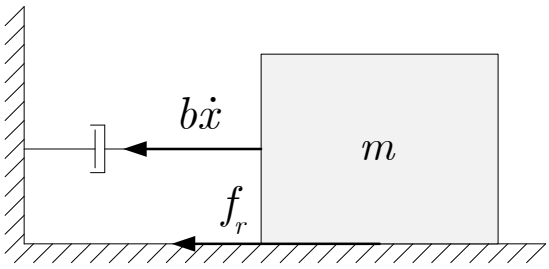


Figure 4.3: Damper-mass mechanical system.

### 4.1.6 Controller

The robot is controlled at the EEF level, considering the decoupled end-effector as a mass moving under the effect of Coulomb and viscous friction, Figure 4.3. The motion equation of center of mass (COM) is defined by:

$$m\ddot{x} + b\dot{x} + f_r = f \tag{4.11}$$

Where $\ddot{x}$ is the linear acceleration of the COM, $\dot{x}$ is the linear velocity of the COM, $m$ is the mass, $b$ is the damping coefficient, $f_r$ is Coulomb friction and $f$ is the external force acting on the mass. The equation of angular rotation around an axis passing through the COM is:

$$I\ddot{\theta} + \beta\dot{\theta} + \tau_r = \tau \tag{4.12}$$

Where $\ddot{\theta}$ is the angular acceleration around rotation axis, $\dot{\theta}$ is the angular velocity, $I$ is the moment of inertia around the rotation axis, $\beta$ is the damping coefficient, $\tau_r$ is the torque due to Coulomb friction and $\tau$ is the external torque.

We consider that the Coulomb and viscous friction effects are much bigger than the effect of the inertial forces due to acceleration/deceleration of the mass. In this context, the inertial terms in the previous equations are omitted and equation (4.11) becomes:

$$\begin{cases} b\dot{x} + f_r = f & |f| > |f_r| \\ \dot{x} = 0 & otherwise \end{cases} \tag{4.13}$$

Where $|f|$ is the absolute value of the external force. By applying the same reasoning for the angular motion equation (4.12) becomes:

$$\begin{cases} \beta\dot{\theta} + \tau_r = \tau & |\tau| > |\tau_r| \\ \dot{\theta} = 0 & otherwise \end{cases} \tag{4.14}$$

Where $|\tau|$ is the absolute value of the external torque.

### 4.1.7 Robot control

The linear motion of the end-effector is controlled by:

$$\begin{cases} \boldsymbol{v} = \frac{\boldsymbol{f}_{cmd}(\|\boldsymbol{f}_{cmd}\| - |f_r|)}{b' \|\boldsymbol{f}_{cmd}\|} & \|\boldsymbol{f}_{cmd}\| > |f_r| \\ \boldsymbol{v} = \mathbf{0} & otherwise \end{cases} \tag{4.15}$$

Where $\boldsymbol{v}$ is the linear velocity vector of the end-effector, $\boldsymbol{f}_{cmd}$ is the force vector command issued to the controller, its magnitude is $\|\boldsymbol{f}_{cmd}\|$. $|f_r|$ is the sensitivity threshold magnitude (motion is only executed when the force command reaches a value above this threshold), and $b'$ is the motion constant that defines the rate of conversion from force measurement to velocity.

The angular motion of the end-effector is controlled by:

$$\begin{cases} \boldsymbol{\varpi} = \frac{\boldsymbol{m}_{cmd}(\|\boldsymbol{m}_{cmd}\| - |\tau_r|)}{\beta' \|\boldsymbol{m}_{cmd}\|} & \|\boldsymbol{m}_{cmd}\| > |\tau_r| \\ \boldsymbol{\varpi} = \mathbf{0} & otherwise \end{cases} \tag{4.16}$$

Where $\boldsymbol{\varpi}$ is the angular velocity vector of the end-effector, $\boldsymbol{m}_{cmd}$ is the moment command issued to the controller, its magnitude is $\|\boldsymbol{m}_{cmd}\|$. $|\tau_r|$ is the sensitivity threshold (the motion is only valid when the input command value is higher than this threshold), and $\beta'$ is a motion constant that defines the rate of conversion from moment measurement to velocity.

Finally, from equation (4.15) and (4.16), the linear and angular velocity of the end-effector $\dot{\boldsymbol{x}}$ is given by:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\varpi} \end{bmatrix} \tag{4.17}$$

The robot joint velocities $\dot{\boldsymbol{q}}_{hg}$, that satisfy the proposed hand-guiding motion, are calculated using the pseudo inverse of the Jacobian:

$$\dot{\boldsymbol{q}}_{hg} = \mathbf{J}^{\dagger} \dot{\boldsymbol{x}} \tag{4.18}$$

Where $\mathbf{J}^{\dagger}$ is calculated using the singular value decomposition of the Jacobian matrix:

$$\mathbf{J} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{\mathrm{T}} \tag{4.19}$$

Where matrices $\mathbf{U}$, $\boldsymbol{\Sigma}$ and $\mathbf{V}$ are the result of the singular value decomposition of the Jacobian. Thus, the pseudo inverse is given by:

$$\mathbf{J}^{\dagger} = \sum_{i=1}^{n} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \tag{4.20}$$

Where $n$ is the number of non-zero diagonal elements of matrix $\boldsymbol{\Sigma}$, $\sigma_i$ is the $i^{th}$ non-zero diagonal element of matrix $\boldsymbol{\Sigma}$, $\boldsymbol{v}_i$ is the $i^{th}$ column of matrix $\mathbf{V}$, and $\boldsymbol{u}_i$ is the $i^{th}$ column of matrix $\mathbf{U}$. To achieve better numerical stability, and according to [106], the alternate Jacobian is used.

The robot can be controlled at joint level using the calculated angular velocities of the joints $\dot{\boldsymbol{q}}$. Nevertheless, many robots does not support real-time control at the joint velocity level, namely industrial robots, however they are normally provided with real-time position control loop at joint level. In this study for example, we demonstrated the proposed method in a sensitive industrial robot (KUKA LBR iiwa) using its internal

**Algorithm 4.1** Precision hand-guiding - control algorithm.

Input : $\boldsymbol{q}_{ini}$ joints positions feedback from robot
$\dot{\boldsymbol{x}}$ calculated EEF velocity (4.17)
$\alpha$ scaling factor for joints limits avoidance
$\triangle t$ time interval for loop update cycle
$\boldsymbol{\varepsilon}$ vector of acceptable Cartesian errors
Output : $\boldsymbol{q}_{goal}$ joitns goal positions

01 : $\boldsymbol{x}_{ini} = \boldsymbol{f}(\boldsymbol{q}_{ini})$ % *Forward kinematics*
02 : $\boldsymbol{x}_g = \boldsymbol{x}_{ini} + \alpha\dot{\boldsymbol{x}}\triangle t$
03 : $\boldsymbol{x} = \boldsymbol{x}_{ini}$
04 : $\boldsymbol{q} = \boldsymbol{q}_{ini}$
05 : $\boldsymbol{e} = (\boldsymbol{x}_g - \boldsymbol{x})$ % *Calculate error*
06 : **while** $(norm(\boldsymbol{e}) > \boldsymbol{\varepsilon})$ % *Loop until error is small enough*
07 : $\mathbf{J}^{\dagger} = \phi(\boldsymbol{q})$ % *Pseudo inverse of the Jacobian*
08 : $\boldsymbol{q} = \boldsymbol{q} + \mathbf{J}^{\dagger}\boldsymbol{e}$ % *Joints positions update*
09 : $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q})$ % *Forward kinematics*
10 : $\boldsymbol{e} = (\boldsymbol{x}_g - \boldsymbol{x})$ % *Calculate error*
11 : **Loop**
12 : $\boldsymbol{q}_{goal} = \boldsymbol{q}$

joint position servo control feature. In this scenario, from equation (4.20), we can deduce a joint level position control according to the following procedure:

1. For each update cycle of the control loop the joint positions $\boldsymbol{q}_{ini}$ are acquired from the robot encoders;

2. At each update cycle, a feedback from the FT sensor is also obtained, from which the command velocity $\dot{\boldsymbol{x}}$ of the EEF is calculated as described previously;

3. Using the above information the desired joint positions are calculated and afterwards updated to the position control loop of the robot controller. In this study, the Algorithm 4.1 is used to calculate the destination joint positions, which will be updated to the position control loop, provided by the robot manufacturer. Such update is performed at each control cycle.

Algorithm 4.1 details the methodology to achieve a high level of numerical precision, although the final precision of robot motion will depend on the precision performance of the real-time position control loop of the robot being used.

### 4.1.8 Referencing motion to the EEF

Using the presented framework the operator can only translate the EEF along the x, y or z axis of the base frame of the robot, one at a time. But in some situations the operator may require to perform an inclined linear motion of the EEF, to solve this, another operation mode is proposed, in this operation mode the operator can reference the motion to the EEF frame, the operator can enter this operation mode by pressing an external button on the robot while the EEF is static.

In this operation mode, the force command is referenced to the end-effector, as such the robot moves in the direction of the maximum component of the hand-guiding

Table 4.2: Values of $a^e$,$b^e$ and $c^e$

| $a^e$ | $b^e$ | $c^e$ | *Condition* |
|---|---|---|---|
| 1 | 0 | 0 | $\|f_x^e\| > \|f_y^e\|$ && $\|f_x^e\| > \|f_z^e\|$ |
| 0 | 1 | 0 | $\|f_y^e\| > \|f_x^e\|$ && $\|f_y^e\| > \|f_z^e\|$ |
| 0 | 0 | 1 | $\|f_z^e\| > \|f_x^e\|$ && $\|f_z^e\| > \|f_y^e\|$ |

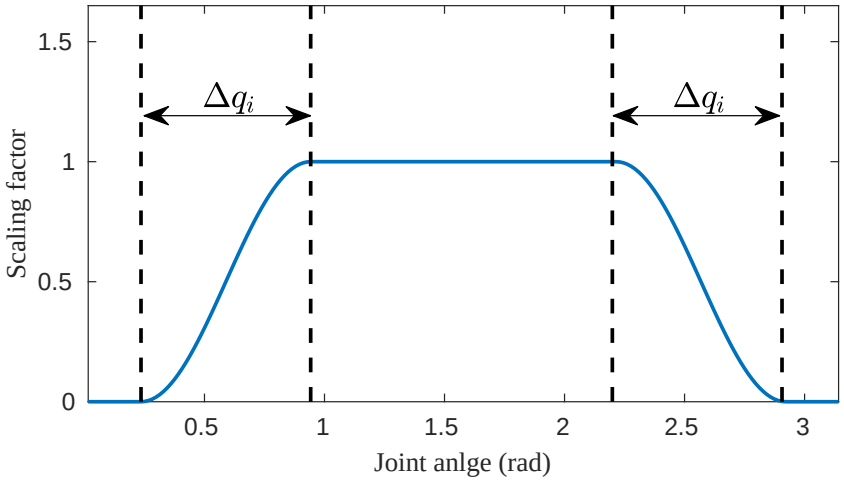


Figure 4.4: Scaling factor for joint limits avoidance.

force described in EEF frame. Where the hand-guiding force command is calculated from:

$$\boldsymbol{f}_{cmd} = \mathbf{R}_e^b \begin{bmatrix} a^e & 0 & 0 \\ 0 & b^e & 0 \\ 0 & 0 & c^e \end{bmatrix} \begin{bmatrix} f_x^e \\ f_y^e \\ f_z^e \end{bmatrix} \tag{4.21}$$

Where ($a^e$,$b^e$,$c^e$) values are calculated as described in Table 4.2.

### 4.1.9 Joints limits

During the precise hand-guiding, the operator might drive the robot near the joint limit. In the case of KUKA iiwa robot, this results in an error causing the manipulator to stop in place. To solve this problem we propose to utilize a scaling factor $\alpha_{min}$, this scaling factor is used to scale down the linear/angular velocity of the end-effector near the limit. For each joint $i$ a scaling factor is calculated:

$$\alpha_i = \begin{cases} \frac{1-cos\left(\pi \frac{q_i - q_{i,ll}}{\triangle q_i}\right)}{2}, & (q_{i,ll} + \triangle q_i > q_i > q_{i,ll}) \\ & \&\& \, (\dot{q}_i < 0) \\ \frac{1-cos\left(\pi \frac{q_i - q_{i,ul}}{\triangle q_i}\right)}{2}, & (q_{i,ul} > q_i > q_{i,ul} - \triangle q_i) \\ & \&\& \, (\dot{q}_i > 0) \\ 1 & otherwise \end{cases} \tag{4.22}$$

Figure 4.4 shows a plot of $\alpha_i$ with joint angle $q_i$. The value of the scaling factors $\alpha_{min}$ is calculated as the product of the elements $\alpha_i$:

$$\alpha_{min} = \prod \alpha_i \tag{4.23}$$

For robots controlled at joint angular velocities level, the angular velocity command of the joints $\dot{\boldsymbol{q}}_{\boldsymbol{cmd}}$ is calculated:

$$\dot{\boldsymbol{q}}_{\boldsymbol{cmd}} = \alpha_{min} \dot{\boldsymbol{q}}_{hg} \tag{4.24}$$

For industrial robotic manipulators that support real-time control functionality, a stream of joint angular positions is used to command the robot (not the angular velocities, this applies for the KUKA iiwa robot), in such case the previous strategy for scaling down the angular velocities does not apply, and the previous method needs to be modified to suit the control strategy of such robots. In the following the method for joints limits avoidance applied for the precise hand-guiding for the KUKA iiwa robot is explained: in the beginning $\alpha_{min}$ is considered to be one, consequently, the angular positions $\boldsymbol{q}_1 = \boldsymbol{q}$ are calculated using Algorithm 4.1 (previously presented in 4.1.7), the subscript 1 in $\boldsymbol{q}_1$ is used to denote the first execution of the Algorithm 4.1. Afterwards, using $\boldsymbol{q}_1$ the value of $\alpha_{min}$ is calculated using equations (4.22, 4.23). Now in case $\alpha_{min}$ at $\boldsymbol{q}_1$ is equal to one, then the calculated angular positions $\boldsymbol{q}_1$ are updated to the controller of the robot. Otherwise if ($\alpha_{min} < 1$), then the angular positions $\boldsymbol{q}_1$ are discarded, and the value of $\alpha_{min}$ is used to scale down the velocity of the end-effector $\dot{\boldsymbol{x}}$ into $\alpha_{min}\dot{\boldsymbol{x}}$. Afterwards, the inverse kinematics resolution is re-initiated using the Cartesian velocity $\alpha_{min}\dot{\boldsymbol{x}}$ instead of $\dot{\boldsymbol{x}}$ (in such way the user feels that the motion is becoming harder when pushing towards the limit of the joints). Then after the recalculation the resulting positions of the joints $\boldsymbol{q}_2$ are applied to the internal control loop of the robot (resulting in a scaled down velocity at the EEF). Now in case the coworker kept pushing against the limit, the scaling factor value reaches zero, and the robot stops in place (does not exceed its physical limit). In case the user moved the robot in the opposite direction, away from the limit, the robot moves freely, this is guaranteed due to the angular velocity inequality (after the && operator of expression 4.22). The proposed method should apply for any industrial robot with realtime position control functionalities, in the case of the KUKA iiwa example an alert light, yellow, is turned on once $\alpha_{min}$ is less than one Figures 4.5 and 4.6, this helps giving a visual feedback to the worker, at the same time markers are added on the joints Figures 4.7 and 4.8, by looking at it the coworker is able to realize which joint is nearing the limit.

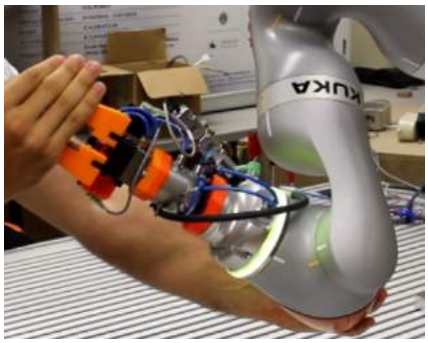

Figure 4.5: Light alert (yellow) turns on when joint five is near the limit.

Figure 4.6: Light alert turns on, when pushing joint four near its limit.

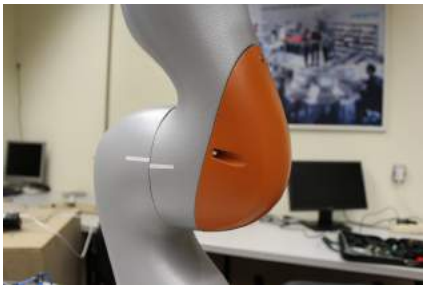

Figure 4.7: Joint two near the zero, white markers are near each others.

Figure 4.8: Joint four near the limit, yellow and white markers are nearing each others.

### 4.1.10 Experiments

The experiments were performed using KUKA iiwa 7 R800 robotic manipulator, the proposed algorithm could be implemented in Java using Sunrise.Workbench (the standard development environment for KUKA iiwa robot), in such case the computations run directly inside the controller of the robot. Otherwise, the algorithm could be implemented remotely (run on outside computer). In such a case, client-sever software [11] shall be used to acquire feedback and to stream motion commands to the robot. In the following experiments a qualitative/quantitative assessment of the performance of the proposed method is reported.

In the first test an assembly operation requiring precision positioning and fine tuning is demonstrated, where a qualitative assessment of the performance of the proposed algorithm is discussed. In the second test, the performance of the proposed method is measured quantitatively. In the third test a comparison between off-the-shelf hand-guiding functionality provided by KUKA and the proposed method is presented. Finally, the precision hand-guiding is implemented in a programming by demonstration application, consequently, any user can program the robot easily and with precision, in a time efficient manner and without the need to write any code.

Figure 4.9: Test 1: Precision hand-guiding for assembly operation.

#### 4.1.10.1 Test 1

The test is shown in Figure 4.9. The robot is used as an assistive third hand for precise assembly operations[1]. Using the hand-guiding application it is required to move parts/instruments (attached at the EEF of the robot) precisely into the fixing location. Afterwards, the instruments are held in place by the robot, allowing the coworker to apply the fixtures. Such approach makes less workers occupied in manual tasks on the factory floor, also in hard-to-access locations the proposed method offers better safety by reducing the risk of fallen parts, which might happen if humans are involved. For mounting heavier parts, the proposed method works as a force magnifier, where the weight of the instrument is balanced by the force from the robot, reducing the effort required from the coworker. In this test, the precision hand-guiding algorithm was implemented inside the robot controller, in a single thread program, with an update rate of 118 Hz. The control algorithm compensates automatically for the part weight such that when no force is applied by the user, the part is held in place by the robot. Different tests demonstrated that in the 1st motion group the robot position along x, y and z can be precisely adjusted, in the 2nd motion group the axis of the end-effector is oriented smoothly in 3D space, while in the 3rd motion group we can rotate around the end-effector axis with accuracy, Figure 4.9.

#### 4.1.10.2 Test 2

The precise hand-guiding algorithm was implemented in a multi-threaded program inside the KUKA iiwa 7 R800 controller. The main thread is used to run the proposed control algorithm, the second thread is used to (1) connect to an external computer using a TCP/IP network, (2) acquire the measurements of the force and the position at the end-effector, which are streamed over the network to an external computer using TCP/IP protocol.

In this test the user starts the proposed hand-guiding application on the KUKA iiwa controller, then he/she hand-guides the robot in the x, y and z directions. The force and the Cartesian position measurements were acquired from the robot, then sent to an external computer, where they were recorded along with their corresponding time stamps.

Figure 4.10 shows end-effector position data recorded during the test. It is noticed that when the hand-guiding force along one of the axes reaches the predefined limit,

[1]This application was developed during the European project Colrbot [3] (Horizon 2020) for the use in satellite assembly operations.

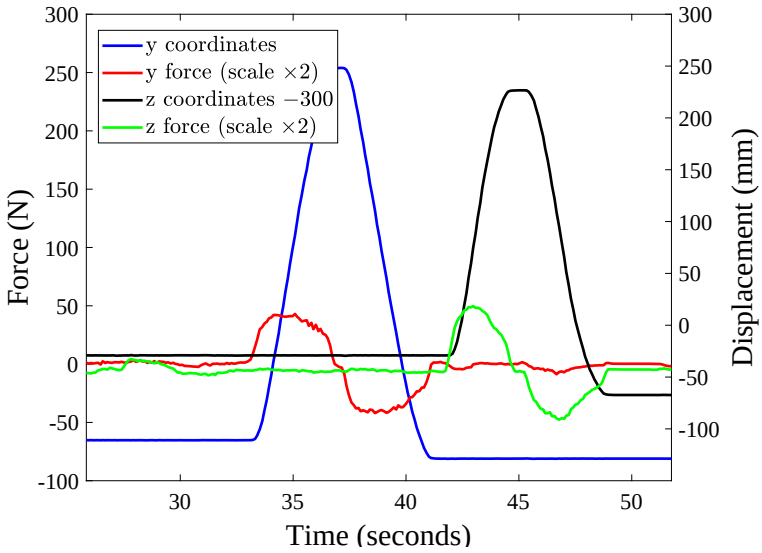


Figure 4.10: Test 2: End-effector position along y and z axes according to applied force.

the controller is activated, and the robot moves in the corresponding direction, in the time interval [33;37] seconds, the operator applies a force bigger than the limit on the positive y direction, this causes the robot to move in the positive y direction. According to the force, the robot moves with an average velocity of 99.7 mm/second. While the coordinates in the x and z direction are almost fixed. From the acquired data we see that the average position in the x direction is 624.16 mm, and the average position in the z direction is 307.47 mm (note that the z coordinate values shown in Figure 4.10 are displaced by -300mm), also we find that the maximum deviation from the average in the x direction is 0.091 mm Figure 4.11, while the maximum deviation from the average in the z direction is 0.092 mm Figure 4.12.

#### 4.1.10.3 Test 3

To test the performance of the proposed method against the hand-guiding at the joints level, we compare with KUKA iiwa off-the-shelf hand-guiding, where the following test is proposed. The manipulator KUKA iiwa 7 R 800 is used for positioning operation of a tool (using precise hand-guiding and KUKA hand-guiding), during each positioning operation the following data were recorded:

- x, y, z position and orientation of the end-effector;
- Vibrations level at the end-effector (measured using IMU);
- Joint angles.

**Experimental setup**

The experimental setup proposed is demonstrated in Figure 4.13. In this setup a tool is attached to the end-effector, and the objective is to insert the tool in a tool holder using

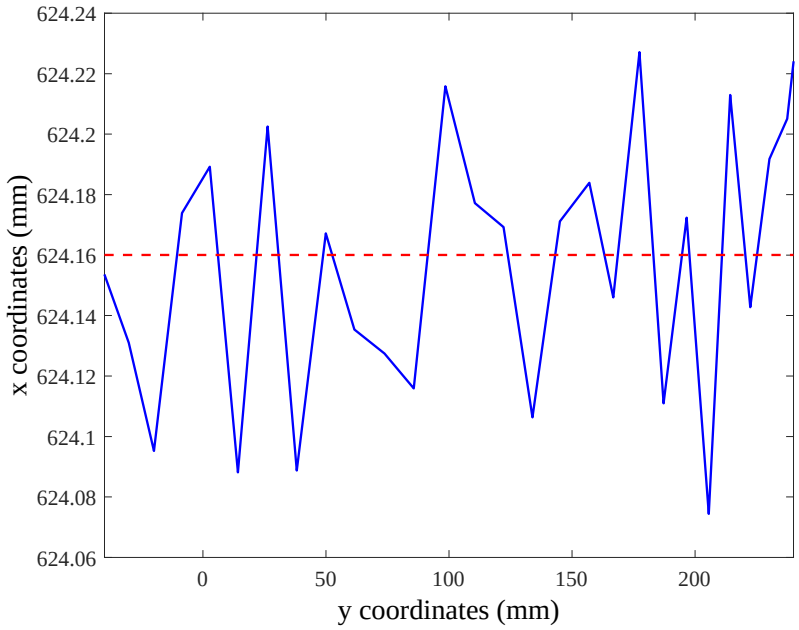


Figure 4.11: Test 2: Nominal end-effector path against the real robot end-effector path in plane xy.

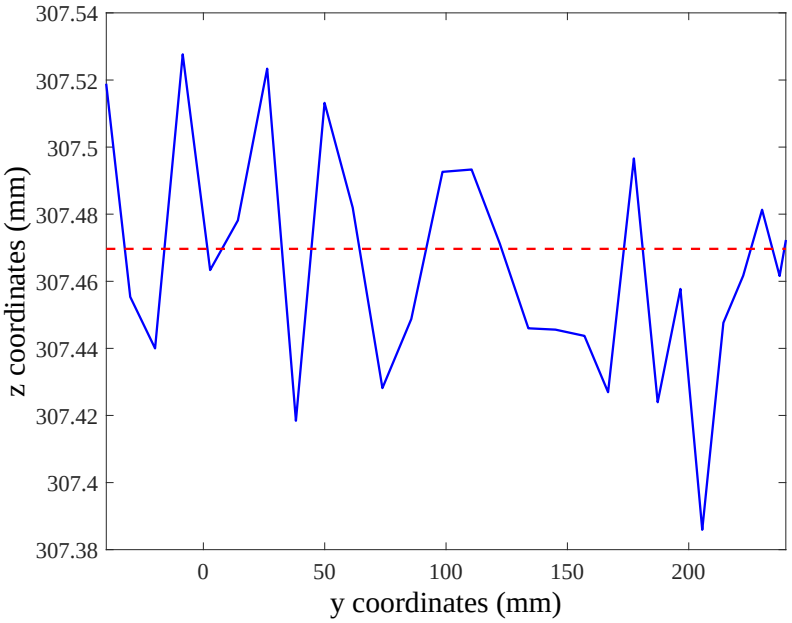


Figure 4.12: Test 2: Nominal end-effector path against the real robot end-effector path in plane zy.

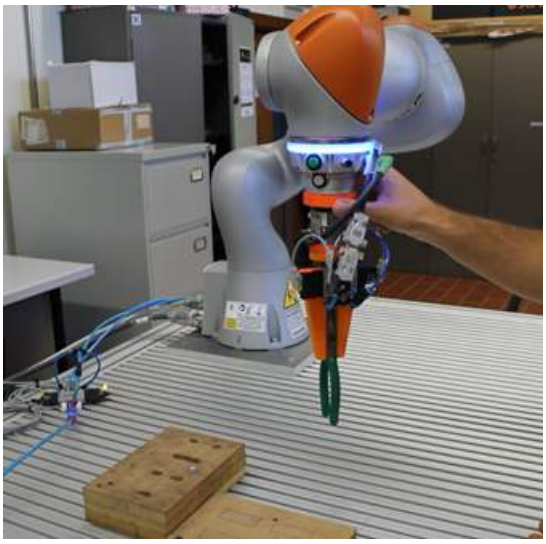

Figure 4.13: Test 3: Experimental setup.

the hand-guiding functionality. To measure the level of vibrations at the end-effector an accelerometer is fixed firmly to it Figure 4.13, the acceleration data are acquired by an external computer connected to the accelerometer hub using USB cable, at the same time the data of joints/end-effector positions are acquired by the same computer from the robot using a TCP/IP connection. A video segment showing Test 3 while using the proposed method is in [137]. Another video segment showing Test 3 while using KUKA off-the-shelf hand guiding is in [138].

### Positioning results

During the placement operation of the tool, the x, y and z positions of the EEF were recorded. The test was performed two times, in the first the precise hand-guiding was used, in the second the off-the-shelf hand-guiding of KUKA iiwa was utilized, position data were collected from robot controller and relayed over a TCP/IP connection to the computer. Figure 4.14 presents the coordinates of end-effector during the prescribed task using the precise hand-guiding function. While Figure 4.15 presents the coordinates of end-effector during the prescribed task using the KUKA off-the-shelf hand-guiding function. From the results we notice that the precise hand-guiding offers smoother motion. An expected result, since that in the precise hand-guiding the robot is controlled at EEF level, so no effort is required by the operator for constraining end-effector's orientation during the motion, unlike the KUKA off-the-shelf hand-guiding which operates at joints level.

At the same time the orientation of the end-effector was acquired and relayed over the TCP/IP connection to an external computer. Figure 4.16 represents the orientation results of the end-effector (represented by a unit quaternion) for the prescribed task using the precise hand-guiding function. Figure 4.17 represents the orientation results of the end-effector during the positioning task using the KUKA off-the-shelf hand-guiding function. In the precise hand-guiding the robot is able to keep the same

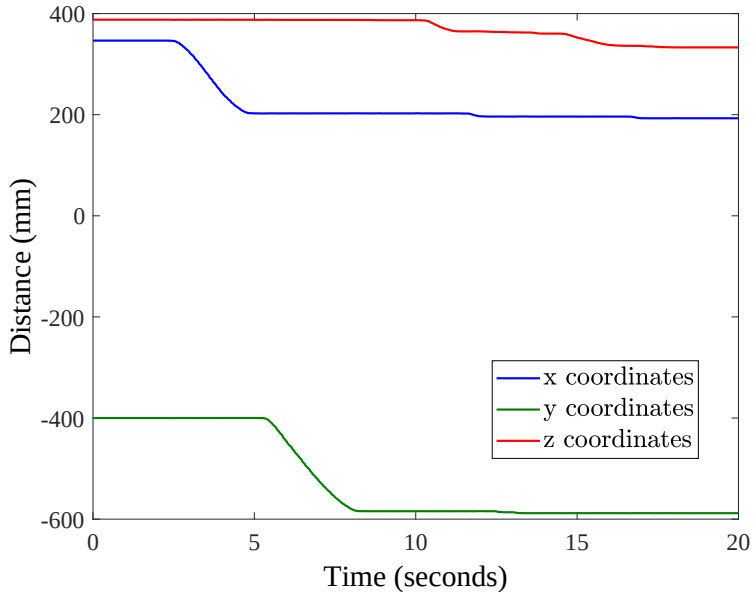


Figure 4.14: Test 3: Coordinates of the end-effector during the positioning task using the precise hand-guiding function.

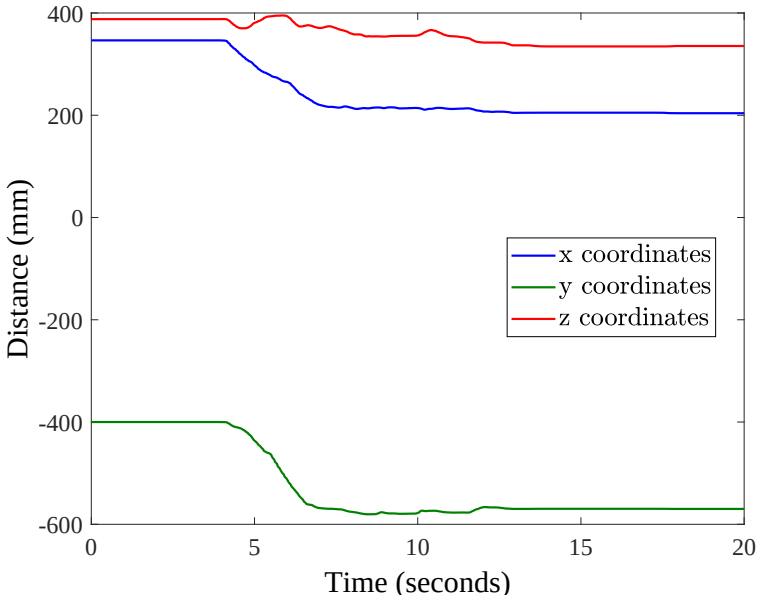


Figure 4.15: Test 3: Coordinates of the end-effector during the prescribed task using the KUKA off-the-shelf hand-guiding.

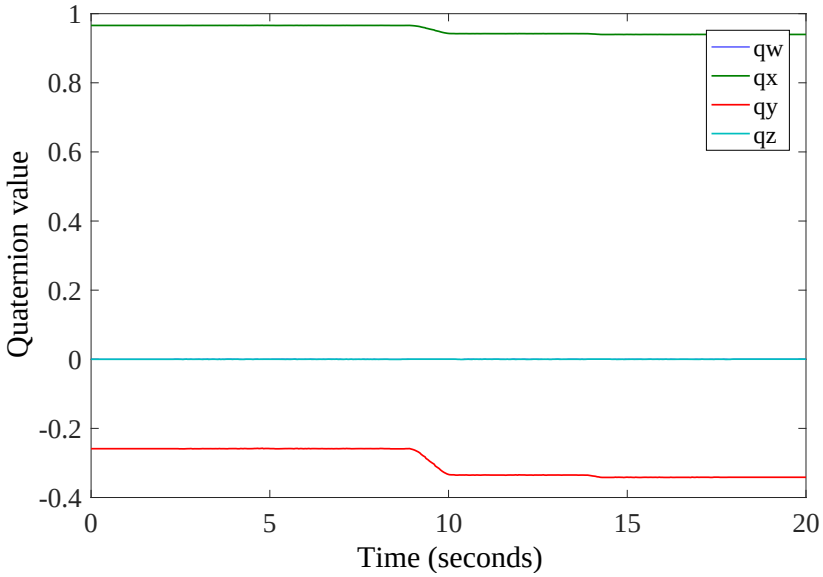


Figure 4.16: Test 3: Orientation of the end-effector in the precise hand-guiding function.

orientation during displacement (x,y,z) motion, while it is almost impossible to achieve the same while using the KUKA off-the-shelf hand-guiding, giving that KUKA off-the-shelf hand-guiding is controlled in joint space, the orientation is as precise as the coworker can be.

### Joint angles

Figure 4.18 and Figure 4.19 show the joints angles for the positioning during the precise and the KUKA hand-guiding respectively. From the figures we find that the precise hand-guiding results in smoother angles of the joints over KUKA off-the-shelf hand-guiding.

### Vibrations level at the end-effector

For measuring the vibrations level at the EEF, an accelerometer was firmly attached to it, Figure 4.20. Then, a gravity acceleration compensation was applied as described in Appendix C.

During the hand-guiding positioning operation of the tool (using KUKA and precise hand-guiding), acceleration data were acquired from the accelerometer, and the results were plotted against time during the manipulation operation.

From Figure 4.21, we notice that the acceleration levels at the end-effector in the case of the precise hand-guiding are less than those of the KUKA hand-guiding, where the maximum magnitude of acceleration in case of KUKA hand-guiding reached 2.38 $m/sec^2$, while in case of the precise hand-guiding 1.2 $m/sec^2$.

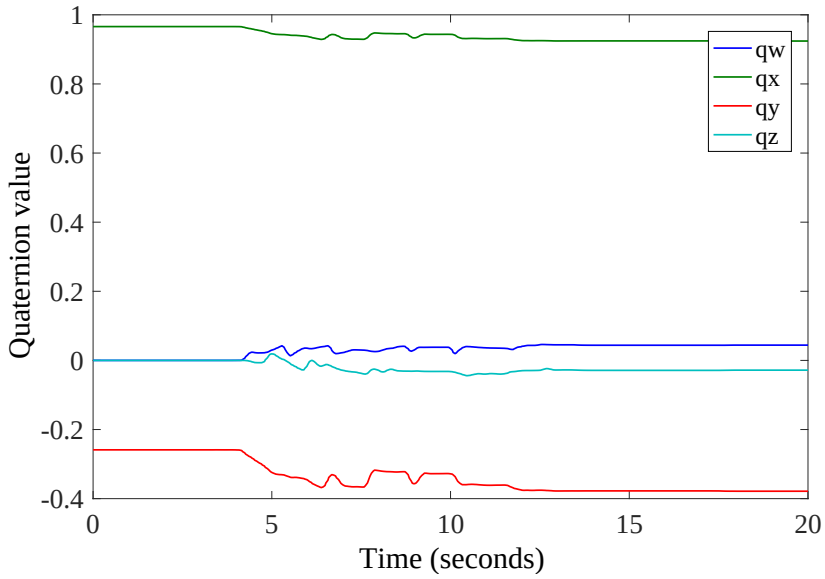


Figure 4.17: Test 3: Orientation of the end-effector during the KUKA off-the-shelf hand-guiding function.

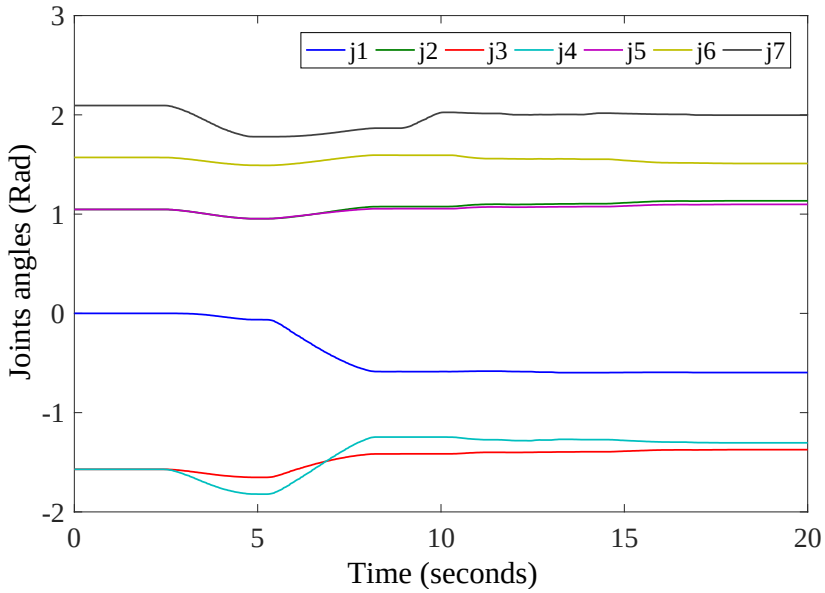


Figure 4.18: Test 3: Joint angles during positioning operation for the precise hand-guiding.

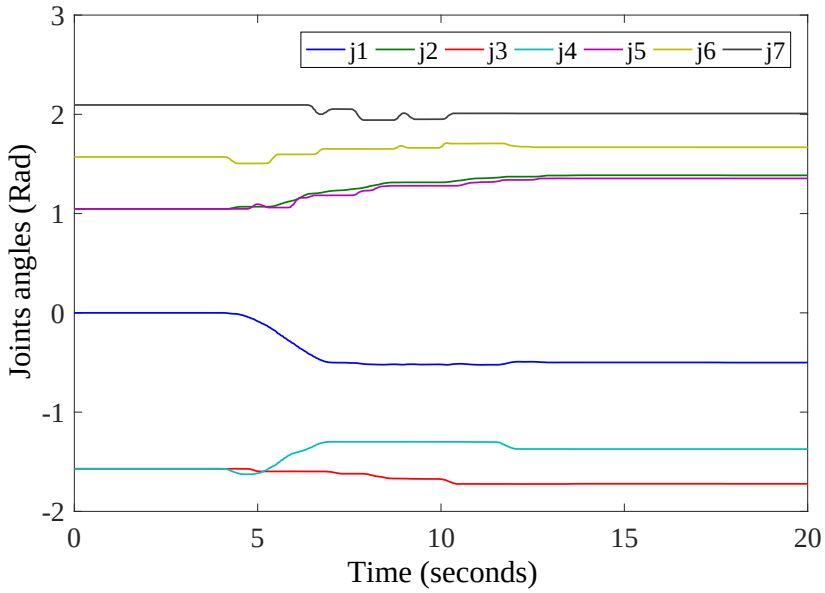


Figure 4.19: Test 3: Joint angles during positioning operation for the KUKA off-the-shelf hand-guiding.

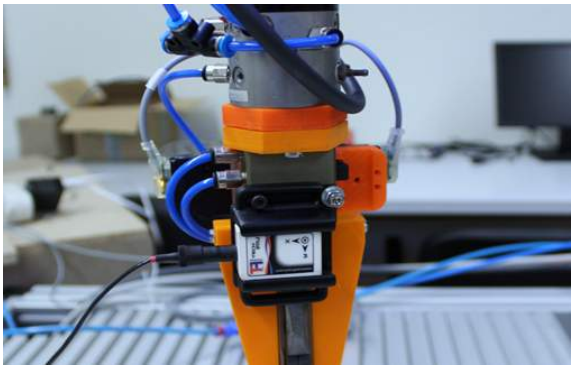

Figure 4.20: Test 3: Accelerometer attached at the end-effector.

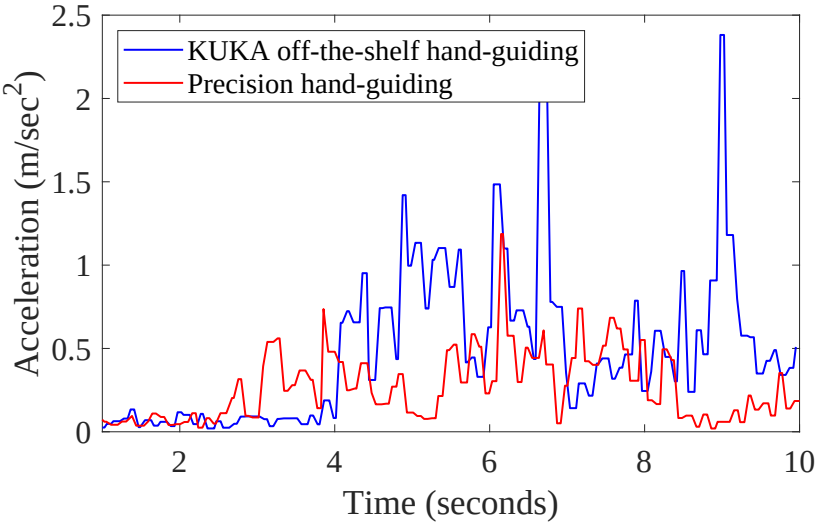


Figure 4.21: Test 3: Acceleration magnitude (after gravity compensation) at end-effector during precise/KUKA hand-guiding.

#### 4.1.10.4 Test 4

To demonstrate the intuitiveness of the precision hand-guiding developed in this study, and to show its importance for facilitating robot teaching, a programming by demonstration application is developed Figure 4.22 (a video segment showing the application is also in [139]).

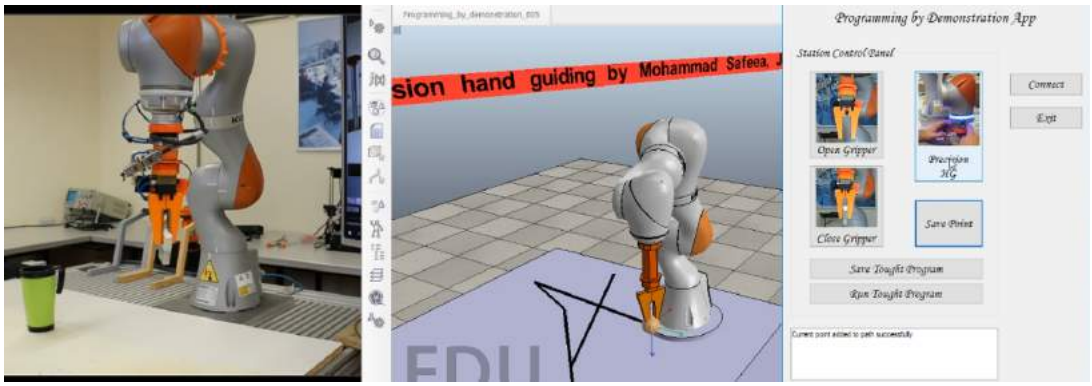


Figure 4.22: Programming by demonstration system based on the precision hand-guiding method, (left) real robot, (middle) real-time simulation of the robot/gripper, the black lines inside the simulation represent the taught path, (right) HMI interface for easy control of the teaching process.

This application includes a friendly interface written using C# (right of Figure 4.22). This interface allows the user to activate the precision hand-guiding application, to open/close the gripper attached to the robot, and to save the exact coordinates of the EEF for constructing the taught path. The application also implements a Vrep

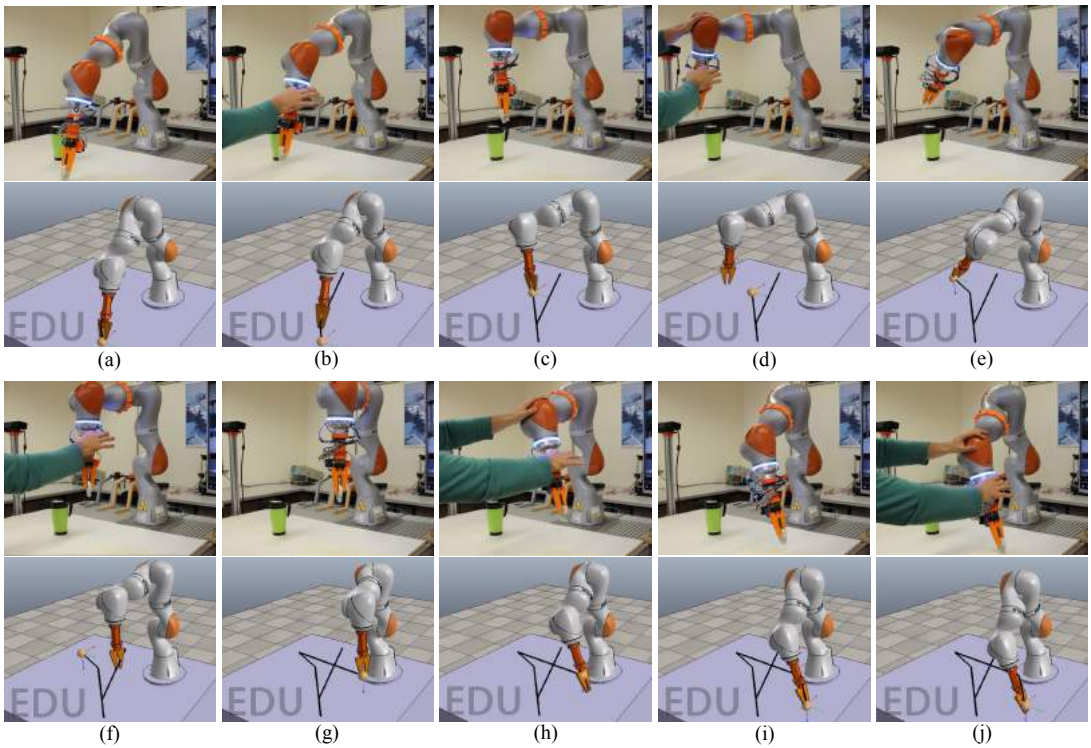


Figure 4.23: Teaching an operation to KUKA iiwa using a programming by demonstration application based on the precision hand-guiding. A virtual reality simulation of the robotic cell in Vrep is used to show the coworker a feedback of the programmed path during the teaching process, allowing the user to visualise, modify and verify the taught path while performing the operation.

simulation used to visualise the taught path during the programming by demonstration process. In such a case, the user is able to verify/modify the taught path intuitively.

Sub-figures (a to j) in Figure 4.23 show the developed application in Test 4. Upper sub-figures correspond to the real robot being programmed by demonstration using the precision hand-guiding method (precision hand-guiding algorithm is running inside the robot controller), lower sub-figures show a real-time simulation of the real robot programmed in Vrep, the simulation is updated in real-time according to sockets received from the robot representing its state (joint angles, gripper state), the simulation also shows the path taught in black lines. The precision hand-guiding is used to teach the robot the exact coordinates to pick an object (box) in sub-figures (a,b), then to perform an operation (empty the contents of the box into a cup) in sub-figures (c,d,e), finally to put the box back on a specific location on the table sub-figures (f,g,h,i,j). It took the user less than 3 minutes to teach the robot the whole operation. After the teaching process, the robot can perform the taught path by clicking on the corresponding button from the HMI interface.

### 4.1.11 Summary

A novel precision hand-guiding method at the end-effector level for collaborative robots is proposed. Experimental tests demonstrated that with the proposed method it is possible to hand-guide the robot with accuracy, with lower level of vibrations, and in a natural way by taking advantage of the proposed three motion groups. It was also demonstrated that the system is intuitive and compares favourably with KUKA off-the-shelf hand-guiding.

## 4.2 Redundancy Resolution & Precision Hand-Guiding

In the previous section, the precision hand-guiding is presented as a tool that allows unskilled users to interact and position the EEF intuitively and precisely. However, the least norm solution was used in generating the motion, such that the redundancy resolution of redundant manipulators was not fully explored. Many industrial applications require precise positioning at the end-effector (EEF) level inside cluttered environments, where manipulator's redundancy is required. This section addresses the subject of precision in hand-guiding at EEF level while using the redundancy for in-contact obstacle navigation. In the presence of a contact with an obstacle, the proposed null space control method actuates in a way that the manipulator slides compliantly with its structure on the body of the obstacle while preserving the precision of the hand-guiding motion at EEF level. Force/torque (FT) data from a FT sensor mounted at the robot flange are the input for EEF precision hand-guiding while the torque data from the joints of the manipulator are used to calculate the contact between robot structure and obstacles. Experimental tests were carried out successfully using a KUKA iiwa industrial manipulator with 7 degrees of freedom (DoF). Where, the EEF is hand-guided on a straight line while the robot is sliding on the obstacle with its structure, results indicate the precision of the proposed method.

### 4.2.1 Proposed method

The method is described in Figure 4.24. Where using the force and torque measurements from an external FT sensor attached to the flange, and based on the precision hand-guiding algorithm (in the previous section), the user is able to hand-guide a sensitive redundant manipulator precisely at the EEF level. At the same time, using the torques feedback from the integrated torque sensors at the joints of the sensitive manipulator, it is possible to perform automatic redundancy resolution, where upon a contact with an external obstacle (wooden box in Figure 4.24), the robot is able to adjust the internal motion manifold to avoid the obstacle. As a result, the robot slides with its structure on the body of the obstacle while keeping the precision at the EEF level according to the hand-guiding motion.

### 4.2.2 Control strategy

The inputs to the proposed controller are:

- Force and torque measurements from an external FT sensor attached at the flange of the robot;
- Torque measurements from the integrated torque sensors at the joints of the robot.

Most industrial robots, including the KUKA iiwa 7R800 and 14R820, are controlled at the kinematic level, by streaming the joints reference positions or velocities to the robot. As such, the proposed control strategy operates at the kinematic level, it includes superimposing the angular velocities $\dot{\boldsymbol{q}}_{hg}$, in equation 4.18, that satisfy hand-guiding motion, with the angular velocities in the null space $\dot{\boldsymbol{q}}_{n}$ that satisfy the in-contact

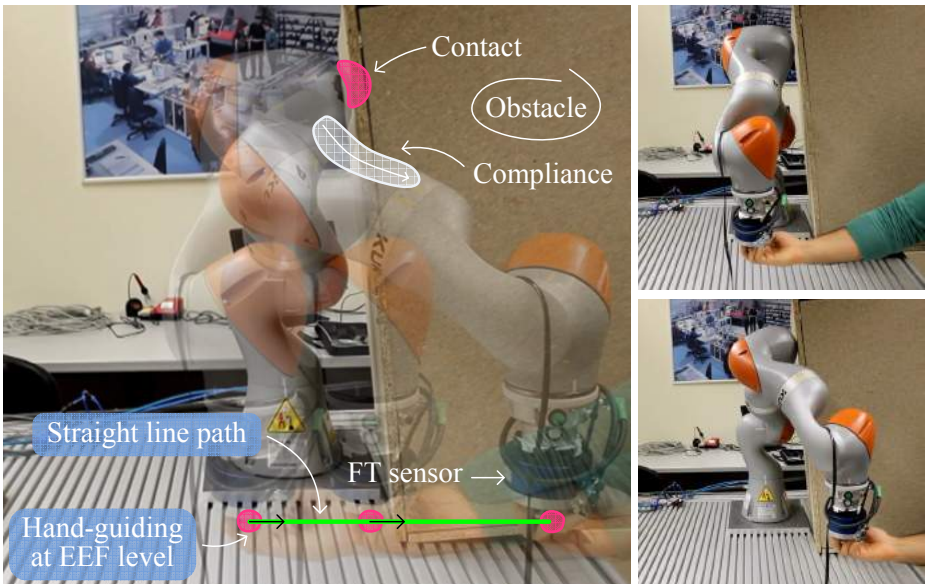


Figure 4.24: Precision hand-guiding on a straight line (along y axis) of a redundant robot subject to a contact with obstacle. An external FT sensor is attached at the robot flange.

obstacle navigation. The mathematical formulas presented hereafter are kept generic. Thereby, the method is applicable for any sensitive robot possessing redundancy with respect to its required task.

#### 4.2.2.1 Joint torques

Once a contact is initiated with the obstacle, extra torques due to contact forces $\boldsymbol{\tau}_{ex}$ start to appear in the joints:

$$\boldsymbol{\tau}_{ex} = \boldsymbol{\tau}_r - \boldsymbol{\tau}_{dyn}$$

Where $\boldsymbol{\tau}_{ex}$ is the vector of external torque due to external forces, $\boldsymbol{\tau}_r$ is the raw torque measurements from the sensors at robot joints, and $\boldsymbol{\tau}_{dyn}$ are the torques due to robot's own motion and gravity. The latter includes the torques due to (1) joints angular acceleration, (2) Coriolis/centrifugal effect, (3) friction at joints and (4) torques due to gravity, it is given by the equation:

$$\boldsymbol{\tau}_{dyn} = \mathbf{A}\ddot{\boldsymbol{q}} + \mathbf{C}\dot{\boldsymbol{q}} + \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g} \tag{4.25}$$

Where $\mathbf{A}$ is the mass matrix of the robot, $\ddot{\boldsymbol{q}}$ is the vector of joints angular acceleration, $\mathbf{C}$ is Coriolis matrix of the robot, $\dot{\boldsymbol{q}}$ is the vector of joints angular velocities, $\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are the torques due to friction in the joints, and $\boldsymbol{g}$ are the torques due to gravitational effect. In such a case, the accuracy in calculating $\boldsymbol{\tau}_{dyn}$ depends mainly on the precision of the dynamical model used to describe the manipulator's dynamics.

When it comes to KUKA iiwa robots, the Sunrise Controller calculates internally the vector of the external torques, allowing the user to acquire the numerical value of

vector $\boldsymbol{\tau}_{ex}$. In this study, we opted to use the KUKA Sunrise Toolbox [11], where we are able to retrieve the components of $\boldsymbol{\tau}_{ex}$ using a TCP/IP communication in MATLAB.

#### 4.2.2.2 Joint torques compensation

In addition to the joint torques due to the external contact forces with the obstacles, the toque vector $\boldsymbol{\tau}_{ex}$ includes some extra components caused by the forces/moments due to the (1) hand-guiding, (2) weight of the tool mounted at FT sensor, and (3) the weight of the FT sensor. The effect of those forces/moments on joints torques shall be accounted for. This procedure is referred to as the torques compensation, in which two main factors are recognized:

1. Compensation for the torques due to sensor's weight, which also includes the weight of the adaptor flange used to mount the FT sensor on the robot. The weight of the sensor and its Center-Of-Mass (COM) are previously known. Consequently, the torque compensation is applied directly after calculating the Jacobian associated with COM as described later in the **Sensor weight compensation**;

2. Compensation for the torques due to external force/moment acting at the FT sensor. This force/moment is measured by the FT sensor, it includes the weight of the tool mounted at the FT sensor and the hand-guiding force/moment applied by the human, as described later in the **Hand-guiding-wrench/tool-weight compensation**.

**Sensor weight compensation:** For robot motions with relatively small accelerations, which is the case in precision hand-guiding, the forces/moments due to the inertia of the FT sensor and the mounting flange can be neglected in comparison to their weight. Thus, to compensate the torques generated due to the weight of the sensor and the mounting flange $\boldsymbol{\tau}_{ws}$, the Jacobian $\mathbf{J}_{ws}$ associated with their COM is utilized:

$$\boldsymbol{\tau}_{ws} = \mathbf{J}_{ws}^{\mathrm{T}} \begin{bmatrix} 0 & 0 & -w_s & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{4.26}$$

Where $w_s$ is the weight of the sensor (including mounting flange). The previous equation is valid for base mounted robots, where the base is mounted horizontally and the z axis is pointing up, otherwise the orientation of the base frame shall be considered.

**Hand-guiding-wrench/tool-weight compensation:** The torques generated due to external forces/moments acting on the FT sensor shall be compensated for. Those forces/moments include the hand-guiding wrench (force/moment), the attached tool's weight and the inertial forces due to its acceleration. For calculating the torques $\boldsymbol{\tau}_{fs}$ due to external forces/moments, the Jacobian $\mathbf{J}_s$ associated with the origin of the measurement frame of the FT sensor is considered:

$$\boldsymbol{\tau}_{fs} = \mathbf{J}_s^{\mathrm{T}} \begin{bmatrix} \mathbf{R}_s^b & 0 \\ 0 & \mathbf{R}_s^b \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_{fs} \\ \boldsymbol{m}_{fs} \end{bmatrix} \tag{4.27}$$

Where $\boldsymbol{f}_{fs}$ and $\boldsymbol{m}_{fs}$ are the force and the moment measured at the FT sensor, respectively. Those force and moment quantities are due to the hand-guiding force/moment plus the weight/inertial force/moment of the tool. The joint torques vector $\boldsymbol{\tau}_{fs}$ is due to $\boldsymbol{f}_{fs}$ and $\boldsymbol{m}_{fs}$, and $\mathbf{R}_s^b$ is the rotation matrix from the measurement frame of the sensor to the base frame of the robot.

#### 4.2.2.3 Joint torques due to contact with obstacles

After calculating the compensation torques $\boldsymbol{\tau}_{ws}$ and $\boldsymbol{\tau}_{fs}$, the torques due to contact with the obstacle $\boldsymbol{\tau}_c$ can be calculated:

$$\boldsymbol{\tau}_c = \boldsymbol{\tau}_{ex} - \boldsymbol{\tau}_{fs} - \boldsymbol{\tau}_{ws} \tag{4.28}$$

Finally, the torques vector $\boldsymbol{\tau}_c$ due to external contact forces with an obstacle is used to calculate the motion in the null space, as described in sub-section 4.2.2.4.

#### 4.2.2.4 Contact controller

Considering that a human user hand-guides the EEF precisely using the measurements from FT sensor, the least squares solution is used for calculating the joints angular velocities vector $\dot{\boldsymbol{q}}_{hg}$. This control command generates the required hand-guiding motion for the EEF (with precision).

In the presence of obstacles, and due to contact forces between the structure of the robot and the obstacles, extra torques start to appear at the joints. Those torques, $\boldsymbol{\tau}_c$, are calculated after acquiring measurements from torque sensors at the joints of the robot and FT sensor measurements at the EEF, as described in the previous section. The torques vector $\boldsymbol{\tau}_c$ is then used as an input to the contact controller. The output of the contact controller is the null space angular velocities $\dot{\boldsymbol{q}}_n$:

$$\dot{\boldsymbol{q}}_n = \mathbf{N}\mathbf{K}\boldsymbol{\tau}_c \tag{4.29}$$

Where, $\mathbf{N}$ is the null space projection matrix of the robot, and $\mathbf{K}$ is a diagonal matrix with constant coefficients.

#### 4.2.2.5 Control command

The total command used to control the robot is the sum of (1) the angular velocity vector for hand-guiding $\dot{\boldsymbol{q}}_{hg}$, and (2) the null space angular velocity vector $\dot{\boldsymbol{q}}_n$, which allows the robot to slide on the obstacle:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{hg} + \dot{\boldsymbol{q}}_n \tag{4.30}$$

#### 4.2.2.6 Robot control

Industrial robots are provided with internal servo control loop at the joint position level, for example the KUKA iiwa robot [140]. Consequently, the industrial robot can be controlled by streaming the reference joint angles, which can be calculated by integrating the joints angular velocities of equation (4.30). However, with time, the

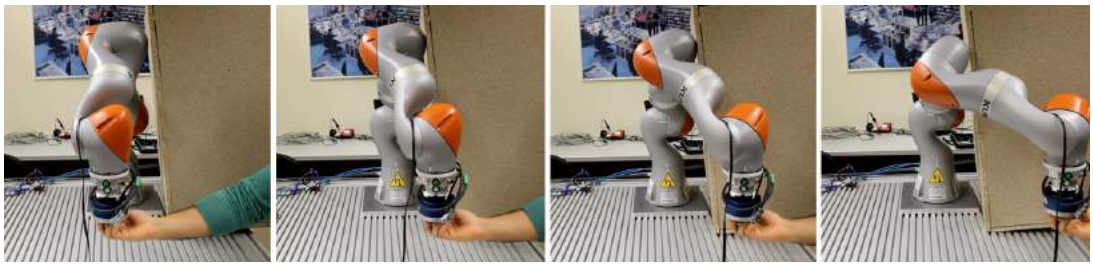

Figure 4.25: Experimental setup. The robot is hand-guided to perform a straight line motion and compliantly avoids the obstacle taking advantage of the redundant axis.

integration causes numerical errors to accumulate which might affect the accuracy in the resulting Cartesian motion. To solve this problem we propose the following novel solution, after calculating the angular velocities in the null space $\dot{\boldsymbol{q}}_n$, the following integration is calculated:

$$\boldsymbol{q}_{ini} = \boldsymbol{q}_f + \int_{t}^{t+\triangle t} \dot{\boldsymbol{q}}_n dt \tag{4.31}$$

Where, $\boldsymbol{q}_f$ is the joints angles feedback from the robot controller and $\triangle t$ is the update time interval of the control loop. Then the result $\boldsymbol{q}_{ini}$ is used as an initial value for the precision hand-guiding algorithm in section 4.1. Which allows the robot to move precisely in the Cartesian space while avoiding the obstacle in the null space.

### 4.2.3 Experiments

The proposed methodology was tested using a KUKA iiwa 7 R800 robot. This is an industrial sensitive collaborative robot with 7 DoF provided with torque sensors integrated into its joints. An external FT sensor, JR3, is attached at the flange of the robot. The robot was controlled from an external computer using KUKA Sunrise Toolbox [11].

Figure 4.25 shows the proposed experimental setup. The robot is hand-guided by applying a force on the FT sensor in the $y$ direction of the robot base frame. In consequence, the tool attached to the robot moves along a straight line in that direction. Meanwhile, during hand-guiding, robot's structure collides with an obstacle, a box in the way of the robot. The robot adjusts its configuration, by utilizing its redundancy, and slides smoothly on the obstacle with its structure. As a result, the robot is able to keep moving on a straight line along the $y$ direction while navigating the obstacle during the contact. In contrast, for traditional hand-guiding solutions at EEF level the robot is blind to its surrounding, such that it keeps pushing with its structure against the obstacle causing joints-torques/motors-currents to increase triggering an emergency stop.

A video segment showing the test is available in [141]. During the experimental test, various data were recorded including (1) robot joints angular positions, Figure 4.26, (2) external torque measurements at robot joints, Figure 4.27, (3) force measurements

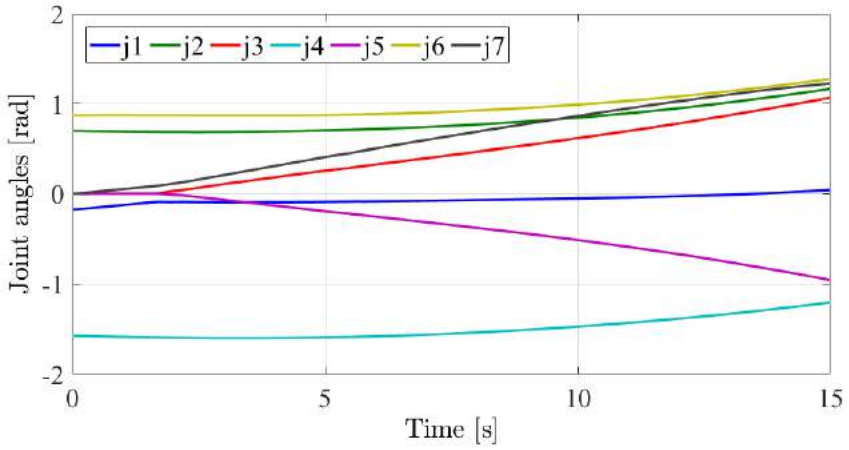


Figure 4.26: Robot joints angular positions.

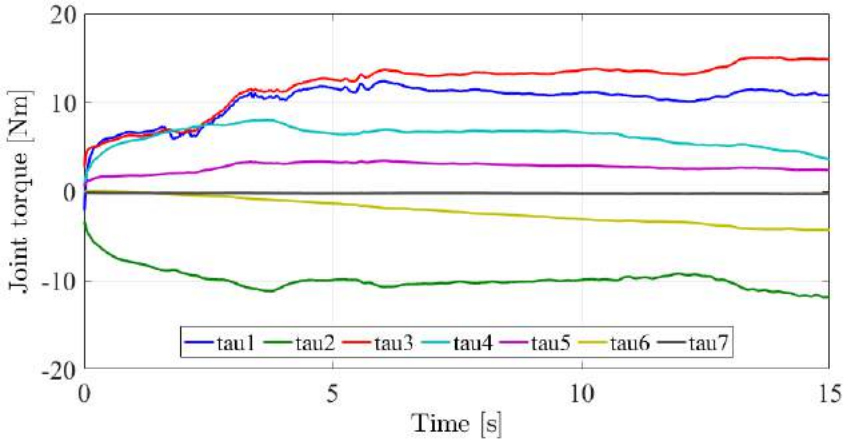


Figure 4.27: External torques measurement at the robot joints.

from the FT sensor, Figure 4.28, (these measurements, represented in robot base frame, are compensated for the weight of the tool), and (4) positional data of the Tool Centre Point (TCP), Figure 4.29.

From Figure 4.28 it is noticed that at the beginning of the test the operator applies a hand-guiding force at EEF, mainly along the $y$ direction of the base frame. As a result the EEF starts moving in the positive $y$ direction, Figure 4.29. From the joints torques graph, Figure 4.27, it is noticed that at around 2 seconds from the beginning of the test, a contact between the robot structure and the obstacle is initiated. After the contact the torques acting on the first and the third joints increase. Due to the proposed null space control, the rate of motion of the first and third joints also increase. This is evident in the joint angles graph, Figure 4.26, where the rate of change of the angular positions of first and third joints increase due to null space motion. This allows the EEF to move as desired by the operator, without conflict due to the presence of the obstacle.

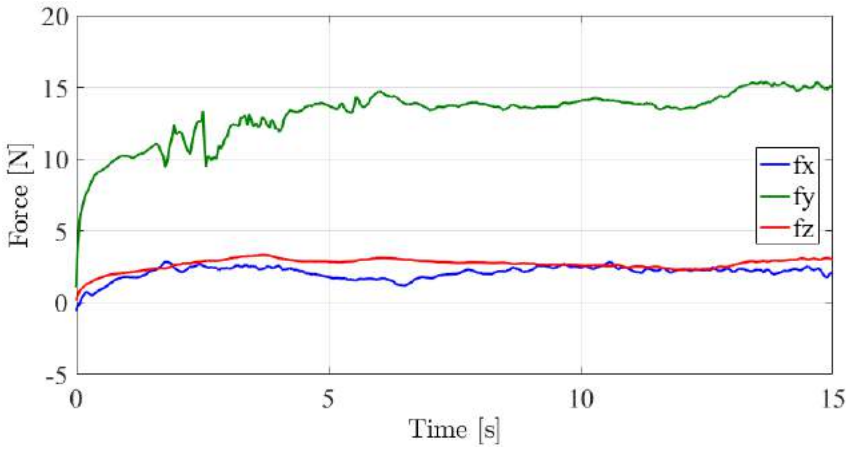


Figure 4.28: Components of hand-guiding force described in base frame of the robot.

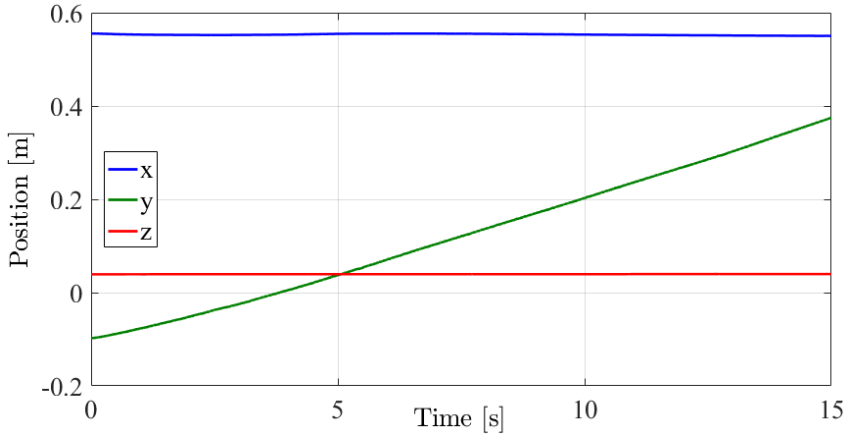


Figure 4.29: Positional data of the EEF.

To show the precision of the motion, the actual path of EEF as recorded from the robot controller is plotted in the xy plane Figure 4.30, and in the zy plane Figure 4.31. It is shown that the actual path deviates from a straight line. From the plots the maximum error of the actual path in the $x$ and $z$ directions are 0.52 and 0.51 mm, respectively. As a comparison, we carried out the same test with KUKA off-the-shelf hand-guiding, joint level controlled, we tried to move the EEF in a line parallel to the $y$ direction, the errors were of order of centimeters. Depending on the user, errors as big as 3 cm and 5 cm of EEF position in the $x$ and the $z$ directions have occurred, an expected result since that hand-guiding the robot at the joints level does not guarantee precision at the EEF level (in best case possible the robot can be as precise as the human operator can be).

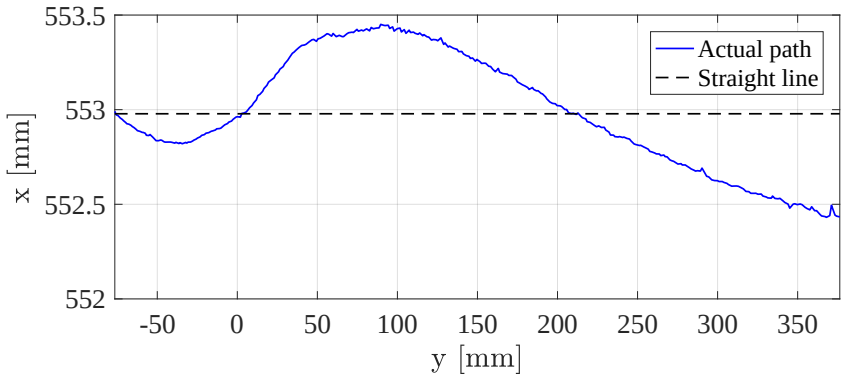


Figure 4.30: Actual path in xy plane.

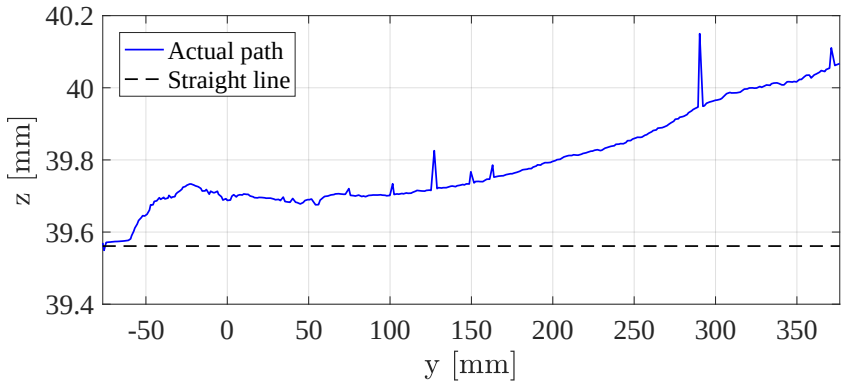


Figure 4.31: Actual path in zy plane.

### 4.2.4 Summary

In this section it is presented a novel method for precision hand-guiding of redundant manipulators with obstacle avoidance capability. Unlike traditional hand-guiding solutions, we take advantage of robot redundancy to avoid in-contact obstacles between the manipulator's structure and the surrounding environment while achieving precision at EEF level. Torque measurements from robot joints are used together with the measurements from an external FT sensor attached at the flange. The acquired sensory data are then treated for calculating two essential quantities: (1) the torques due to contact forces and (2) the hand-guiding force and moment. Using these data a control scheme is proposed such that the manipulator is able to compliantly slide on obstacles during the contact while precisely hand-guided at the EEF level. Tests were carried out successfully on KUKA iiwa robot. From the results it is concluded that the robot successfully manages to perform the desired hand-guiding path with precision while avoiding excessive contact forces with the surrounding environment. As compared to joint level hand-guiding, the proposed method gives superior precision with order of magnitude.

# Chapter 5

# Contribution to Robot Dynamics Formulation

Dynamics of robots is an important topic since that it is highly involved in their design, simulation and control. Owing to its importance this subject had been studied extensively in the past decades. Thus, several algorithms and methods had been developed to calculate robot dynamics [142, 143]. Nevertheless, this subject remains till this day open for extensive research while every year there are new studies being published, methods and algorithms being proposed. This chapter introduces two main contributions into the mathematical formulation of robot dynamics. The first pertains to the efficiency in calculating the joint space inertia (mass) matrix for robots with High Degrees of Freedom (HDoF). The second describes a recursive algorithm for calculating Christoffel symbols efficiently.

## Chapter's breakdown

This chapter of the document is organised into two separate sections:

- Section 5.1: describes an algorithm with a minimal $O(n^2)$ cost[1] for efficiently calculating the joint space inertia matrix for robots with HDoF. This algorithm is obviously important in practice for reducing the execution time required to calculate the mass matrix of HDoF robots. However, it is also important theoretically, since it gives the limit on the second order efficiency possible of calculating the mass matrix for HDoF robots.
- Section 5.2: describes a light weight recursive (non-symbolic) algorithm for calculating Christoffel symbols of robotic manipulators efficiently.

[1] Where $n$ is the number of degrees of freedom of the robot.

## 5.1 Contribution to Mass Matrix Calculation

Increasingly, robots have more Degrees of Freedom[2] (DoF), imposing a need for calculating more complex dynamics. As a result, better efficiency in carrying out dynamics computations is becoming more important. In this section, an efficient method for computing the joint space inertia matrix (JSIM) for serially linked robots with high degrees of freedom is addressed. We call this method the Geometric Dynamics Algorithm for High number of robot Joints (GDAHJ). GDAHJ is non-symbolic, preserves simple formulation, and is convenient for numerical implementation. Results compare favorably with existing methods, achieving better performance over state-of-the-art by Featherstone when applied to robots with more than 13 DoF.

### 5.1.1 Introduction

Robot dynamics can be described by one of two formulations:

1. Operational space formulation. In this formulation the dynamics equations are referenced to the manipulator end-effector. In a pioneering study this approach was described and used to control PUMA600 robot [145]. It is also applied for the combined application of motion and force control [146]. Algorithms for efficient robot dynamics calculations based on operational space formulation are presented in [147] and [148].

2. Joint space formulation. This formulation describes the dynamics of robot in joint space. This formulation manifests the effect of the joints' positions, velocities and accelerations on the torques and vice-versa.

The mathematical formulation of the inverse dynamics in joint space [1, 32, 34, 149] is given by:

$$\boldsymbol{\tau} = \mathbf{A}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g} \tag{5.1}$$

Where $\boldsymbol{\tau}$ is the joints' torques vector, $\boldsymbol{q}$ is the joints' positions vector, $\dot{\boldsymbol{q}}$ is the vector of joints' angular velocities, $\ddot{\boldsymbol{q}}$ is the vector of joints' angular accelerations, $\mathbf{A}(\boldsymbol{q})$ is joint space inertia matrix of the robot, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the joint space Coriolis matrix of the robot, and $\boldsymbol{g}$ is the vector of joints' torques due to gravity. As described in [150], equation (5.1) can be extended to include contact forces, joints elasticity, friction, actuators inertias and dynamics. $\mathbf{A}(\boldsymbol{q})$ is an $n \times n$ matrix, in which $n$ is the number of robot's joints considering that each joint has one degree of freedom, it is symmetric, positive definite and has the property of being a function of only joints' positions. $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is an $n \times n$ matrix, function of joints' positions and velocities, and describes the centrifugal and Coriolis effects on joints' torques.

One of the earliest methods used to deduce the equations of robot dynamics was the one based on Lagrangian formulation. This method is well described in the literature. A methodology for deducing the dynamics of gear-driven serially linked robot by using Lagrangian formulation is described in [95]. This study took into consideration the

[2] For example Caltech's 30 DoF snake like robot presented in [144].

effects of the driving motors. The Lagrangian formulation is widely used as the bases for automatic generation of equations of robot dynamics in symbolic form. Most recent toolboxes for generating equations of robot dynamics using Lagrangian formulation are in [151].

The Lagrangian formulation is a straight forward approach that treats the robot as a whole and utilizes its Lagrangian, a function that describes the energy of the mechanical system:

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{5.2}$$

Where $\mathcal{L}$ is the Lagrangian function, $\mathcal{T}$ is the kinetic energy and $\mathcal{U}$ is the potential energy. The function described previously is formulated in terms of the generalized coordinates $\boldsymbol{q}$, differentiating it gives an expression for the associated generalized forces $\boldsymbol{v}$:

$$\boldsymbol{v} = \frac{d}{dt}(\frac{\delta \mathcal{L}}{\delta \dot{\boldsymbol{q}}})^{\mathrm{T}} - (\frac{\delta \mathcal{L}}{\delta \boldsymbol{q}})^{\mathrm{T}} \tag{5.3}$$

Even though the Lagrangian formulation can be considered as a straight forward approach, the method requires partial differentiation. Despite the fact that symbolic manipulation methods have been utilized to perform the differentiation [152], the method still lacks the efficiency in terms of execution-time. This can be clearly noticed when the robot presents a relatively high number of DoF as noted in [149] and most remarkably in [153], where the author performed comparison of execution-times required to run simulations based on dynamical models derived by Newton-Euler recursive technique and Euler-Lagrange technique. It was reported execution-times difference of order of magnitude which clearly put the case in favor of the Newton-Euler recursion method.

The formulation of robot-specific dynamics using Kane's dynamical equations is in [154]. In this study the authors argue that using Lagrange method to compute dynamics produces huge equations resulting in slow execution and costly computations, while the Recursive Newton-Euler is a generalized method that might perform unnecessary calculations on specific robot. Thus, a faster execution algorithm with less computational-cost could be achieved if robot-specific equations are carefully deduced. The study elaborates in step by step manner the methodology for deriving the dynamics equations of Stanford manipulator starting from Kane's dynamical equations. Nevertheless, the method requires a knowledgeable analyst to take on a pencil and paper in hand and work out the equations of a specific robot. A comprehensive review of Kane's equations and Gibbs-Appell equations is in [155].

A computationally efficient Newton-Euler recursive method is described in [156]. This method is performed in two phases: the first phase (forward propagation) during which the accelerations and velocities of robot links are calculated, and the second phase (backward propagation) where torques and forces are calculated. The method proved to be very efficient for calculating the inverse dynamics. However, the calculations are carried out implicitly such that the inertia matrix cannot be retrieved directly. It is shown in [157] that the inertia matrix $\mathbf{A}(\boldsymbol{q})$ can be calculated from the model of the inverse dynamics by assigning a unit value to one element of the joints' accelerations vector and assigning a zero value to the remaining elements, including

the joints' velocities and the gravity term. In such scenario, the associated column of the inertia matrix is calculated, and by iterating the procedure through all of the elements of the joints' acceleration vector the inertia matrix is achieved. This method was later renamed Composite-Rigid-Body Algorithm (CRBA), by Featherstone [158]. Using CRBA to calculate the inertia matrix proved to be computationally efficient, especially if the calculations are performed in links-attached local frames. Computer code of the algorithm based on 6D or spatial vectors algebra is available in [158]. A comprehensive review of spatial vectors and Plücker basis is in [159] and in chapter 2 of [160].

In this section we propose GDAHJ as a method to calculate JSIM for serially linked bodies with relatively high number of DoF, GDAHJ achieves better efficiency over state-of-the-art method, the famous CRBA. This increase in efficiency is achieved through minimizing the number of operations that have $O(n^2)$ computational complexity. In GDAHJ the number of computations associated with the quadratic terms are reduced to the minimum value possible, from $16n^2$ in the case of CRBA to $5.5n^2$ for GDAHJ.

### 5.1.2 Theory and principles

The proposed algorithm builds on what we call the frame injection principle illustrated in Figure 5.1, where a frame $j$ attached to joint $j$ (according to the modified Denavit Hartenberg convention [32]) transfers to link $k$ a linear acceleration into its center of mass and an inertial moment around its center of mass. In this study we notate them by $\Gamma_{Ckj}$ and $\boldsymbol{\mu}_{Ckj}$, respectively. This transfer is due to the rotational effect of joint $j$ around its axis of rotation, or the $z$ axis of frame $j$. This cause and effect relationship between frame $j$ and link $k$ is referred to by the subscript $kj$ in $\Gamma_{Ckj}$ and $\boldsymbol{\mu}_{Ckj}$, while the subscript C is used to refer to the center of mass of link $k$. The same subscript notation will hold in this chapter for denoting frame-link interaction of cause-and-effect unless stated otherwise.

#### 5.1.2.1 Link's acceleration due to the single-frame effect

Each frame $j$ transfers to link $k$ three acceleration vectors tangential acceleration, normal acceleration and Coriolis acceleration. The first of which is shown in Figure 5.2, it is due to the angular acceleration of frame $j$:

$$\Gamma^{\tau}_{Ckj} = \boldsymbol{\varepsilon}_j \times \boldsymbol{p}_{Ckj} \tag{5.4}$$

Where $\Gamma^{\tau}_{Ckj}$ is the tangential acceleration of the center of mass of link $k$ due to the rotation of frame $j$, the symbol $\times$ is used to denote the cross product and $\boldsymbol{p}_{Ckj}$ is the vector connecting the origin of frame $j$ and the center of mass of link $k$. $\boldsymbol{\varepsilon}_j$ is the angular acceleration of joint $j$:

$$\boldsymbol{\varepsilon}_j = \ddot{q}_j \boldsymbol{k}_j \tag{5.5}$$

where $\boldsymbol{k}_j$ is the unit vector associated with the $z$ axis of joint $j$ and $\ddot{q}_j$ is the angular acceleration of that joint.

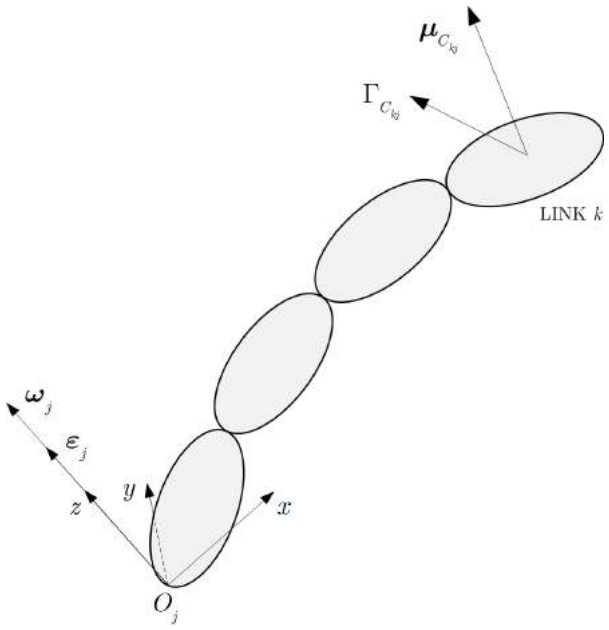


Figure 5.1: Inertial moment $\boldsymbol{\mu}_{Ckj}$ and linear acceleration $\Gamma_{Ckj}$ of center of mass of link $k$ transferred by frame. $j$

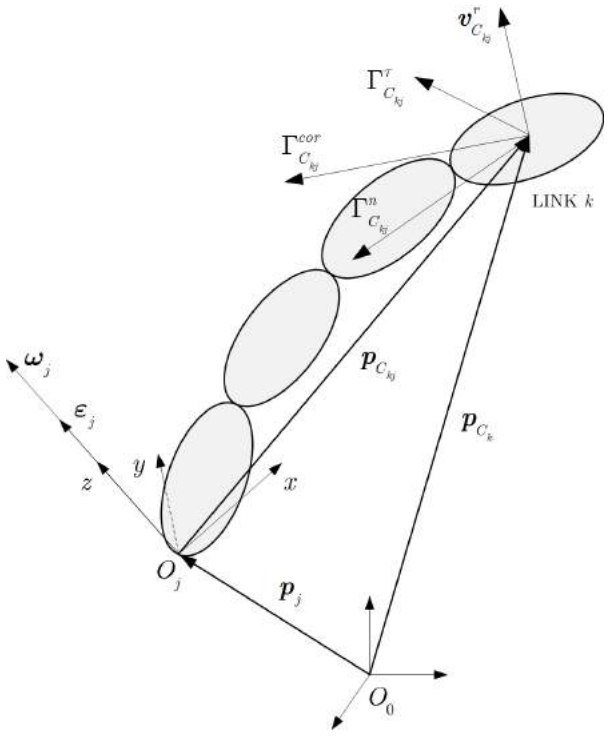


Figure 5.2: Tangential, normal and Coriolis accelerations of center of mass of link $k$ transferred by frame $j$.

Concerning the normal acceleration, each frame $j$ transfers to link $k$ a normal acceleration due to its rotation, Figure 5.2:

$$\Gamma^{n}_{Ckj} = \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \boldsymbol{p}_{Ckj}) \tag{5.6}$$

Where $\boldsymbol{\omega}_j$ is the angular velocity of link $j$ due to the rotational effect of joint $j$. It is given by:

$$\boldsymbol{\omega}_j = \dot{q}_j \boldsymbol{k}_j \tag{5.7}$$

We can rewrite the equation of the normal acceleration transferred to link $k$ due to frame $j$ by:

$$\Gamma^{n}_{Ckj} = \boldsymbol{k}_j \times (\boldsymbol{k}_j \times \boldsymbol{p}_{Ckj})\dot{q}_j^2 \tag{5.8}$$

The third acceleration transferred is Coriolis acceleration, Figure 5.2, in which each frame $j$ transfers to center of mass of link $k$ Coriolis acceleration $\Gamma^{cor}_{Ckj}$:

$$\Gamma^{cor}_{Ckj} = 2\boldsymbol{\omega}_j \times \boldsymbol{v}^{r}_{Ckj} \tag{5.9}$$

Where $\boldsymbol{v}^{r}_{Ckj}$ is the velocity transferred to the center of mass of link $k$ from frames $j+1$ up to frame $k$. The superscript $r$ is used to denote that this is a relative velocity and C to refer to the center of mass of link $k$, so that $\boldsymbol{v}^{r}_{Ckj}$ can be calculated from:

$$\boldsymbol{v}^{r}_{Ckj} = \sum_{i=j+1}^{k} \boldsymbol{\omega}_i \times \boldsymbol{p}_{Cki} \tag{5.10}$$

The total linear acceleration transferred by frame $j$ to the center of mass of link $k$ is given by:

$$\Gamma_{Ckj} = \Gamma^{\tau}_{Ckj} + \Gamma^{n}_{Ckj} + \Gamma^{cor}_{Ckj} \tag{5.11}$$

#### 5.1.2.2 Link's inertial moment due to single-frame effect

Each frame $j$ transfers to link $k$ three inertial moments, the first of which is due to angular acceleration of frame $j$:

$$\boldsymbol{\mu}^{\tau}_{Ckj} = (\mathbf{R}_k \mathbf{I}^{k}_{k} \mathbf{R}^{\mathrm{T}}_{k})\boldsymbol{\varepsilon}_j \tag{5.12}$$

Where $\boldsymbol{\mu}^{\tau}_{Ckj}$ is the moment transferred by frame $j$ into link $k$ due to frame's $j$ angular acceleration, $\mathbf{R}_k$ is the rotation matrix of frame $k$ in relation to base frame and $\mathbf{I}^{k}_{k}$ is $3 \times 3$ inertial tensor of link $k$ around its center of mass represented in frame $k$.

The second inertial moment transferred from frame $j$ to link $k$ is due to centrifugal effect:

$$\boldsymbol{\mu}^{n}_{Ckj} = \frac{1}{2}(\mathbf{L}_k \boldsymbol{\omega}_j) \times \boldsymbol{\omega}_j \tag{5.13}$$

Where $\mathbf{L}_k$ is a $3 \times 3$ matrix that is calculated from:

$$\mathbf{L}_k = \mathbf{R}_k(\mathrm{tr}(\mathbf{I}_k^k)\mathbf{1}_3 - 2\mathbf{I}_k^k)\mathbf{R}_k^{\mathrm{T}} \tag{5.14}$$

The subscript in $\mathbf{L}_k$ is to notate that the matrix calculated pertains to link $k$, $\mathrm{tr}(\mathbf{I}_k^k)$ is the trace of the inertial tensor and $\mathbf{1}_3$ is the identity matrix.

The third inertial moment transferred from frame $j$ to link $k$ is due to Coriolis effect:

$$\boldsymbol{\mu}_{\mathrm{C}kj}^{cor} = (\mathbf{L}_k\boldsymbol{\omega}_j) \times \boldsymbol{\omega}_{kj}^r \tag{5.15}$$

Where $\boldsymbol{\omega}_{kj}^r$ can be calculated from:

$$\boldsymbol{\omega}_{kj}^r = \sum_{i=j+1}^{k} \boldsymbol{\omega}_i \tag{5.16}$$

Thus, the total inertial moment transferred to link $k$ around its center of mass due to the rotational effect of frame $j$ is given by:

$$\boldsymbol{\mu}_{\mathrm{C}kj} = \boldsymbol{\mu}_{\mathrm{C}kj}^{\tau} + \boldsymbol{\mu}_{\mathrm{C}kj}^{n} + \boldsymbol{\mu}_{\mathrm{C}kj}^{cor} \tag{5.17}$$

### 5.1.3 Mass Matrix for High DoF Robot

The GDAHJ algorithm calculates the joint space inertia matrix for robots with high DoF quite efficiently, this increase in efficiency is achieved through minimizing the number of operations that has $O(n^2)$ computational complexity to a minimum, according to our knowledge GDAHJ is the most efficient method for high DoF robots ever proposed till now.

Starting from the basic interpretation of JSIM columns, the mathematical equations of GDAHJ algorithm can be deduced. Where as described in section 3.2 of [142], each column $j$ of the JSIM can be interpreted as: the torques acting on the various joints of the robot, due to the unit acceleration of joint $j$, giving that the angular velocities of all of the joints are equal to zero. In Figure 5.3 we show the free body diagram of one link of the robot, with the inertial moments and inertial forces acting on it.

Following the previous definition of column $j$ of JSIM, we can calculate that column as the following: (1) choose a joint $j$, and (2) write the balance equation of a link $k$ from the robot. By referring to Figure 5.3, the balance equation of link $k$:

$$\boldsymbol{\mu}_{k,j} = \boldsymbol{\mu}_{k+1,j} + (\mathbf{R}_k\mathbf{I}_k^k\mathbf{R}_k^{\mathrm{T}})\boldsymbol{k}_j\ddot{q}_j + m_k\hat{\boldsymbol{p}}_{\mathrm{C}kk}(\ddot{q}_j\boldsymbol{k}_j \times \boldsymbol{p}_{\mathrm{C}kj}) +$$

$$\hat{\boldsymbol{l}}_k \sum_{i=k+1}^{n} m_i(\ddot{q}_j\boldsymbol{k}_j \times \boldsymbol{p}_{\mathrm{C}ij}) \tag{5.18}$$

Where $\boldsymbol{\mu}_{k,j}$ is the total moment acting on joint $k$ due to the acceleration of joint $j$ only. $\boldsymbol{l}_k$ is the vector connecting the origin of frame $k$ to the proceeding frame's origin, the little hat notation above the vector is used to denote the skew symmetric operator associated with that vector. From the definition given in the previous section of column $j$ of the mass matrix, we substitute $\ddot{q}_j$ by its value $\ddot{q}_j = 1$, consequently:

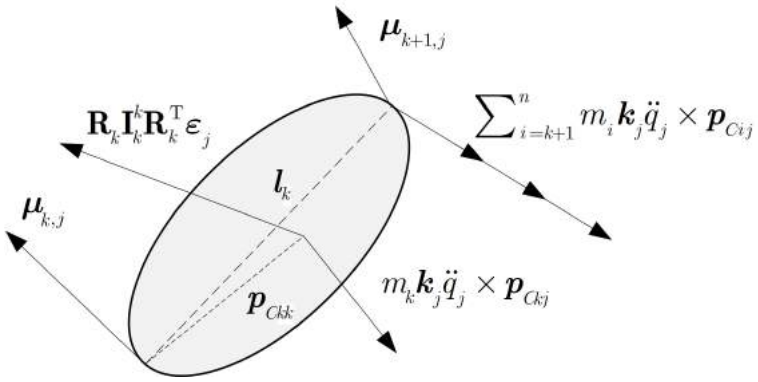


Figure 5.3: Inertial forces and moments acting on link $k$ due to angular acceleration of joint $j$.

$$\boldsymbol{\mu}_{k,j} = \boldsymbol{\mu}_{k+1,j} + (\mathbf{R}_k\mathbf{I}_k^k\mathbf{R}_k^{\mathrm{T}})\boldsymbol{k}_j + m_k\hat{\boldsymbol{p}}_{\mathrm{C}kk}(\boldsymbol{k}_j \times \boldsymbol{p}_{\mathrm{C}kj})$$

$$+ \hat{\boldsymbol{l}}_k \sum_{i=k+1}^{n} m_i(\boldsymbol{k}_j \times \boldsymbol{p}_{\mathrm{C}ij}) \tag{5.19}$$

While:

$$\boldsymbol{p}_{\mathrm{C}kj} = \boldsymbol{p}_{\mathrm{C}k} - \boldsymbol{p}_j \tag{5.20}$$

And:

$$\boldsymbol{p}_{\mathrm{C}ij} = \boldsymbol{p}_{\mathrm{C}i} - \boldsymbol{p}_j \tag{5.21}$$

We substitute the values of $\boldsymbol{p}_{\mathrm{C}kj}$ and $\boldsymbol{p}_{\mathrm{C}ij}$ into (5.19), and we fix:

$$\boldsymbol{\mu}_{kj} = \boldsymbol{\mu}_{k+1,j}$$

$$+ \left(\mathbf{R}_k\mathbf{I}_k^k\mathbf{R}_k^{\mathrm{T}} - m_k\hat{\boldsymbol{p}}_{\mathrm{C}kk}\hat{\boldsymbol{p}}_{\mathrm{C}k} - \hat{\boldsymbol{l}}_k\left(\sum_{i=k+1}^{n} m_i\hat{\boldsymbol{p}}_{\mathrm{C}i}\right)\right)\boldsymbol{k}_j$$

$$- \left(m_k\boldsymbol{p}_{\mathrm{C}kk} + (\sum_{i=k+1}^{n} m_i)\boldsymbol{l}_k\right) \times (\boldsymbol{k}_j \times \boldsymbol{p}_j) \tag{5.22}$$

We define the vector $\boldsymbol{\eta}_k$ by:

$$\boldsymbol{\eta}_k = m_k\boldsymbol{p}_{\mathrm{C}kk} + (\sum_{i=k+1}^{n} m_i)\boldsymbol{l}_k \tag{5.23}$$

And we define the matrix operator $\boldsymbol{\kappa}_k$ by:

$$\boldsymbol{\kappa}_k = -m_k \hat{\boldsymbol{p}}_{\mathrm{C}kk} \hat{\boldsymbol{p}}_{\mathrm{C}k} - \hat{\boldsymbol{l}}_k \left( \sum_{i=k+1}^{n} m_i \hat{\boldsymbol{p}}_{\mathrm{C}i} \right) \tag{5.24}$$

Then we write:

$$\boldsymbol{\mu}_{k,j} = \boldsymbol{\mu}_{k+1,j} + \left( \mathbf{R}_k \mathbf{I}_k^k \mathbf{R}_k^{\mathrm{T}} + \boldsymbol{\kappa}_k \right) \boldsymbol{k}_j - (\boldsymbol{\eta}_k) \times (\boldsymbol{k}_j \times \boldsymbol{p}_j) \tag{5.25}$$

By performing a recursion on previous equation from link $n$ to link $k$, and noticing that $\boldsymbol{\mu}_{n+1,j} = \mathbf{0}$ we get:

$$\boldsymbol{\mu}_{k,j} = \left( \sum_{i=k}^{n} \left( \mathbf{R}_i \mathbf{I}_i^i \mathbf{R}_i^{\mathrm{T}} + \boldsymbol{\kappa}_i \right) \right) \boldsymbol{k}_j - \left( \sum_{i=k}^{n} \boldsymbol{\eta}_i \right) \times (\boldsymbol{k}_j \times \boldsymbol{p}_j) \tag{5.26}$$

To hide the complexity in the previous equation, we denote the terms between parenthesis by:

$$\boldsymbol{b}_k = \left( \sum_{i=k}^{n} \boldsymbol{\eta}_i \right) \tag{5.27}$$

And

$$\mathbf{D}_k = \sum_{i=k}^{n} \left( \mathbf{R}_i \mathbf{I}_i^i \mathbf{R}_i^{\mathrm{T}} + \boldsymbol{\kappa}_i \right) \tag{5.28}$$

Substituting (5.27) and (5.28) in (5.26) yields:

$$\boldsymbol{\mu}_{k,j} = \mathbf{D}_k \boldsymbol{k}_j - \hat{\boldsymbol{b}}_k (\boldsymbol{k}_j \times \boldsymbol{p}_j) \tag{5.29}$$

For calculating the $(k, j)$ entry of JSIM, $\mathbf{A}_{k,j}$, we project $\boldsymbol{\mu}_{k,j}$ on the $z$ axis of joint $k$, or in other words we multiply (5.29) by the unit vector $\boldsymbol{k}_k^{\mathrm{T}}$ :

$$\mathbf{A}_{k,j} = \boldsymbol{k}_k^{\mathrm{T}} \boldsymbol{\mu}_{k,j} = \boldsymbol{k}_k^{\mathrm{T}} \mathbf{D}_k \boldsymbol{k}_j - \boldsymbol{k}_k^{\mathrm{T}} \hat{\boldsymbol{b}}_k (\boldsymbol{k}_j \times \boldsymbol{p}_j) \tag{5.30}$$

By noticing that each entry $k, j$ of the JSIM, or $\boldsymbol{k}_k^{\mathrm{T}} \boldsymbol{\mu}_{k,j}$ is a scalar, then we can transpose the previous equation without loss of generality:

$$\begin{aligned} \mathbf{A}_{k,j} &= \boldsymbol{k}_j^{\mathrm{T}} \left( \mathbf{D}_k^{\mathrm{T}} \boldsymbol{k}_k \right) - \left( \boldsymbol{k}_j \times \boldsymbol{p}_j \right)^{\mathrm{T}} \left( \hat{\boldsymbol{b}}_k^{\mathrm{T}} \boldsymbol{k}_k \right) \\ &= \boldsymbol{k}_j^{\mathrm{T}} \left( \mathbf{D}_k^{\mathrm{T}} \boldsymbol{k}_k \right) + \left( \boldsymbol{k}_j \times \boldsymbol{p}_j \right)^{\mathrm{T}} \left( \hat{\boldsymbol{b}}_k \boldsymbol{k}_k \right) \end{aligned} \tag{5.31}$$

In such a way we have decoupled the dependency between indexes $k$ and $j$. Moreover, we limited the cross-coupling interaction between joint $j$ and bodies $k$ into a minimum. The previous equation states that the effect of acceleration of each joint $j$ is limited to the terms $\boldsymbol{k}_j^{\mathrm{T}}$, and $\boldsymbol{t}_j = \left( \boldsymbol{k}_j \times \boldsymbol{p}_j \right)$. The effect of the articulated bodies from link $n$ to link $k$ is manifested by the terms $\left( \mathbf{D}_k^{\mathrm{T}} \boldsymbol{k}_k \right)$ and $\left( \hat{\boldsymbol{b}}_k^{\mathrm{T}} \boldsymbol{k}_k \right)$. The terms $\boldsymbol{d}_k = \left( \mathbf{D}_k^{\mathrm{T}} \boldsymbol{k}_k \right)$ and $\boldsymbol{y}_k = \left( \hat{\boldsymbol{b}}_k^{\mathrm{T}} \boldsymbol{k}_k \right)$ can be calculated with an $O(n)$ algorithm using

**Algorithm 5.1** Calculating joint space inertia matrix entries, algorithm is based on eq (5.31).

01 :For $k = 1 : n$
02 : For $j = 1 : k$
03 :% calculating $\mathbf{A}_{k,j}$ will require two vector inner products and one
04 :% scalar addition with total cost $(3n^2 + 3n)mul + (2.5n^2 + 2.5n)add$
05 : $\mathbf{A}_{k,j} = \boldsymbol{k}_j^{\mathrm{T}} \boldsymbol{d}_k + \boldsymbol{t}_j^{\mathrm{T}} \boldsymbol{y}_k$
06 : $\mathbf{A}_{j,k} = \mathbf{A}_{k,j}$
07 : End
08 :End

Table 5.1: Operation count for proposed method and other methods, *mul* stands for multiplication and *add* for addition.

| Method | Matrix | Cost | Reference |
|---|---|---|---|
| **GDAHJ** | JSIM | $(3n^2 + 88n - 3)mul + (2.5n^2 + 95.5n - 18)add$ | |
| CRBA | JSIM | $(10n^2 + 22n - 32)mul + (6n^2 + 37n - 43)add$ | [161] and [160] EQ. 10.3 |
| Symbolic-Numeric | JSIM-Coriolis | $(\frac{3}{2}n^3 + \frac{35}{2}n^2 + 9n - 16)mul + (\frac{7}{6}n^3 + \frac{23}{2}n^2 + \frac{64}{3}n - 28)add$ | [162] |

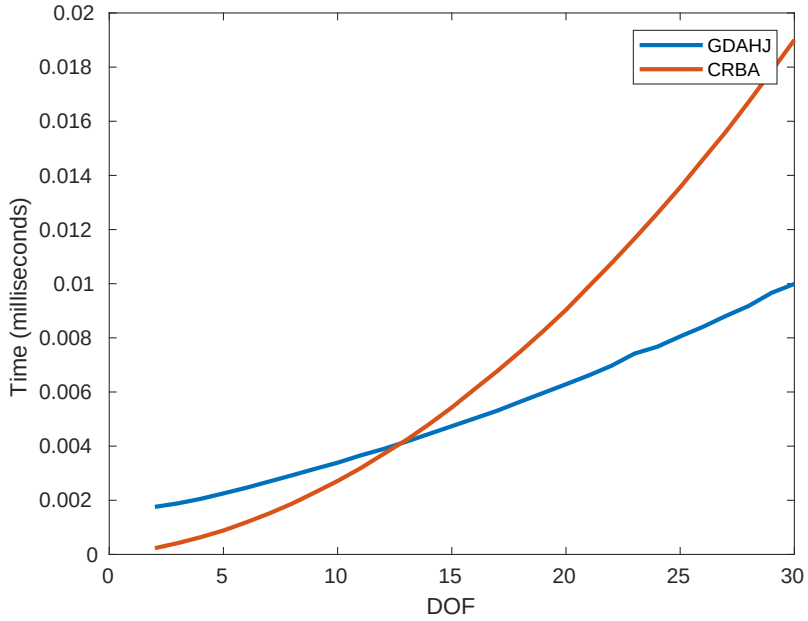


Figure 5.4: Execution time results for GDAHJ vs CRBA.

multiple recursions, while the mass matrix entries can be calculated with minimum quadratic cost using the nested loop in Algorithm 5.1.

The nested loop in Algorithm 5.1 has the minimal quadratic cost. This cost results from two vector-inner products and one scalar addition, with a cost $(3n^2 + 3n)mul + (2.5n^2 + 2.5n)add$, where $mul$ stands for multiplication and $add$ stands for addition. Thus, the $O(n^2)$ computational cost is optimized.

### 5.1.4 Tests

To prove the validity of the proposed method, GDAHJ, and to assess its execution-time performance, a comparison with well established algorithms was performed, namely with CRBA method.

Table 5.1 shows the computational complexity of the proposed algorithm against state-of-the-art algorithm, measured in the number of floating point operations (additions and multiplications) as function of $n$, the number of DoF of the robot. The operation count for CRBA reported in Table 5.1 pertains to the most efficient version of this algorithm [160]. The results reported in Table 5.1 for GDAHJ do not include the number of operations required to perform the forward kinematics of $O(n)$, since that most of robotics applications require forward kinematics calculation, otherwise the cost of the forward kinematics can be added. It can be inferred that GDAHJ performs better than CRBA for articulated bodies (serially connected) that have more than 13 DoF.

To confirm the theoretical results, both algorithms CRBA and GDAHJ were implemented in C++, and a comparison in terms of execution time between the two algorithms was performed, the C++ code with the operation count tables and the

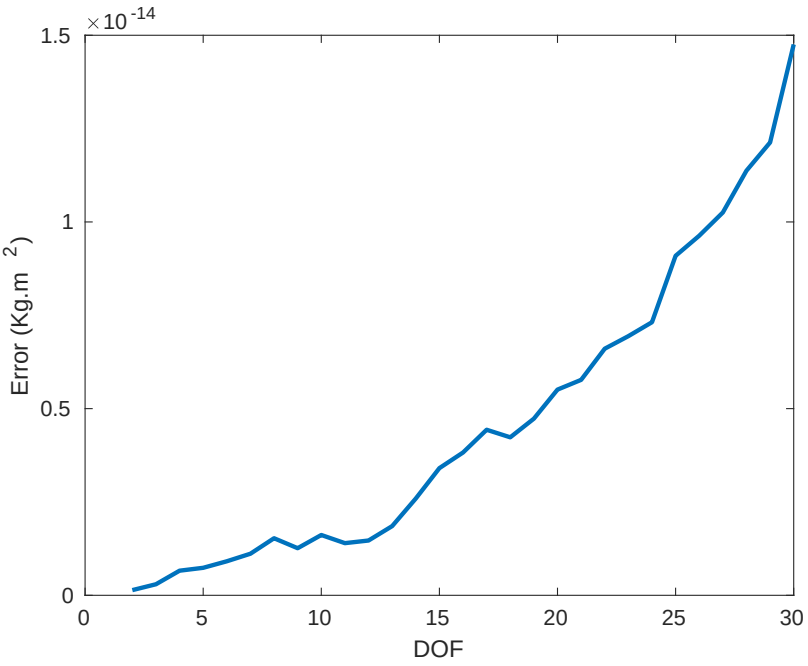


Figure 5.5: Relative error in computation for results achieved using GDAHJ and CRBA.

results are available online in [30]. Numerical tests were carried out by considering a manipulator in which the mass of each link was generated randomly in the range [0,1] $kg$. The inertial tensors of the links were generated as random positive definite matrices in which each element is in the range [0,1] $kg.m^2$. Denavit-Hartenberg parameters of each link were generated randomly as well as the configuration of the robot (joint angles). Afterwards, the JSIM of the manipulator was calculated twice, once using CRBA method and another time using GDAHJ method. Figure 5.4 shows the results in terms of execution time and Figure 5.5 shows the results in terms of numerical error of the calculation between the two methods. From the figures we notice that GDAHJ performs better than CRBA in terms of execution time for high DoF robots (more than 13 DoF).

A metric-value was defined to measure the average error, given by:

$$e = \frac{1}{n^2} \sum_{i,j} \left| \mathbf{A}_{i,j}^{CRBA} - \mathbf{A}_{i,j}^{GAHJ} \right| \tag{5.32}$$

Where $e$ is the resulting average error, $\mathbf{A}_{i,j}^{CRBA}$ and $\mathbf{A}_{i,j}^{GAHJ}$ are the $i, j$ elements of the mass matrix calculated using CRBA and GDAHJ algorithms.

### 5.1.5 Summary

In this section we proposed GDAHJ, a novel algorithm for efficient calculation of JSIM for serially linked robots, the algorithm achieves better efficiency over state-of-the-art when calculating JSIM for hyper-joint manipulators. This increase in efficiency

is achieved through minimizing the number of operations associated with the $O(n^2)$. In such a case, the number of computations associated with the quadratic terms are reduced to the minimum value possible, from $(16n^2)$ in the case of CRBA to $(5.5n^2)$ for the proposed algorithm. At the end comparison between the proposed algorithm against state of the art CRBA was made, the performance of the proposed algorithm was discussed in operation count section of this study. On a theoretical level, this study demonstrates the minimum bound for the $O(n^2)$ operations required for calculating JSIM. Future work will focus on reducing the number of operations associated with $O(n)$ of the algorithm.

## 5.2 Contribution to Christoffel Symbols Calculation

Christoffel symbols of the first kind are very important in robot dynamics. They are used for tuning various proposed robot controllers, for determining the bounds on Coriolis/Centrifugal matrix, for mathematical formulation of optimal trajectory calculation, among others. In the literature of robot dynamics, Christoffel symbols of the first kind are calculated from Lagrangian dynamics using an offline generated symbolic formula. In this study we present a novel and efficient recursive, non-symbolic, method where Christoffel symbols of the first kind are calculated on-the-fly based on the inertial parameters of robot's links and their transformation matrices. The proposed method was analyzed in terms of execution time, computational complexity and numerical error. Results show that the proposed algorithm compares favorably with existing methods.

### 5.2.1 Introduction

Christoffel symbols are important tools in applied sciences, engineering, mathematics and physics. In the latter they appear in rigid body dynamics [95] and general relativity [163]. In the area of robotics, Christoffel symbols of the first kind appear when deducing the equation of robot dynamics using the Lagrangian:

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{5.33}$$

Where $\mathcal{L}$ is the Lagrangian function, $\mathcal{T}$ is the kinetic energy and $\mathcal{U}$ is the potential energy, all described in terms of the generalised coordinates $\boldsymbol{q}$. In such a case, the associated generalised forces $\boldsymbol{\tau}$:

$$\boldsymbol{\tau} = \frac{d}{dt}(\frac{\delta \mathcal{L}}{\delta \dot{\boldsymbol{q}}})^{\mathrm{T}} - (\frac{\delta \mathcal{L}}{\delta \boldsymbol{q}})^{\mathrm{T}} \tag{5.34}$$

Consequently, the canonical form of the inverse dynamics derived using the Lagrangian is:

$$\tau_k = \sum_j a_{kj}\ddot{q}_j + \sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i + g_k \tag{5.35}$$

Where $\tau_k$ is the torque at joint $k$, $a_{kj}$ is the $(k,j)$ element of the mass matrix, $\ddot{q}_j$ is the angular acceleration of joint $j$, $\dot{q}_j$ is the angular velocity of joint $j$, $g_k$ is the torque at joint $k$ due to gravity, and $c^k_{ji}$ represents Christoffel symbols of the first kind (later in the document referred to by Christoffel symbols). From [164] $c^k_{ji}$ is given by:

$$c^k_{ji} = c^k_{ij} = \frac{1}{2}\left(\frac{\partial a_{kj}}{\partial q_i} + \frac{\partial a_{ki}}{\partial q_j} - \frac{\partial a_{ij}}{\partial q_k}\right) \tag{5.36}$$

The Lagrangian formulation can be considered as a straight forward approach. However, the method requires partial differentiation, which makes it unpractical to apply manually for complex systems. Thus, symbolic manipulation methods have been utilized to perform the differentiation by a computer [152]. Nevertheless, the resulting equations generated by a computer lack the efficiency in terms of execution time,

this fact becomes more noticeable for robots with high number of joints or DoF as noted in [149] and most remarkably in [153], where the author performed comparison of execution times required to run dynamics simulations based on models derived by Newton-Euler recursive technique and Euler-Lagrange technique. Where it was reported execution times difference of order of magnitude which put the case in favor of the Newton-Euler recursive methods.

Due to their efficiency, researchers developed various recursive algorithms for calculating the inverse dynamics [156], the forward dynamics [165, 166], the joint space inertia matrix [161, 14]. Nevertheless, we are not aware of any recursive algorithm for calculating Christoffel symbols numerically. As such, in this study we present a method for calculating Christoffel symbols recursively. We also analyze the performance of the proposed algorithm in terms of number of operations, execution time and numerical error. MATLAB code of the proposed algorithm, including implementation examples, are available in the web-page [31].

### 5.2.2 Motivation and Contribution

Calculating Christoffel symbols of the first kind is very important in robot dynamics. Hence, Christoffel symbols have been used for solving various robotics problems. In [167] they are used for calculating the bounds on the Coriolis/Centrifugal matrix. These bounds play an important role for designing and tuning various proposed robot controllers [168, 169, 170, 171, 172, 173]. In addition, Christoffel symbols have been used in a dynamic neurocontroller of robotic arms [174]. They can also be used to calculate a special form of Coriolis matrix that preserves the skew symmetry property [1] (an essential property for various control algorithms). Christoffel symbols are also important for planning time optimal trajectories and optimal velocity-profile generation [175]. In such a case, the geometric path is parametrized using a vector function of a scalar parameter $\theta(t)$. So that the inverse dynamics equation (which enters the optimization as differential constraint) is reformulated to decouple the configuration dependent coefficients from the time dependent parameter $\theta(t)$, as shown in [175]:

$$\boldsymbol{\tau} = \tilde{\boldsymbol{m}}(\boldsymbol{q})\ddot{\theta} + \tilde{\boldsymbol{c}}(\boldsymbol{q})\dot{\theta}^2 + \tilde{\boldsymbol{d}}(\boldsymbol{q}) \tag{5.37}$$

In such a case, Christoffel symbols are utilized for calculating the (configuration dependent) coefficients $\tilde{\boldsymbol{c}}(\boldsymbol{q})$ in each configuration on the discretized geometrical path.

In [176] Christoffel symbols are calculated in symbolic form, based on the Lagrangian formulation of the robot dynamics. This method has become the norm and is presented in standard robotics textbooks [177, 1, 178], including the Handbook of Robotics [179] in page 44, where the deduction of Christoffel symbols is introduced in the Dynamics chapter under the subsection "2.3.2 Lagrange Formulation". Nevertheless, symbolic methods for performing the calculations have major drawbacks, namely:

- The symbolic manipulation of the equations is time consuming, so it has to be performed offline.
- For high DoF, the symbolic equations become very complex resulting in much slower execution times than recursive methods, a fact reported in literature [149].

Apart form that, a recursive method for calculating Christoffel symbols has several advantages over the symbolic:

- Unlike the symbolic methods, recursive methods can be used on-the-fly, and do not require an offline preprocessing. This makes recursive methods essential for calculating the Christoffel symbols on-the-fly in robots that change its kinematic chain and dynamic model, for example by adding or subtracting extra bodies (including Reconfigurable and Self Assembling robots or when attaching objects to the EEF).

- Since that recursive methods are used on-the-fly, the proposed recursive method allows updating the dynamical constants (dynamical model) of the robot on-the-fly. This allows the algorithm to use initial estimates of dynamic constants while learning and tuning them more accurately during operation. This can not be done easily in symbolic methods that require offline code regeneration.

Moreover, the proposed method calculates Christoffel symbols based on the robot's transformation matrices and inertial parameters, without requiring partial differentiation. Apart from the previously listed computational advantages, the proposed method offers more insight into the nature of Christoffel symbols from the point of view of Newton mechanics.

### 5.2.3 Calculating Christoffel symbols

For articulated rigid bodies in a weightless environment (no gravitational field) equation (5.35) becomes:

$$\sum_{j} a_{kj}\ddot{q}_j + \sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i = \tau_k \tag{5.38}$$

We propose a scenario where only joints $i$ and $j$ of the articulated rigid bodies are in motion with constant angular velocities, while the other joints are fixed. Then, the left hand side of equation (5.38) becomes:

$$\sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i = c^k_{jj}\dot{q}^2_j + 2c^k_{ji}\dot{q}_j\dot{q}_i + c^k_{ii}\dot{q}^2_i \tag{5.39}$$

Considering the frame injection principle (described previously in 5.1.2), the right hand side of equation (5.38) is the torque resulting from the sum of three inertial moments:

$$\tau_k = \tau_{k_j} + \tau_{k_{ji}} + \tau_{k_i} \tag{5.40}$$

Where:

- $\tau_{k_j}$ is the torque due to the Centrifugal effect resulting from motion of joint $j$. Thus, it is a function of $\dot{q}^2_j$;

- $\tau_{k_{ji}}$ is the torque due to the Coriolis effect resulting from motion of joints $j$ and $i$. Thus, it is a function of the product $\dot{q}_j\dot{q}_i$;

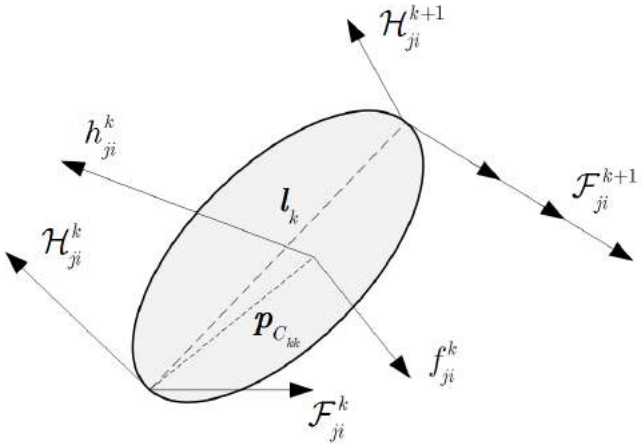


Figure 5.6: Backward recursion on moments and forces.

- $\tau_{k_i}$ is the torque due to the Centrifugal effect resulting from motion of joint $i$. Thus, it is a function of $\dot{q}_i^2$.

From equations (5.39) and (5.40) we find that:

$$\tau_{k_j} = c_{jj}^k \dot{q}_j^2 \tag{5.41}$$

$$\tau_{k_{ji}} = 2c_{ji}^k \dot{q}_j \dot{q}_i \tag{5.42}$$

$$\tau_{k_i} = c_{ii}^k \dot{q}_i^2 \tag{5.43}$$

In such a case, to calculate $c_{jj}^k$, $c_{ji}^k$ and $c_{ii}^k$ we assign a unitary value to the angular velocities $\dot{q}_j$ and $\dot{q}_i$. Then, equations (5.41),(5.42),(5.43) are interpreted as:

- The Christoffel symbol $c_{ji}^k$ is equal to half of the torque $\tau_{k_{ji}}$ which acts on joint $k$ due to Coriolis effect resulting from the unit angular velocities at joints $j$ and $i$;
- The Christoffel symbol $c_{jj}^k$ is equal to the torque $\tau_{k_j}$ which acts on joint $k$ due to Centrifugal effect resulting from the unit angular velocity at joint $j$;
- The same applies for Christoffel symbol $c_{ii}^k$ which results from $c_{jj}^k$ after a change of index.

To calculate the Christoffel symbols $c_{ji}^k$ we apply backward recursion on the forces and moments shown in Figure 5.6, where:

- $\boldsymbol{h}_{ji}^k$ is half of the inertial moment $\boldsymbol{\mu}_{Ckj}^{cor}$ at the center of mass of link $k$. It is due to Coriolis effect resulting from a unit angular velocity at joints $j$ and $i$. From equation (5.15) of the frame injection principle, $\boldsymbol{h}_{ji}^k$ is given by:

$$\boldsymbol{h}_{ji}^k = \frac{1}{2}\boldsymbol{\mu}_{Ckj}^{cor} = \frac{1}{2}(\mathbf{L}_k \boldsymbol{k}_j) \times \boldsymbol{k}_i \tag{5.44}$$

Table 5.2: Comparison for calculating Christoffel symbols of a 5 DoF serially linked robot using different methods

| Criteria | Lagrangian (Optimized) | Lagrangian (Not optimized) | Proposed method |
|---|---|---|---|
| Size of generated file (bytes) | 778 732 | 67 873 609 | 4 497 |
| Off-line time for function generation | 8 days | 897 sec | - |
| On-line execution time (seconds) | 4.6e-04 | 96 | 9.9e-5 |

- $\boldsymbol{f}_{ji}^{k}$ is half of the inertial force at the center of mass of link $k$ due to $\Gamma_{Ckj}^{cor}$. It is due to Coriolis effect resulting from a unit angular velocity at joints $j$ and $i$. From equation (5.9) of the frame injection principle, $\boldsymbol{f}_{ji}^{k}$ is given by:

$$\boldsymbol{f}_{ji}^{k} = \frac{1}{2} m_k \Gamma_{Ckj}^{cor} = m_k \boldsymbol{k}_j \times (\boldsymbol{k}_i \times \boldsymbol{p}_{Cki}) \tag{5.45}$$

We calculate the Christoffel symbols $c_{ji}^{k}$ recursively, by applying a backward recursion on Figure 5.6 for the inertial forces $\boldsymbol{f}_{ji}^{k}$ and the inertial moments $\boldsymbol{h}_{ji}^{k}$:

$$\boldsymbol{\mathcal{F}}_{ji}^{k} = \boldsymbol{\mathcal{F}}_{ji}^{k+1} + \boldsymbol{f}_{ji}^{k} \tag{5.46}$$

$$\boldsymbol{\mathcal{H}}_{ji}^{k} = \boldsymbol{\mathcal{H}}_{ji}^{k+1} + \boldsymbol{h}_{ji}^{k} + \boldsymbol{p}_{Ckk} \times \boldsymbol{f}_{ji}^{k} + \boldsymbol{l}_k \times \boldsymbol{\mathcal{F}}_{ji}^{k+1} \tag{5.47}$$

$$c_{ji}^{k} = \boldsymbol{k}_k^{\mathrm{T}} \boldsymbol{\mathcal{H}}_{ji}^{k} \tag{5.48}$$

Where $\boldsymbol{\mathcal{F}}_{ji}^{k}$ is half of the inertial force calculated recursively at joint $k$ due to the unit angular velocity at joints $j$ and $i$. $\boldsymbol{\mathcal{H}}_{ji}^{k}$ is half of the inertial moment calculated recursively at joint $k$ due to the unit angular velocity at joints $j$ and $i$, and the superscript T in $\boldsymbol{k}_k^{\mathrm{T}}$ is to denote the transpose. By applying a similar approach on normal accelerations we can calculate $c_{ii}^{k}$ ($c_{jj}^{k}$). It is noticed that the resulting equations for calculating $c_{ii}^{k}$ and $c_{jj}^{k}$ are exactly similar to the ones in the presented algorithm (for calculating $c_{ji}^{k}$) only with indices changed.

### 5.2.4 Tests

To prove the validity of the proposed method for calculating Christoffel symbols and to assess its performance, a comparison with symbolic Lagrangian based method was performed. The code is provided in the web-page [31]. The robot used to run the test is a 5 DoF serially linked robot, its structure is described in the file `robotStructure_5DOF.mat`. This robot is generated using the file `generateRandomRobot.m` in which the mass of each link was generated randomly in the range [0,1] kg. The inertial tensor of each link was generated as random positive definite matrix in which each element of the matrix is in the range [0,1] kg m$^2$. Denavit-Hartenberg (DH) parameters of each link were also generated randomly. Afterwards, using MATLAB, Christoffel symbols of the robot were calculated using:

Table 5.3: Computational complexity of the proposed method

| Additions | Multiplications |
|---|---|
| $12n^3 + 19n^2 + 40n - 1$ | $\frac{21}{2}n^3 + \frac{45}{2}n^2 + 49n$ |

1. The proposed algorithm, implemented in the file `christoffelNumerically.m` (MATALB function).
2. An offline generated MATLAB function which contains the symbolic equations generated using Lagrangian method, `chri_symbGen5DOF.m`. In this case, the optimization option of the code generator was set to true, as to optimize the generated symbolic equations.
3. Using an offline generated MATLAB function which contains the symbolic equations generated using Lagrangian method `chri_symbGen5DOFnoOpt.m`. In this case, the optimization option of the symbolic equation generator was set to false.

The Christoffel symbols of the manipulator were calculated twice, once using symbolic function and another using the proposed method. Table 5.2 shows a comparison of achieved results. The proposed recursive method is superior in various aspects, including in terms of execution time (4.6X times faster for a 5 DoF robot). The tests were carried out on a personal computer with Intel(R) Core(TM) i7-6850K CPU @ 3.6 GHz, under Windows 10, running MATLAB 2018a. For a 6 DoF robot, the script has been running for two months without finishing the symbolic equations generation (the automatic optimization of the generated equations is extremely time consuming). On the other hand, using the proposed algorithm to calculate Christoffel symbols for 6 DoF robot requires 13.3e-5 seconds, the file with the test is `a01_timeExecution6DOF.m` found in [31].

Table 5.3 shows a summary of the computational complexity of the proposed algorithm measured in the number of floating point operations (additions and multiplications) as function of $n$, the number of DoF of the robot. A detailed breakdown of the computational complexity is found in the file `Operation_Count.ods` found in [31].

Finally, to measure the numerical accuracy of the calculations, the following metric-value was defined:

$$e = \frac{2}{n^3} \sum_{i,j,k} \left| \frac{c_{ji}^k - \hat{c}_{ji}^k}{c_{ji}^k + \hat{c}_{ji}^k} \right| \quad for\, each, c_{ji}^k \neq 0 \tag{5.49}$$

Where $e$ is the relative error, $c_{ji}^k$ is Christoffel symbol calculated using the proposed method and $\hat{c}_{ji}^k$ is Christoffel symbol calculated using the symbolic method. From various calculations using randomly generated configurations, the maximum (worst) $e$ value achieved is $2.196e-14$, indicating that the error is so small mainly due to numerical rounding errors.

### 5.2.5 Application Example

An important application for the proposed algorithm is in minimum-time trajectory optimization for industrial manipulators working in flexible manufacturing. In such

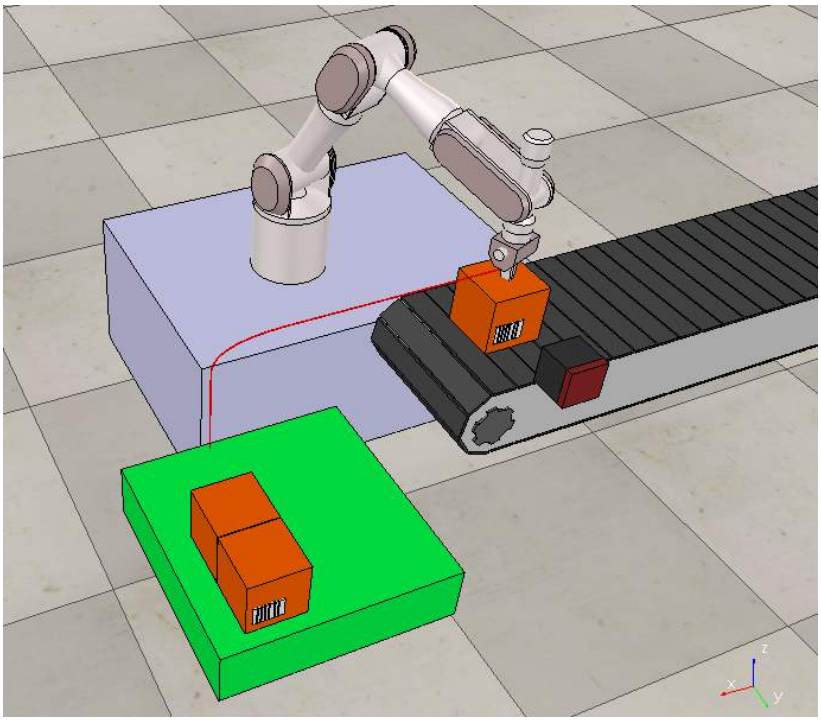

Figure 5.7: Minimum-time trajectory optimization for industrial manipulator performing palletization in flexible-manufacturing scenario.

a case, calculating Christoffel symbols efficiently and on-the-fly is of importance. Because, they enter into the formulation of the minimum-time optimization problem based on robot dynamics as shown in [180, 181], where the elements of vector $\tilde{\boldsymbol{c}}\,(\boldsymbol{q})$ in equation (5.37) are calculated based on the mass matrix and Christoffel symbols:

$$\tilde{c}_k = \sum_j a_{kj} q''_j + \sum c^k_{ij} q'_i q'_j \tag{5.50}$$

Where $\tilde{c}_k$ is the $k^{th}$ element of the vector $\tilde{\boldsymbol{c}}\,(\boldsymbol{q})$ , $q'_j$ is calculated from:

$$q'_j = \frac{dq_j}{d\theta} \tag{5.51}$$

And $q''_j$ is calculated from:

$$q''_j = \frac{d^2 q_j}{d\theta^2} \tag{5.52}$$

A representative scenario is shown in Figure 5.7, where a 7 DoF industrial robot is used to palletize objects in a flexible-manufacturing production line. Optimizing the robot motion for achieving minimum-time trajectory is very important for achieving

high productivity. However, due to the flexible-manufacturing requirements, the robot is required to manipulate various types of objects, with different inertial data (inertial tensor, mass, center of mass). Consequently, the inertial data of the object has to be taken into consideration for optimizing the robot motion while moving the object. This can be done by considering the last link of the robot and the object as a one body when the robot is manipulating it on the planned path.

In such a case, a laser sensor is used to read the bar-code sticker (on the object) which includes the inertial data of the object. When a new object (with different inertial data) is present, the control algorithm calculates the equivalent inertial data of the last link coupled with the object. Considering both, the inertia of the last link of the robot and the object. Afterwards, the optimization problem is invoked, where Christoffel symbols are calculated efficiently and on-the-fly using our algorithm. In comparison, deducing the equations of Christoffel symbols by symbolic manipulation creates a bottleneck in the problem formulation, where generating the symbolic equations is (extremely) time consuming (requires an offline generation phase which is eliminated by our recursive algorithm). This limits the applicability of the traditional method for calculating Christoffel symbols for variable inertias in time critical operations. After formulating the problem, the time required to perform the optimization is of order of seconds [181].

This application example shows the importance of our algorithm for performing Christoffel symbols calculation in a practical application, where our algorithm offers two fundamental advantages:

1. Christoffel symbols are calculated on-the-fly (without requiring an extremely time consuming offline phase), even when the inertial data are changing;

2. Christoffel symbols are calculated more efficiently using our recursive algorithm than using the traditional method (symbolic equation generation).

### 5.2.6 Summary

In this section we proposed recursive algorithm for calculating Christoffel symbols efficiently for serially linked robots. The algorithm achieves better efficiency over Lagrangian based symbolic method. This increase in efficiency is achieved by performing backward recursion on forces and moments. As compared to symbolic method, computational testing proves that the proposed algorithm is (1) efficient (faster execution time), (2) precise (negligible numerical error), and most importantly (3) it does not require a time consuming off-line code generation phase.

# Chapter 6

# Conclusion and Future Work

In this thesis three main topics related to collaborative robotic manipulators were discussed, the collision avoidance of collaborative robots, the hand-guiding of sensitive manipulators in addition to the dynamics of serially linked robots.

The subject of collision avoidance for robotic manipulators has been present in the robotics literature for quite some time. However, it can be noticed that the majority of reported studies are implemented in simulation, on the other hand only a few applied collision avoidance in a real setup (mostly using experimental robot and Kinect vision based sensor, without industrial application context). Thus, in our proposed collision avoidance topic we focused on its implementation in real industrial applications (using collaborative industrial robot KUKA iiwa, and industrial sensors). A laser scanner and IMU sensors are used to capture the configuration of the human coworker around the robot (both are approximated by capsules). An efficient algorithm based on QR factorization is used to calculate the minimum distance between the capsules efficiently. This minimum distance is used as a collision imminence measure. A repulsion vector based on the minimum distance and the relative velocity between the human and the robot is used to repel the robot away from collision. However, unlike other algorithms in robotics literature, we noticed that for industrial applications the presence of a predefined offline generated path is important, and the robot shall be able to switch between different tasks smoothly while moving on the path. Thus, a traditional implementation of the collision avoidance motion without further modification might cause discontinuity in the collision avoidance motion, especially when switching between the tasks that does not allow collision avoidance motion (active assembly) to tasks that allow collision avoidance motion (free motion in the work space) while the human is near the robot. To solve this problem we proposed a solution to guarantee the continuity of the motion. Finally, the proposed algorithm is tested successfully on an industrial robotic cell in automotive industry (car door assembly task).

Going back to the published studies (in the robotics literature) on manipulator's collision avoidance, it can be noticed that most potential fields based algorithms define the attraction and the repulsion vectors as the gradients of the attraction and repul-

sion potential fields. However, moving along the gradient to minimize the potential function is not the best option and suffers drawbacks (known fact in the optimization literature where such implementation is termed as the greedy algorithm), for example at some critical configurations it suffers from intermittent oscillations. This issue is also known in mobile robots literature, where a solution was proposed by applying the Newton method on the potential fields. This solution is inspired from the optimization literature, where the Newton method is preferred over the gradient when the Hessian is known. After a long research into the subject, we could not find any study describing the application of the Newton method on PF based collision avoidance for robotic manipulators. We believe this comes from the fact that for mobile robots the potential field is an explicit function of the configuration space variables, consequently the Hessian is easy to deduce. On the other hand, for robotic manipulators it is not obvious how to calculate its Hessian matrix. In this thesis, a symbolic formula for fast calculation of the Hessian matrix of robotic manipulators is derived, also a systematic method for applying the Newton method to potential fields collision avoidance of robotic manipulators is presented. Where, the Newton method is applied for calculating the direction of the collision avoidance motion in the joint space of the robot. Simulation tests using hyper-redundant planner manipulator proved the effectiveness of the proposed method.

On the subject of hand-guiding collaborative robotic manipulators we presented the precision hand-guiding method, an intuitive alternative to the teach pendant for performing precise positioning operations. It also allows to use the robot as an assistive third hand, which is appealing for tedious assembly tasks. Consequently, a robot can be hand-guided to lift and hold parts in place, while giving the human coworker an opportunity to apply the fixtures (perform the assembly task). Such functionality reduces the risk to workers and properties (falling components), provides precision, allows lifting heavier parts, and increases productivity by keeping less workers occupied in manual tasks on the factory floor. By attaching a force/torque sensor at the flange of the sensitive redundant manipulator, we demonstrated a method that allows to precisely hand-guide the end-effector in the Cartesian space while navigating around surrounding obstacles (upon a contact). Tests were carried out successfully using KUKA iiwa collaborative robot.

On robot dynamics formulation, the thesis presented two recursive algorithms with high computational efficiency. The first is an algorithm with a minimal second order cost for calculating the joint space inertia matrix of serially linked robots. This is important for achieving the lowest computational cost when calculating joint space inertia matrix of hyper-redundant manipulators where the second order term is associated with the highest computational cost. The second algorithm is a recursive non-symbolic method for calculating Christoffel symbols of the robotic manipulator. This method is more efficient (in order of magnitude) than the traditional symbolic method (based on Lagrangian formulation). Various tests were carried out for comparing the performance of the proposed methods against state of the art methods. Results (in terms of execution time and computational complexity analysis) indicate the superior efficiency of the proposed methods.

Future work will be focused in various aspects. For collision avoidance, the proposed method can be extended for mobile manipulators through the implementation of

the potential fields method on both the mobile platform and the robotic manipulator. In addition, it is noticed that in the proposed collision avoidance method the subject of redundancy was addressed implicitly by using the pseudo inverse of the Jacobian of the redundant manipulator when calculating the angular velocities for both the attraction and repulsion vectors. On the other hand, addressing redundancy explicitly (by performing null space projection) offers extra advantages, which for some situations allow collision avoidance motion while performing the required task successfully. For the precision hand-guiding method, we noticed that when the redundant axis is not fixed, a repetitive hand-guiding motion at the EEF does not result in a repetitive motion in the joint space of the robot. This causes a drift in the robot configuration (mainly the redundant axis). While it is easily corrected by readjusting the redundant axis in the null space when required, this solution is not convenient. Therefore, future work will focus on solving this issue. On the subject of robot dynamics, future work will be focused on improving the efficiency of the proposed method for calculating Christoffel symbols and on reducing the $O(n)$ operations of GDAHJ method for calculating the mass matrix.

Appendices

# Appendix A

### Hessian & Gradient in Joints Space

In this section, the formulas for the first and second order derivatives, gradient Equation (3.39) and Hessian Equation (3.38), of the potential field with respect to manipulator's joint angles are deduced. Let $u_k(\boldsymbol{x})$ be a potential field, this potential field is described in Cartesian space, it could be an attraction potential, that attracts the EEF to the goal, or a repulsion potential that repels the robot away from obstacles. Then the second order approximation of the potential field $u_k(\boldsymbol{x}_0)$ near the current position $\boldsymbol{x}_0$ is given by:

$$u_k(\boldsymbol{x}) \approx u_k(\boldsymbol{x}_0) + \triangle \boldsymbol{x}^{\mathrm{T}} \nabla u_k + \triangle \boldsymbol{x}^{\mathrm{T}} \nabla^2 u_k \triangle \boldsymbol{x} \tag{6.1}$$

Where:

- $u_k(\boldsymbol{x})$ is the approximate value of the potential function at the Cartesian position $\boldsymbol{x}$;
- $u_k(\boldsymbol{x}_0)$ is the value of the potential function at the Cartesian position $\boldsymbol{x}_0$;
- $\triangle \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_0$;
- $\nabla u_k$ is the gradient of the potential function $u_k$ with respect to $\boldsymbol{x}$ taken at the point $\boldsymbol{x}_0$;
- $\nabla^2 u_k$ is the Hessian of the potential function $u_k$ with respect to $\boldsymbol{x}$ taken at the point $\boldsymbol{x}_0$.

In addition, differential kinematics gives the following relationship between elemental displacements in Cartesian space and elemental displacements in joint space:

$$\triangle \boldsymbol{x} = \mathbf{J} \triangle \boldsymbol{q} \tag{6.2}$$

Where $\mathbf{J}$ is the Jacobian associated with the point of the manipulator $\boldsymbol{x}_0$, and $\triangle \boldsymbol{q} = \boldsymbol{q} - \boldsymbol{q}_0$. By substituting (6.2) in (6.1) and fixing:

$$u_k(\boldsymbol{x}) \approx u_k(\boldsymbol{x}_0) + \triangle \boldsymbol{q}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \nabla u_k + \triangle \boldsymbol{q}^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \nabla^2 u_k \mathbf{J} \triangle \boldsymbol{q} \tag{6.3}$$

From the previous equation, the joint space gradient $\boldsymbol{g}_k$ and the joint space Hessian $\mathbf{H}_k$ of an individual potential function $u_k$ are deduced. Where the gradient is associated with the first order term of the approximation:

$$\boldsymbol{g}_k = \mathbf{J}^{\mathrm{T}} \nabla u_k \tag{6.4}$$

The Hessian is associated with the second order term of the approximation:

$$\mathbf{H}_k = \mathbf{J}^{\mathrm{T}} \nabla^2 u_k \mathbf{J} \tag{6.5}$$

Finally, in case of a manipulator subjected to several potential fields simultaneously, the total potential field is the sum of all of the potentials, and the total gradient is the sum of the individual gradients:

$$\boldsymbol{g} = \Sigma \boldsymbol{g}_k \tag{6.6}$$

The total Hessian is the sum of the individual Hessians:

$$\mathbf{H} = \Sigma \mathbf{H}_k \tag{6.7}$$

# Appendix B

## List of Methods for KST

Table 6.1: List of KST methods for networking.

| Method | Description |
|---|---|
| net_establishConnection | Connect to KUKA Sunrise.OS |
| net_turnOffServer | Terminate connection to KUKA Sunrise.OS |
| net_updateDelay | Establish a plot of communication delay between computer and controller |
| net_pingIIWA | Ping the robot controller |

Table 6.2: List of KST methods for general purpose functionalities.

| Method | Description |
|---|---|
| gen_DirectKinematics | Calculates the forward kinematics/Jacobian of the manipulator |
| gen_partialJacobean | Calculates the partial Jacobian |
| gen_InverseKinematics | Calculates the inverse kinematics |
| gen_MassMatrix | Calculates the mass matrix |
| gen_CoriolisMatrix | Calculates Coriolis matrix |
| gen_CentrifugalMatrix | Calculates centrifugal matrix |
| gen_GravityVector | Calculates the torque vector acting on the joints due to robot's own weight |
| gen_InverseDynamics | Calculates the inverse dynamics |
| gen_DirectDynamics | Calculates the Forward dynamics |
| gen_NullSpaceMatrix | Calculates the null space matrix of the robot |

Table 6.3: List of KST methods for soft real-time control.

| Method | Description |
|---|---|
| realTime_moveOnPathInJointSpace | Moves the robot continuously in joint space |
| realTime_startDirectServoJoints | Start the DirectServo for controlling the robot in joint space, position mode |
| realTime_stopDirectServoJoints | Stop the DirectServo for controlling the robot in joint space, position mode |
| realTime_startImpedanceJoints | Start impedance control |
| realTime_stopImpedanceJoints | Stop impedance control |
| sendJointsPositions | Updates joints' destination positions and waits for an acknowledgment from the server |
| sendJointsPositionsExTorque | Updates the joints' destination positions while returns back the external torques |
| sendJointsPositionsMTorque | Updates the joints' destination positions while returns back the measured torques |
| sendJointsPositionsGetActualJpos | Updates the joints' destination positions while returns back the actual joint position measurements from encoders |
| realTime_startDirectServoCartesian | Start the DirectServo for controlling the robot in Cartesian space |
| realTime_stopDirectServoCartesian | Stop the DirectServo for controlling the robot in Cartesian space |
| sendEEfPosition | Updates the end-effector destination position and waits for an acknowledgment from the server |
| sendEEfPositionExTorque | Updates the end-effector destination position while returns back the external torques |
| sendEEfPositionMTorque | Updates the end-effector destination position while returns back the measured torques |
| realTime_startVelControlJoints | Start joints' velocity control mode |
| realTime_stopVelControlJoints | Stop joints' velocity control mode |
| sendJointsVelocities | Send reference velocities for robot joints |
| sendJointsVelocitiesExTorques | Updates the joints' destination velocities while returns back the external torques |
| sendJointsVelocitiesMTorques | Updates the joints' destination velocities while returns back the measured torques |
| sendJointsVelocitiesGetActualJpos | Updates the joints' destination velocities while returns back the actual joint position measurements from encoders |
| sendJointsVelocitiesGetActualEEfPos | Updates the joints' destination velocities while returns back the actual end-effector position |

Table 6.4: List of KST methods for point-to-point motion.

| Method | Description |
|---|---|
| movePTPJointSpace | Moves from the current configuration to a new configuration in joint space |
| movePTPLineEEF | Moves the end-effector in a straight line from the current pose to a new pose |
| movePTPHomeJointSpace | Moves the robot to the home configuration |
| movePTPTransportPositionJoi... | Moves the robot to the transportation configuration |
| movePTPLineEefRelBase | Moves the end-effector in a straight line path relative to base frame |
| movePTPLineEefRelEef | Moves the end-effector in a straight line path relative to end-effector initial frame |
| movePTPCirc1OrientationInter... | Moves the end-effector in an arc specified by two frames |
| movePTPArc_AC | Moves the end-effector in an arc specified by center, normal, arc's radius and angle |
| To move the end-effector in an arc in the XY, XZ, YZ plane (movePTPArcXY_AC, movePTPArcXZ_AC, movePTPArcYZ_AC) | |
| To move the end-effector in an ellipse in the XY, XZ, YZ plane (movePTPEllipseXY, movePTPEllipseXZ, movePTPEllipseYZ) | |
| Non-blocking methods for PTP motion (nonBlocking_isGoalReached, nonBlocking_movePTPArcXY_AC, nonBlocking_movePTPArcXZ_AC, nonBlocking_movePTPArcYZ_AC, nonBlocking_movePTPArc_AC, nonBlocking_movePTPCirc1OrintationInter, nonBlocking_movePTPHomeJointSpace, nonBlocking_movePTPJointSpace, nonBlocking_movePTPLineEEF, nonBlocking_movePTPTransportPositionJointSpace) | |
| Interrupt PTP motion according to torque threshold (movePTP_ConditionalTorque_ArcXY_AC, movePTP_ConditionalTorque_ArcXZ_AC, movePTP_ConditionalTorque_ArcYZ_AC, movePTP_ConditionalTorque_Circ1OrintationInter, movePTP_ConditionalTorque_HomeJointSpace, movePTP_ConditionalTorque_JointSpace, movePTP_ConditionalTorque_LineEEF, movePTP_ConditionalTorque_LineEefRelBase, movePTP_ConditionalTorque_TransportPositionJointSpace) | |

Table 6.5: List of KST methods, setters and getters.

| Method | Description |
|---|---|
| setBlueOff | Turns off the blue LED of the pneumatic flange |
| setBlueOn | Turns on the blue LED of the pneumatic flange |
| setPin1Off | Sets the output of Pin1 to low level |
| setPin1On | Sets the output of Pin1 to high level |
| To change the state of Pin2, Pin11 and Pin12 (setPin2Off, setPin2On, setPin11Off, setPin11On, setPin12Off, setPin12On) | |
| getEEF_Force | Returns the measured force at the end-effector reference frame |
| getEEF_Moment | Returns the measured moments at the end-effector reference frame |
| getEEFCartesianOrientation | Returns the orientation (X-Y-Z fixed rotations angles) in radians |
| getEEFCartesianPosition | Returns the position of the end-effector relative to robot base reference frame |
| getEEFPos | Returns the position and orientation of the end-effector relative to robot base reference frame |
| getJointsExternalTorques | Returns the robot joint torques due to external forces |
| getJointsMeasuredTorques | Returns the robot joint torques measured by the torque sensors |
| getJointsPos | Returns the robot joint angles in radians |
| getMeasuredTorqueAtJoint | Returns the measured torque at a specific joint |
| getExternalTorqueAtJoint | Returns the torque at a specific joint due to external forces |
| getEEFOrientationR | Returns the orientation of the end-effector as a rotation matrix |
| getEEFOrientationQuat | Returns the orientation of the end-effector as a quaternion |
| getPin3State | Returns the state of Pin3 |
| Returns the state of Pin4, Pin10, Pin13 and Pin16 (getPin4State, getPin10State, getPin13State, getPin16State) | |

Table 6.6: List of KST methods for physical interaction.

| Method | Description |
|---|---|
| `startHandGuiding` | Initializes hand-guiding functionality |
| `startPreciseHandGuiding` | Initializes precision hand-guiding functionality |
| `performEventFunctionAtDoubleHit` | Detects double touch |
| `eventFunctionAtDoubleHit` | Double touch event |
| `moveWaitForDTWhenInterrupted` | Interruptible linear motion of the end-effector with physical interaction |

# Appendix C

### Gravity compensation of IMU measurements

To compensate for gravity effect in the acceleration measurements of the IMU, first a calibration of the sensor frame in relation to end-effector frame shall be established. This phase is important because the geometry of the sensor's holder (fixture) is not perfect and also due to the uncertainties in the assembly of the sensor on the end-effector. As such the rotation matrix $\mathbf{R}^s_{ef}$, from EEF frame to sensor frame, shall be established. To do this, a calibration phase is proposed. Since that the relationship between the measured gravity vector $\boldsymbol{g}^s$ in the sensor frame, and the gravity vector in end-effector frame $\boldsymbol{g}^{ef}$:

$$\boldsymbol{g}^s = \mathbf{R}^s_{ef}\boldsymbol{g}^{ef} \tag{6.8}$$

And since that:

$$\mathbf{R}^s_{ef} = \left[\vec{X}, \vec{Y}, \vec{Z}\right] \tag{6.9}$$

Where $\vec{X}, \vec{Y}$ and $\vec{Z}$ are the direction vectors along the x, y and z axes (respectively) of EEF frame described in sensor frame. Based on the previous equations three steps are proposed to calculate the rotation matrix $\mathbf{R}^s_{ef}$:

In the first step, the x axis of the end-effector frame is positioned vertically facing upwards. Afterwards, the measurement of the gravity vector $\boldsymbol{g}^s$ using the IMU is acquired, while the robot is static. The relationship between the gravity vector measured in sensor frame and the gravity vector in end-effector frame:

$$\boldsymbol{g}^s = \mathbf{R}^s_{ef}\begin{bmatrix} -\|\boldsymbol{g}^s\| \\ 0 \\ 0 \end{bmatrix} \tag{6.10}$$

Where $\|\boldsymbol{g}^s\|$ is the magnitude of the vector $\boldsymbol{g}^s$. By analyzing the previous equation, the first column vector $\vec{X}$ of the matrix $\mathbf{R}^s_{ef}$ can be calculated by:

$$\vec{X} = \frac{\boldsymbol{g}^s}{-\|\boldsymbol{g}^s\|} \tag{6.11}$$

Figure 6.1 shows configuration of the robot and the sensor for the first calibration step.

In the second step, the y axis of the end-effector is positioned vertically facing upwards, afterwards the measurement of the gravity vector $\boldsymbol{g}^s$ using the accelerometer is acquired, while the robot is static. The relationship between the gravity vector measured in sensor frame and the gravity vector in end-effector frame:

$$\boldsymbol{g}^s = \mathbf{R}^s_{ef}\begin{bmatrix} 0 \\ -\|\boldsymbol{g}^s\| \\ 0 \end{bmatrix} \tag{6.12}$$

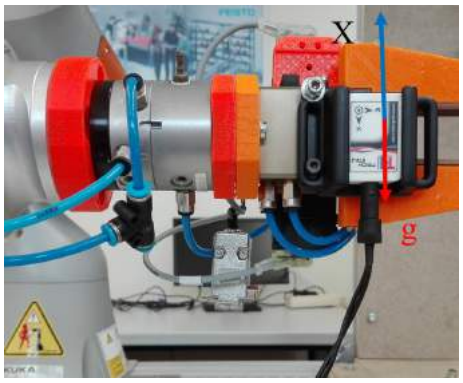


Figure 6.1: Calibration phase first step, x axis of the EEF is positioned vertically facing upwards (front view).


Figure 6.2: Calibration phase second step, y axis of the EEF is positioned vertically facing upwards (upper view).

Where $\|\boldsymbol{g}^s\|$ is the magnitude of the vector $\boldsymbol{g}^s$. By analyzing the previous equation, an approximation $\vec{Y}_0$ of the second column vector $\vec{Y}$ of the matrix $\mathbf{R}^s_{ef}$ can be calculated:

$$\vec{Y}_0 = \frac{\boldsymbol{g}^s}{-\|\boldsymbol{g}^s\|} \tag{6.13}$$

The unit vectors $\vec{X}$ and $\vec{Y}$ of the matrix $\mathbf{R}^s_{ef}$ shall be perfectly normal to each others. As such a Gram-Schmidt like procedures shall be applied to guarantee that $\vec{X}$ and $\vec{Y}$ are orthogonal, this is important to correct for the small errors coming from (1) the inaccuracy in end-effector positioning which depends on the precision of the robot, (2) the measurement uncertainty which depends on the precision of the inertial measuring unit and (3) the external noise due to vibrations and other interfering effects.

$$\vec{Y} = \frac{\vec{Y}_0 - (\vec{Y}_0^T \vec{X})\vec{X}}{\left\|\vec{Y}_0 - (\vec{Y}_0^T \vec{X})\vec{X}\right\|} \tag{6.14}$$

Figure 6.2 shows configuration of the robot and the sensor in the second calibration step.

For calculating the third column of the rotation matrix $\vec{Z}$, a similar procedure could be carried out. Alternatively, the $\vec{Z}$ vector can be calculated from the cross product:

$$\vec{Z} = \vec{X} \times \vec{Y} \tag{6.15}$$

Thus the vibrations $\boldsymbol{\Gamma}^s$ are calculated from the raw measurements of the IMU $\boldsymbol{u}^s$ after gravity compensation using the following equation:

$$\boldsymbol{\Gamma}^s = \boldsymbol{u}^s - \mathbf{R}^s_{ef}\mathbf{R}^{ef}_b \begin{bmatrix} 0 \\ 0 \\ -\|\boldsymbol{g}\| \end{bmatrix} \tag{6.16}$$

Where $\mathbf{R}^{ef}_b$ is the rotation matrix of the robot base frame to EEF frame. Previous equation applies for upright base mounted robots, as the case in our experiment.

# Bibliography


[1] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2009.

[2] Peter Bastian. *Lecture Notes on Scientific Computing with Partial Differential Equations*. Universitat Heidelberg, 2014.

[3] *Colrobot project from the Horizon 2020 research and innovation program, more info are available on-line*. https://colrobot.eu/.

[4] Knut B Kaldestad, Sami Haddadin, Rico Belder, Geir Hovland, David Anisi, et al. Collision avoidance with potential fields based on parallel processing of 3D-point cloud data on the GPU. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3250–3257. IEEE, 2014.

[5] Philip Schneider and David H Eberly. *Geometric tools for computer graphics*. Elsevier, 2002.

[6] Mohammad Safeea. *A video segment demonstrating on-line collision avoidance in an industrial robotic cell, Test 3*. `https://youtu.be/fP3gHxWX_FA/`, 2019.

[7] Knut Berg Kaldestad. Industrial robot collision handling in harsh environments. 2014.

[8] Mohammad Safeea, Richard Bearee, and Pedro Neto. Collision Avoidance of Redundant Robotic Manipulators Using Newton's Method. *Journal of Intelligent & Robotic Systems*, 2020.

[9] Mohammad Safeea, Pedro Neto, and Richard Bearee. Robot dynamics: A recursive algorithm for efficient calculation of Christoffel symbols. *Mechanism and Machine Theory*, 142:103589, 2019.

[10] Mohammad Safeea, Pedro Neto, and Richard Bearee. On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case. *Robotics and Autonomous Systems*, 119:278 – 288, 2019.

[11] Mohammad Safeea and Pedro Neto. KUKA Sunrise Toolbox: Interfacing Collaborative Robots With MATLAB. *IEEE Robotics & Automation Magazine*, 26(1):91–96, March 2019.

[12] Mohammad Safeea, Pedro Neto, and Richard Bearee. Efficient calculation of minimum distance between capsules and its use in robotics. *IEEE Access*, 7:5368–5373, 2019.

[13] Mohammad Safeea and Pedro Neto. Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction. *Robotics and Computer-Integrated Manufacturing*, 58:33 – 42, 2019.

[14] Mohammad Safeea, Richard Bearee, and Pedro Neto. Reducing the Computational Complexity of Mass-Matrix Calculation for High DOF Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5614–5619, Oct 2018.

[15] Mohammad Safeea and Pedro Neto. Human-robot collision avoidance for industrial robots: A V-REP based solution. In *Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0: Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering, July 3–6, 2018*, volume 7, page 301. IOS Press, 2018.

[16] Mohammad Safeea, Richard Bearee, and Pedro Neto. End-effector precise hand-guiding for collaborative robots. In *Iberian Robotics conference*, pages 595–605. Springer, 2017.

[17] Mohammad Safeea, Pedro Neto, and Richard Bearee. Precise hand-guiding of redundant manipulators with null space control for in-contact obstacle navigation. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 693–698, Oct 2019.

[18] Mohammad Safeea, Nuno Mendes, and Pedro Neto. Minimum distance calculation for safe human robot interaction. *Procedia Manufacturing*, 11:99–106, 2017.

[19] Mohammad Safeea, Pedro Neto, and Richard Béarée. A geometric dynamics algorithm for serially linked robots. In *Annual Conference Towards Autonomous Robotic Systems*, pages 425–435. Springer, 2019.

[20] Mohammad Safeea, Pedro Neto, and Richard Béarée. The third hand, cobots assisted precise assembly. In *Annual Conference Towards Autonomous Robotic Systems*, pages 454–457. Springer, 2019.

[21] Mohammad Safeea, Pedro Neto, and Richard Béarée. A quest towards safe human robot collaboration. In *Annual Conference Towards Autonomous Robotic Systems*, pages 493–495. Springer, 2019.

[22] Pedro Neto, Miguel Simão, Nuno Mendes, and Mohammad Safeea. Gesture-based human-robot interaction for human assistance in manufacturing. *The International Journal of Advanced Manufacturing Technology*, Oct 2018.

[23] Nuno Mendes, Mohammad Safeea, and Pedro Neto. Flexible programming and orchestration of collaborative robotic manufacturing systems. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 913–918, July 2018.

[24] João Lopes, Miguil Simão, Nuno Mendes, Mohammad Safeea, Jose Afonso, and Pedro Neto. Hand/arm Gesture Segmentation by Motion Using IMU and EMG Sensing. *Procedia Manufacturing*, 11:107 – 113, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[25] Nuno Mendes, João Ferrer, João Vitorino, Mohammad Safeea, and Pedro Neto. Human behavior and hand gesture classification for smart human-robot interaction. *Procedia Manufacturing*, 11:91–98, 2017.

[26] Mohammad Safeea. KUKA Sunrise Toolbox: a software for controlling KUKA iiwa robots from an external computer using MATLAB. `https://github.com/Modi1987/KST-Kuka-Sunrise-Toolbox/`. Accessed: 2018-12-31.

[27] Mohammad Safeea. SimulinkIIWA: a software for controlling KUKA iiwa robots from an external computer using Simulink. `https://github.com/Modi1987/Simulink-iiwa-interface/`. Accessed: 2019-04-31.

[28] Mohammad Safeea. iiwaPy: a software for controlling KUKA iiwa robots from an external computer using Python. `https://github.com/Modi1987/iiwaPy/`. Accessed: 2019-04-31.

[29] Mohammad Safeea. GDA Algorithm: A MATLAB implementation of dynamics algrithms for serially linked robots. `https://github.com/Modi1987/The-Geometric-Dynamics-Algorithm-GDA-/`. Accessed: 2019-04-31.

[30] Mohammad Safeea. C++ implementation of GDAHJ algorithm for calculating mass matrix of robots with a minimal second order cost. `https://github.com/Modi1987/CRBA-and-GDAHJ-performance-comparison-in-C-/`. Accessed: 2019-06-06.

[31] Mohammad Safeea. Recursive algorithm for calculating Christoffel symbols efficiently implemented in MATLAB. `https://github.com/Modi1987/recursiveChristoffelSymbols/`. Accessed: 2019-06-06.

[32] John J Craig. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River, 2005.

[33] Jacques Denavit and Richard Scheunemann Hartenberg. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 23:215–221, 1955.

[34] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer Science & Business Media, 2011.

[35] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, 108(3):163–171, 1986.

[36] S Chiaverini, O Egeland, and RK Kanestrom. Achieving user-defined accuracy with damped least-squares inverse kinematics. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 672–677. IEEE, 1991.

[37] N. Dehio, D. Kubus, and J. J. Steil. Continuously shaping projections and operational space tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5995–6002, Oct 2018.

[38] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[39] Cheol Chang, Myung Jin Chung, and Bum Hee Lee. Collision avoidance of two general robot manipulators by minimum delay time. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(3):517–522, 1994.

[40] Su Il Choi and Byung Kook Kim. Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, 18(02):143–151, 2000.

[41] Paul Bosscher and Daniel Hedman. Real-time collision avoidance algorithm for robotic manipulators. *Industrial Robot: An International Journal*, 38(2):186–197, 2011.

[42] Fan Ouyang and Tie Zhang. Virtual velocity vector-based offline collision-free path planning of industrial robotic manipulator. *International Journal of Advanced Robotic Systems*, 2015.

[43] Farshid Shadpey. *Force control and collision avoidance strategies for kinematically redundant manipulators*. PhD thesis, Concordia University, 1997.

[44] Cheng Fang, Alessio Rocchi, Enrico Mingo Hoffman, Nikos G. Tsagarakis, and Darwin G. Caldwell. Efficient self-collision avoidance based on focus of interest for humanoid robots. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 1060–1066, Nov 2015.

[45] Lucian Balan and Gary M Bone. Real-time 3d collision avoidance method for safe human and robot coexistence. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference On*, pages 276–282. IEEE, 2006.

[46] Sonny Tarbouriech and Wael Suleiman. On bisection continuous collision checking method: Spherical joints and minimum distance to obstacles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7613–7619. IEEE, 2018.

[47] Stefan Kuhn and Dominik Henrich. Fast vision-based minimum distance determination between known and unkown objects. In *IROS*, pages 2186–2191. IEEE, 2007.

[48] Enrique J Bernabeu and Josep Tornero. Hough transform for distance computation and collision avoidance. *IEEE Transactions on Robotics and Automation*, 18(3):393–398, 2002.

[49] Philipp Ennen, Daniel Ewert, Daniel Schilberg, and Sabina Jeschke. Efficient collision avoidance for industrial manipulators with overlapping workspaces. In *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*, pages 479–489. Springer, 2016.

[50] Agostino De Santis, Alin Albu-Schaffer, Christian Ott, Bruno Siciliano, and Gerd Hirzinger. The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–6. IEEE, 2007.

[51] RV Patel and F Shadpey. 3 Collision Avoidance for a 7-DOF Redundant Manipulator. In *Control of Redundant Robot Manipulators*, pages 35–78. Springer, 2005.

[52] Dirk Biermann, Raffael Joliet, and Thomas Michelitsch. Fast distance computation between cylinders for the design of mold temperature control systems. 2008.

[53] P Chotiprayanakul, DK Liu, D Wang, and G Dissanayake. A 3-dimensional force field method for robot collision avoidance in complex environments. In *Automation and Robotics in Construction-Proceedings of the 24th International Symposium on Automation and Robotics in Construction*, 2007.

[54] Yonghong Bu and Stephen Cameron. Active motion planning and collision avoidance for redundant manipulators. In *Assembly and Task Planning, 1997. ISATP 97., 1997 IEEE International Symposium on*, pages 13–18. IEEE, 1997.

[55] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. A depth space approach to human-robot collision avoidance. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 338–345. IEEE, 2012.

[56] Frank Wallhoff, Jürgen Blume, Alexander Bannat, Wolfgang Rösel, Claus Lenz, and Alois Knoll. A skill-based approach towards hybrid assembly. *Advanced Engineering Informatics*, 24(3):329–339, 2010.

[57] Leena Singh, Harry Stephanou, and John Wen. Real-time robot motion control with circulatory fields. In *Proceedings., 1996 IEEE International Conference on Robotics and Automation*, volume 3, pages 2737–2742. IEEE, 1996.

[58] Sami Haddadin, Rico Belder, and Alin Albu-Schäffer. Dynamic motion planning for robots in partially unknown environments. In *IFAC World Congress (IFAC2011), Milano, Italy*, volume 18, 2011.

[59] Rabie Soukieh, Iman Shames, and Baris Fidan. Obstacle avoidance of non-holonomic unicycle robots based on fluid mechanical modeling. In *Control Conference (ECC), 2009 European*, pages 3269–3274. IEEE, 2009.

[60] Rainer Palm and Dimiter Driankov. Fluid mechanics for path planning and obstacle avoidance of mobile robots. In *Informatics in Control, Automation and*

*Robotics (ICINCO), 2014 11th International Conference on*, volume 2, pages 231–238. IEEE, 2014.

[61] Ahmad Masoud, Samer Masoud, Mohamed M Bayoumi, et al. Robot navigation using a pressure generated mechanical stress field: the biharmonic potential approach. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 124–129. IEEE, 1994.

[62] T. Nishimura, K. Sugawara, I. Yoshihara, and K. Abe. A motion planning method for a hyper multi-joint manipulator using genetic algorithm. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 4, pages 645–650 vol.4, 1999.

[63] Leon Žlajpah and Bojan Nemec. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1898–1903. IEEE, 2002.

[64] L Žlajpah and T Petrič. Serial and parallel robot manipulators kinematics, dynamics, control and optimization, chap obstacle avoidance for redundant manipulators as control problem, 2012.

[65] S Mitsi, K-D Bouzakis, and G Mansour. Optimization of robot links motion in inverse kinematics solution considering collision avoidance and joint limits. *Mechanism and machine theory*, 30(5):653–663, 1995.

[66] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[67] Tobias Kunz, Ulrich Reiser, Mike Stilman, and Alexander Verl. Real-time path planning for a robot arm in changing environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5906–5911. IEEE, 2010.

[68] Peter Leven and Seth Hutchinson. A framework for real-time path planning in changing environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002.

[69] Frank Dittrich, Stephan Puls, and Heinz Wörn. A Modular Cognitive System for Safe Human Robot Collaboration: Design, Implementation and Evaluation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), WS Robotic Assistance Technologies in Industrial Settings*, page 10, 2013.

[70] Ahmad Yasser Afaghani and Yasumichi Aiyama. On-line collision avoidance between two robot manipulators using collision map and simple escaping method. In *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration, SII 2013, Kobe, Japan, December 15-17, 2013*, pages 105–110, 2013.

[71] Ahmad Yasser Afaghani and Yasumichi Aiyama. Advanced-collision-map-based on-line collision and deadlock avoidance between two robot manipulators with ptp commands. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1244–1251. IEEE, 2014.

[72] Ahmad Yasser Afaghani and Yasumichi Aiyama. On-line collision detection of n-robot industrial manipulators using advanced collision map. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 422–427. IEEE, 2015.

[73] Jianing Zhou, Kazuyuki Nagase, Shinya Kimura, and Yasumichi Aiyama. Collision avoidance of two manipulators using rt-middleware. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1031–1036. IEEE, 2011.

[74] Fumi Seto, Kazuhiro Kosuge, and Yasuhisa Hirata. Self-collision Avoidance Motion Control for Human Robot Cooperation System using RoBE. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005.

[75] J Ernesto Solanes, Luis Gracia, Pau Muñoz-Benavent, Jaime Valls Miro, Marc G Carmichael, and Josep Tornero. Human-robot collaboration for safe object transportation using force feedback. *Robotics and Autonomous Systems*, 2018.

[76] Stefanos Nikolaidis, Przemyslaw Lasota, Ramya Ramakrishnan, and Julie Shah. Improved human–robot team performance through cross-training, an approach inspired by human team training practices. *The International Journal of Robotics Research*, 34(14):1711–1730, 2015.

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

[78] J. Saenz, N. Elkmann, O. Gibaru, and P. Neto. Survey of methods for design of collaborative robotics applications-why safety is a barrier to more widespread robotics uptake. In *4th International Conference on Mechatronics and Robotics Engineering*, 2018.

[79] Xiaoyu Chu, Quan Hu, and Jingrui Zhang. Path planning and collision avoidance for a multi-arm space maneuverable robot. *IEEE Transactions on Aerospace and Electronic Systems*, 54(1):217–232, 2018.

[80] Boseong Jeon, Hyoin Kim, and H Jin Kim. Collision avoidance of robotic arm of aerial manipulator. In *Control Conference (ASCC), 2017 11th Asian*, pages 1859–1864. IEEE, 2017.

[81] Fabrizio Flacco and Alessandro De Luca. Real-time computation of distance to dynamic obstacles with multiple depth sensors. *IEEE Robotics and Automation Letters*, 2(1):56–63, Jan 2017.

[82] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Control of Redundant Robots Under Hard Joint Constraints: Saturation in the Null Space. *IEEE Transactions on Robotics*, 31(3):637–654, June 2015.

[83] A. Fenucci, M. Indri, and F. Romanelli. A real time distributed approach to collision avoidance for industrial manipulators. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, Sept 2014.

[84] Andrea Maria Zanchettin, Bakir Lacevic, and Paolo Rocco. Passivity-based control of robotic manipulators for safe cooperation with humans. *International Journal of Control*, 88(2):429–439, 2015.

[85] Ramy Meziane, Martin J.-D. Otis, and Hassan Ezzaidi. Human-robot collaboration while sharing production activities in dynamic environment: Spader system. *Robotics and Computer-Integrated Manufacturing*, 48(Supplement C):243 – 253, 2017.

[86] Francisco Rubio, Carlos Llopis-Albert, Francisco Valero, and Josep Lluis Suñer. Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory. *Robotics and Autonomous Systems*, 86:106–112, 2016.

[87] Véronique Perdereau, C Passi, and Michel Drouin. Real-time control of redundant robotic manipulators for mobile obstacle avoidance. *Robotics and Autonomous Systems*, 41(1):41–59, 2002.

[88] Ashwini Shukla, Ekta Singla, Pankaj Wahi, and Bhaskar Dasgupta. A direct variational method for planning monotonically optimal paths for redundant manipulators in constrained workspaces. *Robotics and Autonomous Systems*, 61(2):209–220, 2013.

[89] Rainer Palm. Control of a redundant manipulator using fuzzy rules. *Fuzzy Sets and Systems*, 45(3):279 – 298, 1992.

[90] Bernard Schmidt and Lihui Wang. Depth camera based collision avoidance via active robot control. *Journal of Manufacturing Systems*, 33(4):711 – 718, 2014.

[91] Abdullah Mohammed, Bernard Schmidt, and Lihui Wang. Active collision avoidance for human robot collaboration driven by vision sensors. *International Journal of Computer Integrated Manufacturing*, 30(9):970–980, 2017.

[92] Alexander Skoglund, Boyko Iliev, Bourhane Kadmiry, and Rainer Palm. Programming by demonstration of pick-and-place tasks for industrial manipulators using task primitives. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, pages 368–373. IEEE, 2007.

[93] Mohammad Safeea. *A video segment demonstrating collision avoidance with human coworker, Test 2.* `https://youtu.be/ljVXxmENFGo/`, 2019.

[94] Yoshihiko Nakamura. *Advanced robotics: redundancy and optimization.* Addison-Wesley Longman Publishing Co., Inc., 1990.

[95] Lorenzo Sciavicco, Bruno Siciliano, and Luigi Villani. Lagrange and newton-euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects. *Advanced robotics*, 10(3):317–334, 1995.

[96] M. Parigi Polverini, A. M. Zanchettin, and P. Rocco. Real-time collision avoidance in human-robot interaction based on kinetostatic safety field. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4136–4141, Sept 2014.

[97] Fabrizio Flacco and Alessandro De Luca. A reverse priority approach to multi-task control of redundant robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2421–2427, Sept 2014.

[98] Jinsuck Kim, Roger A Pearce, and Nancy M Amato. Extracting optimal paths from roadmaps for motion planning. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2424–2429. IEEE, 2003.

[99] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[100] Jing Ren, Kenneth A McIsaac, and Rajnikant V Patel. Modified newton's method applied to potential field-based navigation for mobile robots. *IEEE Transactions on Robotics*, 22(2):384–391, 2006.

[101] Jing Ren, Kenneth A McIsaac, and Rajni V Patel. Modified newton's method applied to potential field-based navigation for nonholonomic robots in dynamic environments. *Robotica*, 26(1):117–127, 2008.

[102] K. A. McIsaac, Jing Ren, and Xishi Huang. Modified newton's method applied to potential field navigation. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 6, pages 5873–5878 Vol.6, Dec 2003.

[103] J Denhavit. A kinematic notation for lower-pair mechanisms based on matrices. *ASME J. Appl. Mech.*, 1955.

[104] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[105] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

[106] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

[107] Mohammad Safeea. *A video segment showing collision avoidance simulation for hyper-redundant manipulator using Newton method in Test 1.* `https://youtu.be/4zq59RSVcrk/`, 2019.

[108] Mohammad Safeea. *A video segment showing collision avoidance simulation for hyper-redundant manipulator using Newton method in Test 2.* `https://youtu.be/fKGaL9EJA3I/`, 2019.

[109] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[110] Peter I Corke. A robotics toolbox for MATLAB. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996.

[111] Peter Corke. MATLAB toolboxes: robotics and vision for students and teachers. *IEEE Robotics & Automation Magazine*, 14(4), 2007.

[112] P. I. Corke. The Machine Vision Toolbox: a MATLAB toolbox for vision and vision-based control. *IEEE Robotics & Automation Magazine*, 12(4):16–25, Dec 2005.

[113] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo. Kuka control toolbox. *IEEE Robotics & Automation Magazine*, 18(4):69–79, Dec 2011.

[114] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen. Controlling Kuka Industrial Robots: Flexible Communication Interface JOpenShowVar. *IEEE Robotics & Automation Magazine*, 22(4):96–109, Dec 2015.

[115] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen. Jopenshowvar: An open-source cross-platform communication interface to kuka robots. In *2014 IEEE International Conference on Information and Automation (ICIA)*, pages 1154–1159, July 2014.

[116] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[117] *ROS Indigo/Kinetic metapackage for the KUKA LBR IIWA R800/R820 [Online]*. Available: http://github.com/IFL-CAMP/iiwa_stack/.

[118] Saeid Mokaram, Jonathan M Aitken, Uriel Martinez-Hernandez, Iveta Eimontaite, David Cameron, Joe Rolph, Ian Gwilt, Owen McAree, and James Law. A ROS-integrated api for the kuka lbr iiwa collaborative robot. *IFAC-PapersOnLine*, 50(1):15859–15864, 2017.

[119] Hidetoshi Fukui, Satoshi Yonejima, Masatake Yamano, Masao Dohi, Mariko Yamada, and Tomonori Nishiki. Development of teaching pendant optimized for robot application. In *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, pages 72–77. IEEE, 2009.

[120] Hoo Man Lee and Joong Bae Kim. A survey on robot teaching: Categorization and brief review. In *Applied Mechanics and Materials*, volume 330, pages 648–656. Trans Tech Publ, 2013.

[121] Daisuke Kushida, Masatoshi Nakamura, Satoru Goto, and Nobuhiro Kyura. Human direct teaching of industrial articulated robot arms based on force-free control. *Artificial Life and Robotics*, 5(1):26–32, 2001.

[122] Daniele Massa, Massimo Callegari, and Cristina Cristalli. Manual guidance for industrial robot programming. *Industrial Robot*, 42(5):457–465, 2015.

[123] Guilherme Boulhosa Rodamilans, EmÃlia Villani, LuÃs Gonzaga Trabasso, Wesley Rodrigues de Oliveira, and Ricardo Suterio. A comparison of industrial robots interface: force guidance system and teach pendant operation. *Industrial Robot*, 43(5):552–562, 2016.

[124] Matteo Ragaglia, Andrea Maria Zanchettin, Luca Bascetta, and Paolo Rocco. Accurate sensorless lead-through programming for lightweight robots in structured environments. *Robotics and Computer-Integrated Manufacturing*, 39:9–21, 2016.

[125] Luca Bascetta, Gianni Ferretti, Gianantonio Magnani, and Paolo Rocco. Walk-through programming for robotic manipulators based on admittance control. *Robotica*, 31(7):1143–1153, 2013.

[126] Ravi Balasubramanian, Ling Xu, Peter D Brook, Joshua R Smith, and Yoky Matsuoka. Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *IEEE Transactions on Robotics*, 28(4):899–910, 2012.

[127] Sonehara M. Fujii M., Murakami H. Study on application of a human-robot collaborative system using hand-guiding in a production line. In *IHI Engineering Review, Vol.49 (1) 24-29*, 2016.

[128] Christian Emmerich, Arne Nordmann, Agnes Swadzba, Jochen J Steil, and Sebastian Wrede. Assisted gravity compensation to cope with the complexity of kinesthetic teaching on redundant robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4322–4328. IEEE, 2013.

[129] Gianni Ferretti, Gianantonio Magnani, and Paolo Rocco. Assigning virtual tool dynamics to an industrial robot through an admittance controller. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

[130] Magnus Hanses, Roland Behrens, and Norbert Elkmann. Hand-guiding robots along predefined geometric paths under hard joint constraints. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–5. IEEE, 2016.

[131] Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen Steil. A user study on kinesthetic teaching of redundant robots in task and configuration space. *J. Hum.-Robot Interact.*, 2(1):56–81, February 2013.

[132] Sang-Duck Lee, Kuk-Hyun Ahn, and Jae-Bok Song. Torque control based sensorless hand guiding for direct robot teaching. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 745–750. IEEE, 2016.

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

[134] Fanny Ficuciello, Luigi Villani, and Bruno Siciliano. Variable impedance control of redundant manipulators for intuitive human–robot physical interaction. *IEEE Transactions on Robotics*, 31(4):850–863, 2015.

[135] K Kosuge, H Yoshida, and T Fukuda. Dynamic control for robot-human collaboration. In *Robot and Human Communication, 1993. Proceedings., 2nd IEEE International Workshop on*, pages 398–401. IEEE, 1993.

[136] Mohammad Safeea. *A video segment demonstrating the motion groups for the precise hand-guiding.* `https://youtu.be/yodLQ0YFJEM/`, 2019.

[137] Mohammad Safeea. *A video segment demonstrating Test 3 for precision hand-guiding.* `https://youtu.be/Rvqq-OOtUkA/`, 2019.

[138] Mohammad Safeea. *A video segment demonstrating Test 3 for KUKA off-the-shelf hand-guiding.* `https://youtu.be/xKlyHxptIEE/`, 2019.

[139] Mohammad Safeea. *A video segment showing programming by demonstration using the precision hand-guiding.* `https://youtu.be/pdvLkcEUcXM/`, 2019.

[140] KUKA iiwa Connectivity Manual. *KUKA Sunrise.Connectivity Servoing 1.7. KUKA Roboter GmbH, 2015.*

[141] Mohammad Safeea. *A video segment demonstrating the precision hand-guiding with in-contact obstacle navigation method applied to KUKA iiwa 7R800 robot.* `https://youtu.be/XT_hsQ7A6uO/`, 2019.

[142] Roy Featherstone and David Orin. Robot dynamics: equations and algorithms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 826–834, 2000.

[143] CA Balafoutis. A survey of efficient computational methods for manipulator inverse dynamics. *Journal of Intelligent and Robotic Systems*, 9(1-2):45–71, 1994.

[144] Gregory S Chirikjian and Joel W Burdick. Design and experiments with a 30 dof robot. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 113–119. IEEE, 1993.

[145] Oussama Khatib. Dynamic control of manipulator in operational space. In *Proc. 6th IFToMM World Congress on Theory of Machines and Mechanisms*, pages 1128–1131, 1983.

[146] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, 1987.

[147] Kyong-Sok Chang. *Efficient algorithms for articulated branching mechanisms: dynamic modeling, control, and simulation.* PhD thesis, Stanford, 2000.

[148] Kyong-Sok Chang and Oussama Khatib. Efficient recursive algorithm for the operational space inertia matrix of branching mechanisms. *Advanced Robotics*, 14(8):703–715, 2001.

[149] Wisama Khalil. Dynamic modeling of robots using recursive newton-euler techniques. In *ICINCO2010*, 2010.

[150] Mark W Spong. Control of robots and manipulators. *The control handbook*, pages 1339–1351, 1996.

[151] Metin Toz and Serdar Kucuk. Dynamics simulation toolbox for industrial robot manipulators. *Computer Applications in Engineering Education*, 18(2):319–330, 2010.

[152] Charles P Neuman and John J Murray. Symbolically efficient formulations for computational robot dynamics. *Journal of robotic systems*, 4(6):743–769, 1987.

[153] Herman Hoifodt. Dynamic modeling and simulation of robot manipulators: the newton-euler formulation. 2011.

[154] Thomas R Kane and David A Levinson. The use of kane's dynamical equations in robotics. *The International Journal of Robotics Research*, 2(3):3–21, 1983.

[155] H. Baruh. *Analytical dynamics.* MacGraw-Hill, 1999.

[156] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980.

[157] Michael W Walker and David E Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 1982.

[158] Roy Featherstone. Spatial vectors and rigid-body dynamics. `http://royfeatherstone.org/spatial/`.

[159] Roy Featherstone. Plucker basis vectors. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1892–1897. IEEE, 2006.

[160] Roy Featherstone. *Rigid body dynamics algorithms.* Springer, 2014.

[161] Roy Featherstone. Efficient factorization of the joint-space inertia matrix for branched kinematic trees. *The International Journal of Robotics Research*, 24(6):487–500, 2005.

[162] M Vukobratović, Shi-Gang Li, and N Kirćanski. An efficient procedure for generating dynamic manipulator models. *Robotica*, 3(03):147–152, 1985.

[163] Charles W Misner, Kip S Thorne, and John Archibald Wheeler. *Gravitation*. W.H. Freeman and Company, 1973.

[164] Yang Liu and Hongnian Yu. A survey of underactuated mechanical systems. *IET Control Theory & Applications*, 7(7):921–935, 2013.

[165] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9):867–875, 1999.

[166] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 2: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9):876–892, 1999.

[167] R Gunawardana and F Ghorbel. On the uniform boundedness of the coriolis/centrifugal terms in the robot equations of motion. *International Journal of Robotics and Automation*, 14(2):45–53, 1999.

[168] Rafael Kelly. Comments on Adaptive PD controller for robot manipulators. *IEEE Transactions on Robotics and Automation*, 9(1):117–119, 1993.

[169] Patrizio Tomei. Adaptive PD controller for robot manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):565–570, 1991.

[170] Jose Alvarez-Ramirez, Ilse Cervantes, and Rafael Kelly. PID regulation of robot manipulators: stability and performance. *Systems & control letters*, 41(2):73–83, 2000.

[171] Ilse Cervantes and Jose Alvarez-Ramirez. On the PID tracking control of robot manipulators. *Systems & control letters*, 42(1):37–46, 2001.

[172] Zhihua Qu and John Dorsey. Robust tracking control of robots by a linear feedback law. *IEEE Transactions on Automatic Control*, 36(9):1081–1084, 1991.

[173] R Ortega, A Loria, and R Kelly. A semiglobally stable output feedback PID regulator for robot manipulator, automatic control. *IEEE Transaction August*, 40, 1995.

[174] Juan Ignacio Mulero-Martinez. An improved dynamic neurocontroller based on Christoffel symbols. *IEEE transactions on neural networks*, 18(3):865–879, 2007.

[175] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[176] S Yin and J Yuh. An efficient algorithm for automatic generation of manipulator dynamic equations. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1812–1817. IEEE, 1989.

[177] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, New York, NY, USA, 1st edition, 2017.

[178] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[179] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[180] P Reynoso-Mora, W Chen, and M Tomizuka. A convex relaxation for the time-optimal trajectory planning of robotic manipulators along predetermined geometric paths. *Optimal Control Applications and Methods*, 37(6):1263–1281, 2016.

[181] Diederik Verscheure, Bram Demeulenäre, Jan Swevers, Joris De Schutter, and Moritz Diehl. Practical time-optimal trajectory planning for robots: a convex optimization approach. *IEEE Transactions on Automatic Control*, 2008.