TOWARDS A SAFE INTERACTION BETWEEN HUMANS AND INDUSTRIAL ROBOTS THROUGH PERCEPTION ALGORITHMS AND CONTROL STRATEGIES

Doctoral Dissertation of:
Matteo Ragaglia

Supervisor:
Prof. Paolo Rocco

Tutor:
Prof. Luca Bascetta

The Chair of the Doctoral Program:
Prof. Andrea Bonarini

2016 – XXVIII
Acknowledgements

First of all, I would like to thank Prof. Paolo Rocco for the experience that I have had the opportunity to enjoy in these three years of PhD program. I wish to thank him for all the fruitful discussions, and for his precious support and guidance during the whole PhD program.

I am especially grateful to Prof. Luca Bascetta and to Dr. Andrea Zanchettin for their kind availability and for the huge amount of practical help, interesting discussions and suggestions I received from them.

I also want to thank Prof. Gianantonio Magnani for the opportunity of working with him and all my office colleagues: Giovanni Buizza Avanzini, Nicola Ceriani, Amir Ghalamzan, Cecilia Lamperti, Davide Nicolis, Matteo Parigi Polverini, Roberto Rossi and Antonio Valsecchi, that have been good friends and great travelling companions in this journey.

Finally, I wish to thank Prof. Luigi Villani from Università Federico II di Napoli, who served as a reviewer of this thesis, for the valuable suggestions on how to improve the quality of this work and on how to strengthen the thoroughness of the results.
In the past few years the need for more flexibility in industrial production has implied, in the field of industrial robotics, a growing attention towards the possibility of making humans work directly in touch with robots. As a matter of fact, it is today a common opinion that Human-Robot Interaction (HRI) represents the key factor that will facilitate industrial robots to spread in Small and Medium sized Enterprises (SMEs).

Nevertheless, HRI introduces a series of safety issues which are uncommon in industrial settings where physical separation between robot and human workspaces is typically enforced. In order to achieve safe and efficient HRI, this thesis was developed around two main goals:

- enhance the perception capabilities of a typical control system of an industrial robot by integrating information coming from different exteroceptive sensors, like for instance RGB and depth cameras;
- develop reactive control strategies and trajectory generation algorithms that not only rely on the information acquired by these sensors, but that also guarantee human workers’ safety by satisfying safety standards and regulations.

From the perception perspective, two main problems have been approached. At first we have developed sensor fusion strategies able to merge information coming from several RGB cameras, or from multiple depth cameras, or from both kind of sensors. Then we have chosen a simple yet effective human kinematic model and we have developed algorithms able to detect, track and predict human motion on the basis of the information acquired via sensor fusion.

Switching from the perception domain to the control perspective, we initially approached the problem of formalizing safety requirements and regulations in a mathematical way. To this purpose “safety constraints” have been formalized in order to express collision avoidance requirements with respect to both a-priori known obstacles and obstacles perceived at runtime. On the basis of these safety constraints, several safety-oriented control strategies and trajectory generation algorithms have been developed in order to exploit the information acquired from the perception system.

Finally, also the problem of safety in physical HRI has been investigated, with a peculiar focus on Lead-Through Programming (LTP).
Negli ultimi anni, la necessità di rendere la produzione industriale sempre più rapida nell’adattarsi a specifiche in continua evoluzione ha determinato, nel campo della robotica industriale, un crescente interesse verso la possibilità di far lavorare operatori umani e robot a stretto contatto. Non a caso, gli esperti del settore concordano nell’affermare che l’interazione uomo-robot rappresenta, al giorno d’oggi, il fattore chiave che faciliterà la diffusione dei robot industriali all’interno delle medie e piccole imprese.

Tuttavia, l’interazione diretta uomo-robot presuppone l’eliminazione delle barriere che separano lo spazio di lavoro dedicato ai robot da quello di competenza degli operatori umani, dando così origine a rilevanti problemi di sicurezza. Nell’ottica di realizzare un’interazione uomo-robot sicura ed efficiente, questa tesi è stata sviluppata attorno a due obiettivi fondamentali:

- potenziare le capacità percettive dei tipici sistemi di controllo per robot industriali attraverso l’integrazione di informazione proveniente da svariati sensori estero-rettivi, come per esempio telecamere di sorveglianza e sensori di profondità;

- sviluppare strategie di controllo reattivo e di generazione della traiettoria che non solo si basano sulle informazioni acquisite dai sensori di cui sopra, ma, soprattutto, che siano in grado di garantire la sicurezza degli operatori umani attraverso il soddisfacimento dei requisiti di sicurezza imposti delle normative vigenti.

Per quanto concerne l’ambito della percezione, due problemi fondamentali sono stati affrontati. I primo luogo abbiamo sviluppato diverse strategie di fusione sensoriale allo scopo di integrare in maniera coerente le informazioni acquisite da telecamere RGB e sensori di profondità. Successivamente abbiamo adottato un modello cinematico dell’essere umano e abbiamo sviluppato algoritmi in grado di rilevare, effettuare il tracking e predire il movimento degli operatori umani all’interno della cella robotica.

Da ultimo, anche il problema della sicurezza durante l’interazione fisica uomo-robot è stato affrontato, ponendo l’attenzione sul tema del Lead-Through Programming (LTP).
Industrial manipulators represent a well-established technology in several industry sectors, such as automotive, or for machines tending and parts movement. However, their diffusion in Small and Medium-sized Enterprises is hampered by an insufficient flexibility, or by an excessive cost for their installation. Human-Robot Interaction (HRI) represents a promising solution to such a problem, as cooperation between robots and workers could greatly increase robots flexibility, and, at the same time, the adoption of manipulators that are safe for human robot interaction would reduce the costs related to environment structuring.

Nevertheless, the deployment of industrial robots in HRI scenarios poses new challenges for robot manufacturers and system integrators: guaranteeing safety for human operators cooperating with robots, while maximizing productivity. Robots should avoid collisions and reduce the risk of consequent injuries. At the same time, the need for safety must not diminish robots productivity, nor should it disrupt the possibility of task completion or generate a risk of damages for the manipulator or the production setup. Despite the fact that robots specifically designed for HRI are becoming increasingly available in the market, standard industrial robot controllers currently lack the features needed to ensure safe and productive HRI. Moreover, traditional manipulators largely used in the industry cannot be considered inherently safe.

For these reasons, this thesis aims at overcoming current industrial robot controllers limitations in order to achieve safe and efficient Human-Robot Interaction during the execution of cooperative tasks.

The first main goal that is tackled in this work consists in enhancing the perception capabilities of a typical control system of an industrial robot by integrating information coming from different exteroceptive sensors, especially RGB and depth cameras. Different kinematic models describing human motion are presented and several approaches to the problem of human motion tracking are proposed, relying on sensor fusion strategies that merge together data acquired from RGB cameras and depth sensors. Finally, on the basis of the tracked kinematic configuration of the human worker, the problem of human motion prediction is approached.

Moving from the perception field to the control domain, several safety-oriented control strategies and trajectory generation algorithms are proposed. These approaches are
mainly based on the formalization of the safety constraints and on the information produced by the perception system. The main goal of these control techniques is to guarantee human worker’s safety while trying to maximize the productivity of the manipulator during the execution of a cooperative task.

**Thesis Contributions and Organization**

In this thesis, the following main contributions are given:

1. novel sensor fusion techniques are presented, allowing to merge data acquired from RGB cameras and depth sensors;
2. two distinct motion tracking algorithms are proposed that rely on the previously mentioned sensor fusion strategies and on different human kinematic models in order to detect and track the motion of human worker inside a supervised robotic cell;
3. two distinct strategies for human motion prediction are developed. The first one consists in predicting the final destination of a human walking trajectory, while the second one focuses on predicting the entire volume a human worker can occupy within a time span on the basis of his/her current kinematic configuration;
4. two different trajectory generation algorithms that are able to guarantee human workers’ safety during HRI by relying on the previously mentioned motion prediction strategies;
5. a novel approach to lead-through programming that allows a human operator to safely lead the manipulator, while guaranteeing an accuracy superior with respect to traditional LTP techniques.

The thesis is organised as follows:

**Part I** tackles two distinct problems: human motion modelling and human motion tracking. The kinematic models here introduced and the novel sensor fusion strategies here proposed represent the fundamental basis on top of which several motion tracking algorithms are developed.

In particular, **Chapter 2** introduces three different kinematic models describing human motion. The first one consists in a simple non-holonomic model describing walking kinematics. Then, this first model is modified to take into account full holonomic walking, thus obtaining a second model. Finally, the model is extended to take into account the motion of the upper limbs in terms of torso, head and arms (from shoulders up to wrists).

**Chapter 3** presents an algorithm that performs human motion tracking inside a robotic cell. The proposed approach relies on the non-holonomic model of human walking kinematics and on the fusion of multiple RGB images. The obtained unique image is then processed by computer vision algorithms and, finally, a series of particle filters are used to track detected humans. Results obtained from several experiments are presented in order to prove the effectiveness of the proposed algorithm.
Finally, Chapter 4 describes an algorithm that performs depth sensor fusion and motion tracking on the basis of the full human kinematic model. The proposed solution relies on a Linear Kalman Filter (LKF) that merges the different measurements in order to estimate the kinematic configuration of the human worker. The estimation is further processed in order to satisfy fixed bounds on human joint positions, velocities and accelerations. Once again, experimental validation is proposed and the corresponding results are discussed.

On the basis of the previously described motion tracking algorithms, Part II approaches the problem of human motion prediction with a twofold approach.

First, Chapter 5 tackles the problem of predicting the final destination of a human worker’s walking trajectory inside a robotic cell. Starting from the algorithm already introduced in Chapter 3, a representation of the supervised environment is developed in order to estimate the human final destination. Finally, extensive experimental validation is presented and the corresponding results are discussed.

Then, in order to realize a prediction that takes into account the full human kinematic model, Chapter 6 presents an algorithm able to predict the entire swept volume that a human worker can occupy within a defined time span. Two different approaches are discussed. The first one is based on human joint positions and bounded joint velocities, while the second one relies on joint positions and velocities and bounded joint accelerations. Moreover, a comparison between the two approaches is presented.

Finally, moving from the perception field to the control domain, Part III presents several safety-oriented control strategies and trajectory generation algorithms for HRI.

At first, Chapter 7 introduces and extends the formalization of the safety constraints. Starting from the original formulation, that considers point-shaped obstacles only, two distinct extensions are presented. The first one allows to formalize the constraints taking into account uncertainties, like for instance robot and/or obstacles geometry and dimension. On the other hand, the second one allows to formalize safety constraints in presence of arbitrarily-shaped convex obstacles.

Then Chapter 8 proposes two distinct approaches to the safety-aware trajectory motion planning problem. The first control algorithm simply scales the robot velocity along a pre-programmed path in order to enforce safety requirements, thus guaranteeing that the robot stops before a possible collision with a human worker. The second approach consists in a point-to-point trajectory generation algorithm that allows the robot to modify the pre-programmed path in order to satisfy safety constraints. Both algorithms rely on the motion tracking and prediction strategies described in Chapters 4 and 6. Extensive experimental validation of the two algorithms is presented and the corresponding results are discussed. Furthermore, the possibility to integrate the proposed control algorithms with the motion prediction strategy described in Chapter 5 is discussed and experimentally tested.

Finally, Chapter 9 approaches with the problem of safety and efficiency during physical-HRI (i.e. HRI with direct contact between the robot and the human operator), with a specific focus on Lead-Through Programming. The proposed solution allows a human operator to safely perform accurate Lead-Through Programming (LTP) without the need of dedicated hardware. The algorithm is based on the manipulator dynamic
model, a real-time external force estimation observer, a voting system selecting the largest Cartesian component of the force/momentum applied by the operator, an implicit admittance filter and an optimization framework. The effectiveness of the proposed approach is shown through experimental validation results.

Publications

This thesis is based on the following publications:


and on the following submitted material:

Finally, the following publication contains relevant results, that are not covered in the doctoral dissertation:

1. M. Ragaglia, M. Prandini, L. Bascetta - “Poli-RRT*: optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics”, 14th annual European Control Conference, ECC 2015 - Linz (Austria) - July 2015, 15th - 17th;
## Contents

1 Introduction and State of the Art  
  1.1 Safe HRI - Assessing Interaction Safety  
  1.2 Safe HRI - Control  
    1.2.1 Collision Avoidance  
    1.2.2 Collision Detection and Reaction  
  1.3 Safe HRI - Detecting and Predicting Human Motion  

2 Perception: Modelling and Tacking Human Motion via Sensor Fusion  

2 Modelling Human Motion  
  2.1 Introduction  
  2.2 Non-Holonomic Model of Human Walking Kinematics  
  2.3 Holonomic Model of Human Walking Kinematics  
  2.4 Complete Kinematic Model of Human Motion  
    2.4.1 Human Arm Kinematics  
    2.4.2 Complete Kinematic Model  

3 Tracking Walking Motion with Multiple RGB Cameras  
  3.1 Introduction  
  3.2 Multiple RGB Images Fusion  
  3.3 Human Motion Detection via BG/FG Segmentation  
  3.4 Using K-d Trees to Update detected humans  
  3.5 Human Walking Motion Tracking via Particle Filtering  
  3.6 Experimental Validation  
    3.6.1 Experiment #1: Single Person Detection and Tracking  
    3.6.2 Experiment #2: Multiple Person Detection and Tracking  
    3.6.3 Experiment #3: Single Person Detection and Tracking with Moving Robot  


## Contents

### 4 Tracking Full Upper Body Motion with Multiple Depth Sensors

4.1 Introduction ........................................... 27
4.2 Kalman filter based multiple depth sensor fusion .......... 27
4.3 Bounded kinematic state estimation .................. 30
4.4 Experimental Validation .......................... 32

### II Perception: Predicting Human Motion

#### 5 Human Motion Prediction: Estimating Human Walking Trajectories

5.1 Introduction ........................................... 41
5.2 Mapping the Robotic Cell Environment ................. 42
5.3 Map-based Particle Filtering .......................... 45
5.3.1 Estimating Destination Areas’ Probabilities .......... 45
5.3.2 Particles Resampling Strategy ................... 46
5.4 Experimental verification .......................... 47
5.4.1 Exp. 01 - Trajectories reaching predefined destinations 47
5.4.2 Exp. 02 - Trajectories reaching undefined destination 47
5.4.3 Exp. 03 - Trajectories with switching destination .... 48
5.4.4 Exp. 04 - Multiple simultaneous trajectories .......... 48

#### 6 Human Motion Prediction: Swept Volumes

6.1 Introduction ........................................... 53
6.2 Human Kinematics Reachable Set ..................... 54
6.2.1 Reachable Set based on Joint Positions and Bounded Joint Velocities 54
6.2.2 Reachable Set based on Joint Positions and Velocities .... 56
6.3 Human Swept Volumes Calculation ................... 58
6.3.1 Swept Volumes of Convex Objects ................ 58
6.3.2 Human Swept Volumes Calculation Algorithm .......... 59
6.3.3 Experimental Validation and Discussion ............ 61

### III Control: Safety-Oriented Trajectory Generation Algorithms

#### 7 Safety Constraints

7.1 Introduction ........................................... 67
7.2 Background on Safety Constraints .................... 68
7.3 Accounting for Uncertainty in Safety Constraints ....... 70
7.4 Safety Constraints for arbitrarily-shaped convex obstacles 72

#### 8 Safety-oriented control strategies and trajectory generation algorithms

8.1 Introduction ........................................... 75
8.2 Kinematic Scaling Algorithm .......................... 76
8.2.1 Experimental Validation .......................... 77
8.3 Point-to-Point Trajectory Generation Algorithm ...... 80
8.3.1 Experimental Validation .......................... 82
8.4 Integrating prediction of human walking trajectories inside the Safety Controller ............. 85
## Contents

9 Safety in physical HRI: an approach to safe and accurate Lead-Through Programming 89
  9.1 Introduction 89
  9.2 Manipulator Dynamic Model 91
  9.3 Lead-through Programming Algorithm 92
    9.3.1 Online External Forces/Moments Estimation 93
    9.3.2 Voting System and Finite State Machine 94
    9.3.3 Admittance Filtering 95
    9.3.4 Lead-through programming algorithm and redundancy resolution 98
  9.4 Experiments 100
    9.4.1 Unknown Dynamic Parameter Identification 101
    9.4.2 Experimental Validation 101

10 Conclusions 109

Bibliography 119

A Appendix A - Forward Human Kinematics 127

B Appendix B - Inverse Human Kinematics 131
CHAPTER 1

Introduction and State of the Art

Industrial robots are able to offer fast and accurate task execution in various industrial applications like for instance: welding, painting, packaging, palletizing, etc. Nevertheless they are still rarely used within Small and Medium-sized Enterprises (SMEs), mainly because of some well-known issues. As a matter of fact installation, setup and programming of a robotized workstation are time-consuming activities that require a lot of skilled engineering effort. Moreover, industrial robots usually need to be separated from the human workspace by physical barriers (see [1]) for safety reasons. Given these limitations and considering also limited budget and constraints on space consumption, it is clear why the large majority of SMEs cannot afford to use extensively industrial robots.

In this context, Human-Robot Interaction (HRI) is one of the most interesting and promising areas of research and technology development since a continuous and fruitful cooperation between men and machines could determine massive productivity increases (and also faster adaptability to rapidly changing requirements) of all those tasks that nowadays are still accomplished mainly through human labour. Not by chance, several research projects exploring the field of HRI have been recently funded by the European Union (e.g. [6–8]).

Obviously industry has a significant interest in the development of such a technology. Even though emerging countries are still offering low cost workforce, the progressive development of such countries will necessary entail the improvement of working conditions and salaries [32], thus progressively making cheap labour less and less available. On the other hand, developed countries are seeking solutions providing a higher level of automation of the productive processes in order to mitigate the disadvantage of high labour cost.

Clearly, HRI represents a very promising solutions to the mentioned problems.
Chapter 1. Introduction and State of the Art

Combining human adaptability and problem solving capability with robot speed, precision and indefatigability will definitely make robots more and more interesting for those industry sectors and enterprises where manual work is still predominant. As a matter of fact, not only human workers will be relieved from fatiguing and/or alienating tasks, but also the flexibility of industry environments will be dramatically increased, since the presence of a human worker decreases the need of structuring the working environment.

However, safety issues arise from the idea itself of HRI: industrial robots normally operate in closed-ended environments delimited by fences and gates (see Fig. 1.1), where direct interaction takes place only when the robot is not operating. In an HRI scenario human operators and robots must collaborate in close proximity, thus requiring to remove all these safety barriers (see Fig. 1.2).

Figure 1.1: Traditional industrial robotic cell with safety barriers.

Nevertheless, despite the lack of physical separation of the workspace, safety can be guaranteed by equipping the robot control system with advanced safety-oriented functionalities. Moreover, even the last regulations for using robots in industrial environments [3] impose strict requirements like for instance a maximum end-effector linear velocity of 250 mm/s.

In the scientific literature the problem of guaranteeing human workers’ safety during HRI has been tackled from many different perspectives: robot intrinsically designed for safety, collision avoidance strategies, collision detection and reaction algorithms, and, finally, sensing systems able to detect, track and predict the motion of human
1.1 Safe HRI - Assessing Interaction Safety

Assessing safety during man-machine interaction consists in evaluating the severity of a possible impact between a human being and a moving industrial robot or, in other words, determining how dangerous the impact itself will be for the human involved. Initially, this problem has been addressed in literature using well-established methodologies developed in the automotive industry, like the Gadd Severity Index (GSI) [46] or the Head Injury Criterion (HIC) [133]. However, both the GSI and the HIC have been developed to deal with head injury, while in an industrial environment it is likely that a human-robot collision involves the worker’s arm, his/her leg or thorax.

More recently, new attempts have been carried out to evaluate the severity of human-robot collisions using standard automotive crash-testing equipment and protocols. In [51,52] a large number of impact experiments and a wide basis of impact testing results are presented, mainly considering the contact forces and the acceleration of the human body parts involved in the collision.

In [58] and [59] Ikuta, Ishii and Nokata describe the first systematic severity evaluation methodology specifically developed for human-robot cooperation. Nevertheless, it is not explained how to implement this methodology to allow real-time severity calculation. In [56] the “Impact Potential” is defined as the maximum impact force with which a mechanical moving system can hit a still obstacle, while in [71] and [70] a methodology is proposed that takes advantage of the information regarding severity in order to influence path planning and/or trajectory generation in real-time.

Finally, in [86,94] the authors provide the results of several numerical simulations of impact between a generic mass and a specific finite element model of the human being, named the “THUMS” model [14,79]. In [17,107,109,110] a safety-oriented control architecture is proposed that relies on this injury knowledge. A different approach to embed injury knowledge into control algorithms and strategies is presented in [53].

1.2 Safe HRI - Control

In the last years several relevant contribution addressing the problem of developing suitable control strategies and architectures for safe HRI have been proposed. From a taxonomical point of view, these approaches can be divided in two large families: collision avoidance and collision detection and reaction.
1.2.1 Collision Avoidance

Collision avoidance strategies have been largely investigated in the scientific literature since avoiding impacts between a human worker and a robot is the most basic method to ensure safety during HRI. Indeed, the most recent safety standards [2–5] establish speed and separation monitoring criteria, according to which a minimum separation distance (possibly depending on Tool Center Point velocity and/or payload) must be kept between an industrial manipulator and a human worker. Moreover, it is worth noticing that human operators working inside the robot workspace could continuously modify the robot working environment (for instance by adding and/or removing objects and tools). Consequently, the robot control system should be able to adapt the robot trajectory to the changing conditions in order to accomplish its task both efficiently and safely.

A fundamental contribution to the design of real-time collision avoidance strategies is represented by [65], where the artificial potential field method is proposed. Virtual repulsive forces are applied both at the operational space level (to guide the robot towards the goal while avoiding obstacles) and at the joint level (to avoid kinematic limitations). A similar approach developed for a mobile robot is presented in [21, 22]. The artificial potential field method is combined with real-time sensor-based obstacle detection to allow the mobile robot to reach its goal while navigating inside the partially unknown environment.

In order to combine real-time reaction to obstacles with global path achievement, the concept of elastic bands was introduced [106]. The main idea behind this approach is to treat the global path as an elastic which can be deformed by obstacles. In this way the robot trajectory can be adapted in presence of obstacles without altering its goal. A different approach is represented by the concept of evasive motion in presence of obstacles, originally introduced in [44].

The use of online trajectory generation algorithms for the design of reactive control strategies such as collision avoidance has recently gained significant attention. In [45] an obstacle avoidance strategy based on measurements of a depth sensor, which generates repulsive forces from robot-obstacles distances, and computes evasive motions using the online trajectory generation algorithms presented in [67, 68], was proposed.

A relevant contribution in the field of collision avoidance strategies is surely represented by the concept of Danger Field (DF), originally introduced in [72] and further refined in [73, 74]. The DF is an artificial field that can be used to determine the danger level of a point in close proximity to the robot. This assessment is based on the idea that the robot, rather than the human, is a source of danger. If we consider for instance a virtual impedance control scheme [132], the DF can be used as the virtual force that, once converted into a variation of the joint position and velocity references, causes the manipulator to keep a certain distance from the identified obstacles. In [25, 31], the authors use the DF-based control scheme previously described in combination with an exteroceptive distributed LED-based distance sensor. In this way it is possible to perform collision avoidance on the basis of several distances measured from several points on the manipulator. Other DF-based approaches are proposed in [139, 140], where the authors present a passivity based control algorithm for safe human-robot coexistence that produces slower motion profiles when the human/robot distance decreases. Finally, a dual version of the DF, named “Safety Field”, that measures the level of safety a point
1.3. Safe HRI - Detecting and Predicting Human Motion

in close proximity to the robot is proposed in [100]. Unfortunately, the main drawback of the large majority of the aforementioned control strategies consists in their incompatibility with path constraints, i.e. the need for the robot to minimize the deviation from the pre-programmed trajectory imposed by the task. As a matter of fact, in case the robot is executing a spray painting or arc welding task, a large modification of the pre-programmed trajectory could be as dangerous as a collision with a human operator (for both the worker and the robot). Moreover, large deviations from the pre-defined path necessarily disrupt the robot productivity. For these reasons, several contributions [85, 136, 138] have recently focused on the development of control strategies that are compliant with respect to both the aforementioned safety standards and these path constraints. In particular in [136, 138] the authors propose a simple, yet very effective kinematic scaling strategy that preserves the geometry of the pre-programmed path, while satisfying the minimum separation distance criterion.

1.2.2 Collision Detection and Reaction

Another possible strategy to ensure safety during HRI consists in collision detection and reaction. This control strategy is actually based on a two-fold approach: at first the control system must be able to detect in real-time the occurrence of a collision between the robot and a human being and then it actively controls the robot impact behaviour, so as to guarantee a safe collision for the human operator.

The problem of sensing man-machine collisions without resorting to dedicated sensors has been addressed in [36] and [47]. More in depth, in [36] a simulation study is presented on the evaluation of contact forces for a planar manipulator, without a force sensing system. The same methodology for sensorless estimation of the interaction forces is then used in [47], where the authors realize a control system for pHRI on the basis of joint position signals, motor current signals, and a rough dynamical model of the robot. Moreover, in [34] the concept developed in [36] was extended in order to develop collision reaction strategies. A complete overview of the manipulator adopted in the previously cited work, the DLR LWR, can be found in [11].

The very same robot is used in [50] as a testbed for different collision detection and reaction strategies, that are experimentally validated through several impact tests with humans. The possibility to guarantee safe collisions even for high operating velocities was demonstrated, and intentional contacts could be discriminated from undesired collisions, thus allowing a human operator to control the robot by simply touching it.

More recently, several developments of these collision detection and reaction strategies have been proposed. For instance, in [54] a solution based on a vision sensor is presented, while in [53] a collision detection and reaction strategy based on injury knowledge is presented and experimentally tested.

1.3 Safe HRI - Detecting and Predicting Human Motion

In a HRI scenario the need to remove physical barriers necessarily entails a lack of artificially imposed safety that can be compensated for by endowing the robot control systems with more advanced safety-oriented functionalities. For instance, if the robot control system is able to detect, track and predict the motion of human workers inside the robotic cell, it will be able to choose the most suitable control strategy (speed re-
duction, protective stop, and/or trajectory modification) in order to avoid and/or resolve a possibly dangerous situation.

In the scientific literature, two distinct problems can be clearly identified: Human motion Detection and Tracking (HDT) and human motion prediction, which is also known as Intention Estimation (IE). HDT consists in detecting the presence of one or more human beings inside the supervised environment and to track their motion (in terms of position and, if possible, velocity) on the basis of a series of consecutive “descriptions” of the scene provided by one or more sensors. On the other hand, IE consists in predicting the destinations of a human walking trajectory in a structured environment on the basis of the tracked positions and velocities.

Although the problems of HDT and IE have been addressed in several contexts, we here consider only approaches proposed in the field of industrial robotics and automation. Techniques to perform HDT in an industrial environment using single camera or multiple cameras are described in [39, 117], while high-visibility industrial clothing detection strategies based on RGB and IR cameras have been proposed in [89] and [90]. Approaches based on pressure-sensitive sensors mounted on the floor have been proposed as well, like for instance [93]. Finally examples of HDT relying on RGB-D sensor can be found in [91] and [92].

Regarding IE, in [70, 71] techniques combining vision and psychological signal measurement for human motion estimation during HRI are presented, while [13] describe a system for predicting the probability of an accident in a HRI industrial scenario based on a dynamic stochastic model of human motion. A different approach to the problem of IE is presented in [15] and [129], where human intentions are defined as complex manipulation operations, without motion prediction. A similar strategy, based on Hidden Markov Models, has been adopted in [66] using a vision system equipped with laser range finders. A combined approach to HDT and IE is presented in [19], where the final destination of a human walking trajectory is estimated using commercial RGB surveillance cameras. More recently, a similar approach, developed for unstructured environments, has been proposed in [78], while a different technique based on RGB-D and LIDAR data is presented in [135].

Finally, also the usage of human detection and occupancy prediction strategies for control purposes has been investigated, like for instance in [16, 69, 83].
Part I

Perception: Modelling and Tacking
Human Motion via Sensor Fusion
CHAPTER 2

Modelling Human Motion

2.1 Introduction

In order to track the motion of a human worker inside a robotic cell, it is necessary to model human kinematics and to define a kinematic configuration in terms of “human joint” positions, velocities and, if possible, accelerations.

Moreover, this kinematic model should be simple enough to allow real-time execution of the motion tracking (and prediction) algorithms that will be described in the following Chapters of this thesis.

More in depth, this Chapter introduces three different kinematic models describing human motion. The first model describes human walking kinematics using a non-holonomic approach. The second model consists in an extension of the first one obtained by removing the non-holonomic constraint in order to take into account also lateral walking. Finally, a third model is introduced that not only describes walking kinematics, but also models the motion of the upper limbs (in terms of torso, head and arms).

2.2 Non-Holonomic Model of Human Walking Kinematics

To properly describe the kinematics of human walking it is convenient to approximate the human being with a single point moving in a 2D environment. Consequently, by fixing a world-base Cartesian frame on the ground plane, the kinematic configuration of a walking human can be described as:

\[
\begin{align*}
\mathbf{p} &= [x, y, \theta] \\
\mathbf{\dot{p}} &= [\dot{x}, \dot{y}, \omega]
\end{align*}
\]  

(2.1)

where:
Chapter 2. Modelling Human Motion

- $x$ is the coordinate with respect to the world base frame X-axis;
- $y$ is the coordinate with respect to the world base frame Y-axis;
- $\theta$ is the angle formed between the tangent to the walking path and the world base frame X-axis;
- $\dot{x}$ is the velocity along the world base frame X-axis;
- $\dot{y}$ is the velocity along the world base frame Y-axis;
- $\omega$ is the angular velocity;

However, according to several scientific contributions available in the literature, like for instance [12, 19], human locomotion can be modelled as non-holonomic, since the linear velocities $\dot{x}$ and $\dot{y}$ are actually coupled to the body orientation. Consequently, the kinematics of human walking can be described using the following unicycle-based non-holonomic model:

\[
\begin{align*}
\dot{x} &= v \cos(\theta) \\
\dot{y} &= v \sin(\theta) \\
\dot{\theta} &= \omega \\
\dot{v} &= a_l \\
\dot{\omega} &= a_\omega 
\end{align*}
\] (2.2)

where $v$ is the tangential velocity directed along $\theta$.

Finally, according to the assumption that both the linear velocity $v$ and the angular velocity $\omega$ are piece-wise constant (see again [12]), the linear and angular accelerations, $a_l$ and $a_\omega$, can be modelled as two independent and uncorrelated Gaussian white noises acting respectively on $v$ and $\omega$:

\[
a_l \sim N(0, 1) \\
a_\omega \sim N(0, 1)
\] (2.3)

A graphical representation of the model is given in Figure 2.1.

2.3 Holonomic Model of Human Walking Kinematics

In order to account for lateral walking, it is necessary to extend model (2.2) by removing the non-holomic constraint. A possible solution to this problem is proposed in [87], where a second linear velocity term is introduced that is orthogonal with respect to the body orientation $\theta$.

By introducing the orthogonal velocity term $v_\perp$, the following holonomic model is obtained:

\[
\begin{align*}
\dot{x} &= v \cos \theta - v_\perp \sin \theta \\
\dot{y} &= v \sin \theta + v_\perp \cos \theta \\
\dot{\theta} &= \omega \\
\dot{v} &= a_l \\
\dot{\omega} &= a_\omega 
\end{align*}
\] (2.4)
2.4. Complete Kinematic Model of Human Motion

![Diagram of non-holonomic kinematic model describing human walking motion.]

**Figure 2.1:** Representation of the non-holonomic kinematic model describing human walking motion.

It is worth noticing that model (2.4) can be expressed in terms of a linear formulation by simply considering fully de-coupled linear velocities $v_x$ and $v_y$:

\[
\begin{aligned}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{\theta} &= \omega \\
\dot{\nu} &= \alpha_l \\
\dot{\omega} &= \alpha_w
\end{aligned}
\]  

(2.5)

As a matter of fact, the new formulation is completely equivalent to the previous one, since a univocal correspondence exists between the two different sets of velocities:

\[
\begin{bmatrix}
v_x \\
v_y
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
v \\
v_{\perp}
\end{bmatrix}
\]

\[
\begin{bmatrix}
v \\
v_{\perp}
\end{bmatrix} = \begin{bmatrix}
\cos \theta & \sin \theta \\
-\sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
v_x \\
v_y
\end{bmatrix}
\]

To summarize, the kinematic configuration of a walking human being can still be described in terms of pose and velocity vectors $p$ and $\dot{p}$ (see (2.1)), but, differently from the non-holonomic case, velocities components $\dot{x}$ and $\dot{y}$ can be parameterized either in terms of forward and orthogonal components, according to (2.4), or in terms of de-coupled Cartesian components, according to (2.5).

### 2.4 Complete Kinematic Model of Human Motion

Up to now only the walking kinematics have been considered. To properly guarantee human workers’ safety during the execution of cooperative tasks, it is convenient to take into account also the motion of the human upper limbs: torso, head and arms. To this purpose, this Section first introduces a kinematic model describing the motion of the
human arm (from the shoulder up to the wrist) and then presents a complete kinematic model obtained by combining together walking and arm kinematics.

### 2.4.1 Human Arm Kinematics

In order to describe the human arm kinematics, we make use of the kinematic model originally introduced in [33] and further refined in [137], see Figure 2.2. As a matter of fact, only the first 4 DOFs or the original model are considered since the human arm is roughly approximated with two segments: the first one (upper arm) connecting the shoulder to the elbow and the second one (lower arm) going from the elbow to the wrist.

![Kinematic model of the human (right) arm and torso flexion/extension angle \( \rho \).]

Finally, on the basis of this approximation, the kinematic model of the human arm can be formulated in terms of four integrators for each joint angle:

\[
\begin{align*}
\frac{d}{dt} \alpha &= \dot{\alpha} \\
\frac{d}{dt} \dot{\alpha} &= \ddot{\alpha} \\
\frac{d}{dt} \ddot{\alpha} &= \ldots \\
\frac{d}{dt} \ldots &= \eta
\end{align*}
\]

where

\[
\alpha = [\alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4]
\]

is the vector containing the joint angles represented in Figure 2.2 while, under the hypothesis that the jerk is piece-wise constant, \( \eta \) components can be modelled as independent and uncorrelated Gaussian white noises:

\[
\eta \sim N(0, I)
\]
2.4. Complete Kinematic Model of Human Motion

2.4.2 Complete Kinematic Model

Combining together walking and arm kinematics, a model is obtained that describes the full motion of the upper human body. More in depth this new kinematic model is composed of a 3-DOF base moving on the ground plane, one lumped 1-DOF (flexion/extension) torso, a head (fixed) and two 4-DOF arms. A graphical representation of such kinematic model is given in Figure 2.3.

![Figure 2.3: Complete human kinematic model: DOFs, frames and bodies](image)

Given the new model, the kinematic configuration $p$ can be extended in the following way:

$$p = [x \ y \ \theta \ \rho \ \alpha_{right} \ \alpha_{left}]^T$$

(2.9)

where:

- $x$, $y$ and $\theta$ corresponds to the walking kinematic configuration;
- $\rho$ is the torso bending angle;
- $\alpha_{right}$ and $\alpha_{left}$ are respectively the joint angles vectors for the right arm and for the left arm.

Finally, since models (2.5) and (2.6) are both linear and all the status components are completely de-coupled, also the complete kinematic model can be formalized by means of a chain of four integrators for each joint variable:

$$\begin{cases}
\frac{d}{dt} p = \dot{p} \\
\frac{d}{dt} \dot{p} = \ddot{p} \\
\frac{d}{dt} \ddot{p} = \dddot{p} \\
\frac{d}{dt} \dddot{p} = \eta
\end{cases}$$

(2.10)

It is worth noticing that an analogous representation of this kinematic model can be given in terms of a set of 3D points composing a rough scheme of the human skeleton, see Figure 2.4. Since only the motion of the upper part of the human body is considered, the points of interest are: Thorax ($T$), Neck ($N$), Head ($H$), Left Shoulder ($LS$), Right Shoulder ($RS$), Left Elbow ($LE$), Right Elbow ($RE$), Left Wrist ($LW$), and Right Wrist ($RW$).

This skeletal representation is particularly useful because several well-known algorithms (for instance [9]) can be used to extract skeletal points from a depth map.
acquired by either a depth sensor or an RGB-D camera. Consequently, it is convenient to consider the skeletal representation as the output of model (2.10) and to formulate the corresponding forward kinematic calculations.

On the other hand, in order to convert a generic skeletal representation into the corresponding kinematic configuration \( p \), it is necessary to formalize also inverse kinematic calculations. To this purpose, details regarding forward and inverse human kinematics are provided in Appendix A and Appendix B respectively.
CHAPTER 3

Tracking Walking Motion with Multiple RGB Cameras

3.1 Introduction

In this Chapter we propose an algorithm that performs HDT on the basis of a sensor fusion strategy that merges the information contained in several RGB images. Starting from a scene monitored by multiple RGB surveillance cameras, the corresponding images are acquired and warped together to create a unique combined image. Background/Foreground Segmentation (from now on “BG/FG Segmentation”) is applied to the combined image to detect human workers. K-dimensional trees data structures (from now on “k-d trees”) are then used to efficiently update in time the information regarding detected humans’ silhouettes. Finally multiple Particle Filters perform the tracking functionality.

With respect to the state of the art, the main novelties of this approach can be summarized as follows:

- **RGB Sensor Fusion**: multiple images simultaneously acquired from different surveillance cameras are warped together to obtain a unique combined image describing the whole supervised environment;

- **Abstraction from Physical Sensors**: Image Fusion completely decouples the HDT processing pipeline from physical sensors. Though multiple physical cameras are used, the HDT pipeline “sees” only one logical sensor from which the combined image is acquired;

- **K-d Trees**: the use of k-d trees provides an efficient and elegant solution to the problem of updating in time the information regarding detected human workers.
Chapter 3. Tracking Walking Motion with Multiple RGB Cameras

3.2 Multiple RGB Images Fusion

The fusion of images acquired from $R$ different cameras relies on calibration of every vision sensor. For every available surveillance camera both the intrinsic and the extrinsic calibration matrices, respectively $I_r$ and $E_r$, are identified. While $I_r$ maps the $r$-th camera Cartesian frame to the $r$-th camera pixel coordinate frame, $E_r$ maps 3D points expressed in the world-base Cartesian frame to the $r$-th camera Cartesian frame, as sketched in Figure 3.1. Moreover, also the radial and tangential distortion coefficients $d_r$ are identified:

$$d_r = \{k_{1r}, k_{2r}, p_{1r}, p_{2r}, k_{3r}\} \quad (3.1)$$

In order to obtain a unique image describing the whole supervised environment, images acquired from several surveillance cameras must be compensated for distortion effects (using $d_r$) and then they must be warped together. Image warping consist in mapping every pixel $P_o$ in the original image to a different pixel $P_w$ through a warping matrix $W$:

$$P_w = [u_w, v_w]^T = W P_o = W [u_o, v_o]^T \quad (3.2)$$

A reference camera is selected so that the coordinate transform between the world frame and the combined image pixel coordinate frame (and viceversa) can be described by the extrinsic and intrinsic calibration matrices of the reference camera. At this point images must be warped together in such a way that pixels describing corresponding points on the floor plane can be exactly overlapped. To obtain this result the homography matrix $H_r$ of the $r$-th (non-reference) camera image plane with respect to the reference camera image plane must be identified, with both image planes corresponding to the scene floor.

Since $H_r$ is a 3x3 matrix defined up to a scale factor, the problem of identifying its elements can be solved considering four corresponding points between the reference camera image and the $r$-th camera image. In order to find the homography matrices that map the $r$-th camera image plane to the reference camera image plane while preserving the scene floor, four different points $P_i^W$ belonging to the scene floor ($z_i^W = 0$) are

Figure 3.1: Example of a setup including two calibrated cameras and corresponding image planes.
3.3. Human Motion Detection via BG/FG Segmentation

Having warped together all the images acquired by the different RGB surveillance cameras, it is possible to perform BG/FG Segmentation on the combined image in order to detect human beings entering the robotic cell or walking inside it.

The BG/FG Segmentation algorithm adopted in this work is part of the OpenCV library [23] and it is described in [141] and [142]. It consists in an efficient adaptive algorithm that performs background subtraction at pixel level and that relies on Gaussian mixture probability density. It also offers the possibility to trigger online background update. As shown in Figure 3.3, the algorithm’s output consists in two different images:

- **Foreground Mask**: binary image whose pixels are white (black) if the corresponding pixel of the input image belongs to the foreground (background);

Finally, to determine the r-th homography matrix $H_r$ so that:

$$ p_i^{ref} = H_r p_i^r \quad i \in [1, 4] $$

the procedure described in [55] is followed. Since surveillance cameras are fixed, the identification of homography matrices $H_r$ can be performed entirely offline so that the warping stage of the HDT pipeline simply warps every acquired image using the corresponding homography and overlaps the warped images to obtain the combined image, as shown in Figure 3.2.

---

**Figure 3.2**: Example of multiple camera image fusion. Left: image acquired from camera #01. Middle: image acquired from camera #02. Right: combined image resulting from image fusion.
Chapter 3. Tracking Walking Motion with Multiple RGB Cameras

Figure 3.3: Example of Single Camera BG/FG Segmentation. Left: input image. Middle: foreground mask. Right: foreground image.

- **Foreground Image**: RGB colour image containing only the foreground pixels. It is obtained by simply applying the binary mask to the input image.

Moreover the algorithm provides a shadow detection functionality [62] that allows to perform object detection while discarding shadows of segmented objects.

After BG/FG Segmentation, the Foreground Mask is further processed performing “image opening”, i.e. applying in sequence an erosion and a dilation kernel [24]. The main advantage brought by applying image opening to the Foreground Mask consists in removing image noise (especially isolated pixels erroneously classified as foreground) while preserving large foreground areas.

At this point the contours of the connected components in the Foreground Mask image are extracted and a last “plausibility check” is introduced. As a matter of fact it is reasonable to assume that foreground areas must be large enough to represent a human being walking inside the scene. Consequently if a foreground area’s surface (measured in square pixels) is smaller than an experimentally determined threshold value, the object is considered a false positive and it is discarded. Otherwise it is actually classified as a detected human worker.

3.4 Using K-d Trees to Update detected humans

The main problem related to the output of BG/FG Segmentation stage is to determine for every foreground area detected at time step $i$, the corresponding area inside the foreground image computed at time step $i - 1$. As a matter of fact a continuous update of the contours of the silhouette describing the same human being across a series of consecutive time instants is fundamental to feed the different particle filters with coherent information (see Section 3.5). To solve this issue the information regarding detected humans is structured in k-d trees, but first the following “plausibility hypotheses” are considered:

- humans cannot suddenly appear inside the robotic cell or either disappear from it;
- humans can enter/exit the cell only through one or more access areas (i.e. gates, doors, ecc.);
- it is likely that the position of the same human being will undergo limited variations from one time step to the following one.
3.4. Using K-d Trees to Update detected humans

Thanks to these hypotheses the problem of erroneous robot detection can be easily overcome: even if a moving industrial robot is detected by BG/FG Segmentation, it won’t be considered as a human being. While [19] tackled this problem by masking out the entire robot’s workspace inside the Foreground Image, the approach here presented does not require this further image-processing step and avoids large parts of the acquired image to be ignored, thus resulting simpler, more efficient and more effective.

A k-d tree is a space-partitioning data structure that allows to organize points belonging to a k-dimensional space [88] in a binary tree. Considering a variant of k-d trees, where actual points can be stored only in the leaf nodes, every non-leaf node represents a splitting hyperplane that divides the k-d space into two half-spaces. Points to the left of this hyperplane are represented by the left subtree of that node and points right of the hyperplane are represented by the right subtree. An example of a 2-dimensional tree is shown in Figure 3.4.

Using k-d trees the problem of updating online the information regarding detected humans can be elegantly formalized as the identification of couples of nearest neighbours between two different 2-d trees: one, named $FG_{prev}$, containing the Center-of-Gravity (CoG) of the human silhouettes detected on the combined image at the previous time step and another one, named $FG_{now}$, containing the CoG of the foreground areas segmented at the current time step. Algorithm 1 explains how this nearest neighbour search can be performed.

**Algorithm 1** Data Association Algorithm based on K-d Trees

1: \textbf{for all } $f_{now} \in FG_{now}$ \textbf{do}
2: \hspace{1em} $f_{prev} \leftarrow \text{nearest}(FG_{prev}, f_{now})$
3: \hspace{1em} \textbf{if } $f_{now} == \text{nearest}(FG_{now}, f_{prev})$ \textbf{then}
4: \hspace{2em} add $(f_{prev}, f_{now})$ to results
5: \hspace{1em} \textbf{end if}
6: \textbf{end for}

Not only this solution is very elegant, but it is also very efficient. If we suppose that both sets contain $n$ elements, the time complexity of building the corresponding 2-d trees and searching for couples of nearest neighbours is $O(n \log n)$, while the time complexity of performing distance checks between every possible pair of elements would
Chapter 3. Tracking Walking Motion with Multiple RGB Cameras

Figure 3.5: Left: human worker detected silhouette (grey), pose and circumscribed box according to the explained motion model. Middle: graphic output of a particle motion model state considering a single camera. Right: pixel determining the particle’s value considering a single camera.

be $O(n^2)$.

After identifying the couples of nearest neighbours between the two 2-d trees, two last checks are performed: every foreground area detected near an entrance zone, but not associated to a previously detected human, is considered as a new human entering the cell, and every detected human no longer associated to a foreground area is considered as a person that left the cell.

3.5 Human Walking Motion Tracking via Particle Filtering

The tracking strategy here adopted is inspired by the one proposed in [19]. After BG/FG Segmentation and foreground areas update, human workers are tracked by a series of particle filters that rely on the kinematic model (2.2).

Moreover, given the following assumptions:

- the scene consists of a flat ground plane on which humans walk around;
- a human worker does not walk sideways;
- human workers and industrial robots are the unique moving objects in the camera field of view, but, since robots do not enter the scene from the entrance zones, their detection is automatically avoided.

a simple yet effective observation model has been chosen. By circumscribing a rectangular box around a walking person, we are able to describe his/her walking motion in terms of translation on the floor and rotation around the vertical axis crossing the base in its centre (see Figure 3.5a). In other words we observe the walking motion by considering the volumetric occupancy of the moving person.

In our scenario deterministic evaluation of the human motion state is not possible mainly because of significant measurement noise. Moreover, analytical calculation of the motion model output in terms of multiple rectangular boxes (each one projected according to a single camera point of view) is not feasible.

Consequently, our tracking strategy consists in assigning to every detected human a probability distribution over the possible states in the form of a set weighted particles, propagated in time according to the chosen kinematic model. In this way, for every moving worker, multiple virtual representations are generated and his/her motion state is estimated by selecting the particle whose representation best matches the measured
3.5. Human Walking Motion Tracking via Particle Filtering

foreground. At any time instant \( i \) the motion state of a single walking human being is composed by a set of \( N \) particles:

\[
Q_i = \left\{ q_i^{(j)} \mid j = 1, \ldots, N \right\}
\]  

(3.9)

where every particle represents a possible motion state configuration:

\[
q_i^{(j)} = \left( x_i^{(j)}, y_i^{(j)}, \theta_i^{(j)}, v_i^{(j)}, \omega_i^{(j)} \right)
\]  

(3.10)

The initial distribution can be considered known a priori and it corresponds to a scene without moving workers. Right after instantiation, every filter is considered “inactive” and its particle set is initialised via uniform random sampling inside a subspace of the model state space defined around the entrance areas. As soon as a new human is detected (see Section 3.3), an “inactive” filter is assigned the corresponding foreground area and thus, it becomes “active”.

While receiving continuously updated information regarding the foreground area it is tracking (see Section 3.4), the particle filter keeps propagating particles. Defining \( f \) the transfer function corresponding to the discrete motion model, the particle set propagation from time step \( i \) to time step \( i + 1 \) can be simply defined as:

\[
Q_{i+1} = \left\{ q_{i+1}^{(j)} = f(q_i^{(j)}) \mid j = 1, \ldots, N \right\}
\]  

(3.11)

The probability that each particle corresponds to the actual state of the walking human is computed on the basis of two binary images: the first contains the foreground area describing the human appearance (see Figure 3.5c), the latter represents the appearance of the particle itself. The box vertices are computed on the basis of the particle and projected in every camera perspective. A binary image is created where non-zero pixels belong to the superposition of the box projections (see Figure 3.5b). The probability measure is finally obtained by counting the number of non-zero pixels contained in the logic AND of the two binary images, as depicted in Figure 3.5c. After evaluation, particles probabilities are normalised using their sum as a normalizing factor:

\[
\bar{\alpha}_i = \sum_{j=1}^{N} \alpha_i^{(j)}
\]  

(3.12)

\[
\alpha_i^{(j)} \leftarrow \frac{\alpha_i^{(j)}}{\bar{\alpha}_i}, \forall j \in [1, \ldots, N]
\]  

(3.13)

To update the estimate of the human state, a best particle is extracted from the filter’s particle set. Particles are sorted in descending order with respect to probability values and the best particle is computed as the weighted average of the best \( n \) particles (i.e. the first \( n \) particles within the sorted set).

The re-sampling stage realizes a balance between exploitation and exploration. Particles being the nearest with respect to the actual state of the walking human are mixed to new particles obtained via uniform random sampling inside a subspace of the model state space defined around the best particle previously extracted.

Finally, when the tracked human being exits the supervised environment, the filter goes back to the “inactive” state and waits until it is assigned another human to track.
Chapter 3. Tracking Walking Motion with Multiple RGB Cameras

Figure 3.6: The experimental setup comprising the two AXIS fish-eye surveillance cameras, the PC running the HDT application and the Ethernet connection between the different components. The entrance area is highlighted in yellow and the two walking paths and the corresponding destination areas are highlighted respectively in black and grey.

The design and implementation of the filtering stage has been realized so that the number $M$ of particle filters running in parallel, the dimension $N$ of each filter particle set, the number $n$ of particles to average during best particle extraction and the percentage of maintained particles can be configured by the user prior to the actual execution.

3.6 Experimental Validation

The experimental setup depicted in Figure 3.6 includes three industrial robots (an ABB IRB140, an ABB FRIDA prototype robot and a COMAU Smart-Six) and two AXIS 212 PTZ RGB Network cameras connected via Ethernet to the PC hosting the HDT application. Walking paths and destination areas have been drawn on the floor in order to provide ground-truth for the experiments described in the following. In addition to the results here presented, the experimental validation is also documented by video 104.

The particle filters’ parametrization adopted during the experiments was the following one:

- 3 particle filters running in parallel;
- 250 particles composing each particle set;
- best particle extraction via weighted average of the 1% best particles;
- 20% best particles maintained during re-sampling.

3.6.1 Experiment #1: Single Person Detection and Tracking

In the first experiment a human worker enters the robotic cell and reaches the destination area #1 (the one coloured in black in Figure 3.6) following the path defined by the black dotted lines drawn on the floor. Figure 3.9 contains a series of screenshots of the experiment showing that the human worker is correctly tracked. Considering the drawn black path as ground-truth, Figure 3.7 demonstrates the effectiveness of our approach to HDT by showing that the best particle two-dimensional position (i.e. the human worker
3.6. Experimental Validation

Figure 3.7: Graph showing that the human position estimate computed by the HDT System (dashed red) is always included inside the path drawn on the ground (dashed black).

trajectory estimated by the particle filter) is always included in the area delimited by the black dotted lines.

Figure 3.9 contains a series of screenshots of the experiment showing that the human worker is correctly tracked. Considering the drawn black path as ground-truth, Figure 3.7 demonstrates the effectiveness of our approach to HDT by showing that the best particle two-dimensional position (i.e. the human worker trajectory estimated by the particle filter) is always included in the area delimited by the black dotted lines.

3.6.2 Experiment #2: Multiple Person Detection and Tracking

During the second experiment two human workers enter the robotic cell. The first directs himself towards destination area #1, following the path drawn in black, while the latter reaches destination area #2, following the path defined by grey dotted lines. Figure 3.10 contains a series of screenshots of the experiment showing that the human worker is correctly tracked, while Figure 3.8 shows once again that the trajectories followed by the two human workers estimated by the particle filters are always included in the area delimited by the drawn dotted lines.

3.6.3 Experiment #3: Single Person Detection and Tracking with Moving Robot

The third experiment replicates the same protocol defined for experiment #1, but while the human worker walks towards destination area #1, the COMAU Smart-Six industrial robot moves along a programmed trajectory. The screenshots contained in Figure 3.11 show that, while the human worker is correctly tracked, the robot motion is completely ignored, demonstrating the effectiveness of our approach also in terms of erroneous robot detection avoidance.
Chapter 3. Tracking Walking Motion with Multiple RGB Cameras

Figure 3.8: Graph showing that the human worker position estimates computed by the HDT System (dashed red and dashed green) are always included inside the corresponding paths drawn on the ground (respectively dashed black and dashed grey).

Figure 3.9: Screenshots taken from Experiment #1 showing that a single human worker is correctly tracked by the HDT System.
3.6. Experimental Validation

Figure 3.10: Screenshots taken from Experiment #2. The first human worker is tracked by a red box, while the other is tracked by a green box.
Figure 3.11: Screenshots taken from Experiment #3 showing that a single human worker is correctly tracked by the HDT System while the robot motion is ignored.
CHAPTER 4

Tracking Full Upper Body Motion with Multiple Depth Sensors

4.1 Introduction

In this Chapter we provide a detailed description of the sensor fusion algorithm that allows to acquire multiple skeletal tracking point sets from several depth sensor and merge this information in order to obtain a unique estimation of the kinematic state of the human worker, according to the model described in Chapter 2.

More in depth, at first we present a sensor fusion strategy based on a Linear Kalman Filter (LKF) [63] and then we discuss how the kinematic state estimation provided by the filter can be modified in order to satisfy the following bounds:

- \( p_{\text{inf}} \) and \( p_{\text{sup}} \): lower and upper bounds on human joint positions;
- \( \dot{p}_{\text{inf}} \) and \( \dot{p}_{\text{sup}} \): lower and upper bounds on human joint velocities;
- \( \ddot{p}_{\text{inf}} \) and \( \ddot{p}_{\text{sup}} \): lower and upper bounds on human joint accelerations;

4.2 Kalman filter based multiple depth sensor fusion

In order to merge together the different measurements acquired from all the available depth sensors, we make use of a LKF. The adopted process model consists in a discretized version of model (2.10) and it is composed by a chain of three discrete-time integrators for each joint variable. Moreover, the filter state vector \( s \) (that already contains joint positions, velocities, accelerations and jerks) is further extended to take into account also the parameters of the human kinematic model that are specific for each individual (i.e. the distances between the skeletal points acquired by the depth sensors,
see Appendix for further details). These parameters, labelled as $\pi$ in equation (4.2), can be estimated by simply imposing a constant dynamics. In detail:

$$s_{k+1} = Fs_k + \eta_k \quad (4.1)$$

$$s_k = \begin{bmatrix} p_k \\ \dot{p}_k \\ \ddot{p}_k \\ \pi_k \end{bmatrix}, \quad F = \begin{bmatrix} I & \Delta tI & \frac{\Delta t^2}{2}I & \frac{\Delta t^3}{6}I & 0 \\ 0 & I & \Delta tI & \frac{\Delta t^2}{2}I & 0 \\ 0 & 0 & I & \Delta tI & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \quad (4.2)$$

In particular:

$$\eta_k \sim \mathcal{N} (0, G) \quad (4.3)$$

models the process noise, whose covariance matrix $G$ can be parametrized as follows. For each block of the state vector we consider the corresponding first truncated element of the Taylor approximations contained in matrix $F$ (see equation (4.2)) and we obtain:

$$G = \begin{bmatrix} \sigma_p^2 \frac{\Delta t^4}{24}I & 0 & 0 & 0 & 0 \\ 0 & \sigma_p^2 \frac{\Delta t^3}{6}I & 0 & 0 & 0 \\ 0 & 0 & \sigma_p^2 \frac{\Delta t^2}{2}I & 0 & 0 \\ 0 & 0 & 0 & \sigma_p^2 \Delta tI & 0 \\ 0 & 0 & 0 & 0 & \sigma_p^2 \end{bmatrix} \quad (4.4)$$

where standard deviations $\sigma_p$, $\sigma_\dot{p}$, $\sigma_{\ddot{p}}$ and $\sigma_{\dddot{p}}$ are tunable parameters.

Regarding the observation model, since the inverse kinematics of the human motion model can be computed in closed-form, we consider as observed output all the joint positions and all the kinematic parameters corresponding to the data acquired from the $n$ available depth sensors. Consequently we consider as observation model the following linear transformation:

$$z_k = Hs_k + \zeta_k \quad (4.5)$$

$$z_k = \begin{bmatrix} p_{1,k} \\ \vdots \\ p_{n,k} \\ \pi_{1,k} \\ \vdots \\ \pi_{n,k} \end{bmatrix}, \quad H = \begin{bmatrix} I & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & I \end{bmatrix} \quad (4.6)$$

where $p_{i,k}$ and $\pi_{i,k}$ represent the set of joint positions and the set of parameters, respectively, computed via inverse kinematics on the basis of the skeletal points acquired by the $i$-th sensor.

Moreover, $\zeta_k$ models the measurement noise, whose covariance matrix $R$ is given by:

$$\zeta_k \sim \mathcal{N} (0, R_k) R_k = \sigma_z^2 I \quad (4.7)$$
where standard deviation $\sigma_z$ can be determined on the basis of the accuracy of the skeletal points acquired by the depth sensors.

A relevant issue in the design of this sensor fusion strategy is the lack of synchronization between the acquisition process and the process implementing the LKF. As a matter of fact, in order to ensure accurate estimation we need to run the LKF at a frequency that usually is much greater than the data acquisition frequency. Consequently the filter will execute several iterations on the basis of the same set of measurements.

A possible solution consists in updating the observation covariance matrix $R_k$ according to the presence (or not) of new measurements, in such a way that each element on the main diagonal of $R_k$ follows a saw tooth shape. More specifically, every time the LKF receives a new set of measurements, the corresponding blocks of matrix $R_k$ are reset to $\sigma_z^2 I$. On the other hand, whenever the same set of measurements is re-used, the corresponding blocks in $R_k$ are updated by adding $\sigma_z^2 I$. In this way we obtain an uncertainty that is reset to a starting value whenever a new measurement arrives and that grows linearly in time between two consecutive measurements.

Clearly, the described LKF produces an estimation of both the kinematic state of the human worker and of his/her kinematic parameters:

\[
\hat{s}_k = \begin{bmatrix}
\hat{p}_k^T & p_k^T & \dot{p}_k^T & \ddot{p}_k^T & \ldots & \hat{\pi}_k^T
\end{bmatrix}^T
\] (4.8)

More specifically, in case at time step $k$ there are no valid measurements available, the filter directly outputs the a-priori estimates:

\[
\hat{s}_{k|k-1} = F\hat{s}_{k-1}
\] (4.9)

\[
P_{k|k-1} = F P_{k-1} F^T + G
\] (4.10)

\[
\hat{s}_k \leftarrow \hat{s}_{k|k-1}
\] (4.11)

\[
P_k \leftarrow P_{k|k-1}
\] (4.12)

while if there is at least one valid measurement, the filter also executes the prediction update and outputs the a-posteriori state estimation:

\[
\hat{y}_k = z_k - H\hat{s}_{k|k-1}
\] (4.13)

\[
K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R_k)^{-1}
\] (4.14)

\[
\hat{s}_{k|k} = \hat{s}_{k|k-1} + K_k \hat{y}_k
\] (4.15)

\[
P_{k|k} = (I - K_k H) P_{k|k-1}
\] (4.16)

\[
\hat{s}_k \leftarrow \hat{s}_{k|k}
\] (4.17)

\[
P_k \leftarrow P_{k|k}
\] (4.18)

Unfortunately, data acquired from depth sensors can suffer from non-valid measurements due to various reasons (occlusions, human workers being too near with respect to the sensor, human workers leaving the sensor field of view, etc.). Since the acquisition process is able to detect these situations, each set of measurements $p_{i,k}$ and $\pi_{i,k}$ is accompanied by a boolean variable, named $valid_i$, that is true if the measurements are valid, and false otherwise.
Chapter 4. Tracking Full Upper Body Motion with Multiple Depth Sensors

Thanks to this boolean flag, during the calculation of the innovation signal $\tilde{y}_k$ in equation (4.13), we set to zero all the innovation components corresponding to a non-valid set of measurements:

$$\forall i \in [1, n], \text{valid}_i = False \implies \tilde{y}_{i,k} \leftarrow 0$$ (4.19)

In this way we prevent non-valid measurements from affecting the a-posteriori state estimate $\hat{s}_{k|k}$, thus ensuring that it is determined only by valid data. On the other hand both stability and correctness property of the LKF are not invalidated by this calculation since it is equivalent to consider a time-varying observation matrix $H_k$ (see equation (4.6)), whose blocks switch between the identity and the null matrix.

4.3 Bounded kinematic state estimation

Since the estimation computed by the LKF $\hat{s}_k$ does not necessarily satisfy the chosen bounds on joint positions ($p_{\text{inf}}$ and $p_{\text{sup}}$), velocities ($\dot{p}_{\text{inf}}$ and $\dot{p}_{\text{sup}}$) and accelerations ($\ddot{p}_{\text{inf}}$ and $\ddot{p}_{\text{sup}}$), we introduce a hierarchy of optimization problems.

At first we find the set of positions that are nearest to the LKF estimation and that satisfy position bounds by solving the following quadratic programming (QP) problem:

$$\min_{\tilde{p}_k} \|\tilde{p}_k - \hat{p}_k\|^2$$ (4.20a)

$$p_{\text{inf}} \leq \tilde{p}_k \leq p_{\text{sup}}$$ (4.20b)

and we update the estimated joint positions as follows:

$$\hat{p}_k \leftarrow \tilde{p}_k$$

Then, we introduce a second QP problem to find the nearest set of velocities with respect to the LKF estimation, that are also inside velocity bounds:

$$\min_{\tilde{p}_k} \|\tilde{p}_k - \dot{\hat{p}}_k\|^2$$ (4.21a)

$$\dot{p}_{\text{inf}} \leq \tilde{p}_k \leq \dot{p}_{\text{sup}}$$ (4.21b)

where bounds $\dot{p}_{\text{inf}}$ and $\dot{p}_{\text{sup}}$ are computed as

$$\dot{p}_{\text{inf}} = \max (\dot{p}_{i,k}^\text{inf}, (\dot{p}_{i,k}^\text{inf} - \dot{\hat{p}}_{i,k}) / \Delta t)$$

$$\dot{p}_{\text{sup}} = \min (\dot{p}_{i,k}^\text{sup}, (\dot{p}_{i,k}^\text{sup} - \dot{\hat{p}}_{i,k}) / \Delta t)$$

in order to satisfy velocity bounds and to ensure that the a-priori position estimate at time step $k + 1$ stays inside position bounds. Once again, we update the estimated joint velocities:

$$\dot{\hat{p}}_k \leftarrow \dot{\tilde{p}}_k$$
Finally, we solve a last QP problem to ensure that also the estimation of joint accelerations satisfies the aforementioned bounds:

$$\min_{\tilde{\dot{p}}_k} \| \ddot{\tilde{p}}_k - \ddot{\dot{p}}_k \|^2$$ (4.22a)

$$\frac{\dot{\omega}_\text{inf}}{p} \leq \ddot{\tilde{p}}_k \leq \frac{\dot{\omega}_\text{sup}}{p}$$ (4.22b)

where, once again, bounds \( \frac{\dot{\omega}_\text{inf}}{p} \) and \( \frac{\dot{\omega}_\text{sup}}{p} \) are computed as:

$$\dot{\omega}_{i,k}^\text{inf} = \max \left( \dot{\omega}_{i,k}^\text{inf}, \frac{\dot{\omega}_{i,k}^\text{sup} - \dot{\dot{\omega}}_{i,k}}{\Delta t} \right)$$

$$\dot{\omega}_{i,k}^\text{sup} = \min \left( \dot{\omega}_{i,k}^\text{sup}, \frac{\dot{\omega}_{i,k}^\text{inf} - \dot{\dot{\omega}}_{i,k}}{\Delta t} \right)$$

to satisfy acceleration bounds and to keep the a-priori velocity estimate at time step \( k + 1 \) inside velocity bounds. Finally, by updating the estimated joint accelerations

$$\dot{\dot{\tilde{p}}}_k \leftarrow \ddot{\tilde{p}}_k$$

we obtain an estimation of the kinematic state of the human worker that is coherent with respect to position, velocity and acceleration bounds and that, at the same time, is close to the estimation computed by the LKF.

Obviously, the refined estimation of the kinematic state is sent back to the LKF in order to keep the evolution of the filter coherent with respect to the output of the sensor fusion algorithm.

It is worth noting that this hierarchical optimization stage can be completely solved in closed-form without the need of numerical solvers. As a matter of fact QP problems (4.20), (4.21) and (4.22) can be re-formulated as

$$\min_{\tilde{s}} \| \tilde{s} - \hat{s} \|^2$$ (4.23a)

$$A\tilde{s} \leq b$$ (4.23b)

where:

$$A = \begin{bmatrix} -I & I \end{bmatrix}, \quad b = \begin{bmatrix} -s^\text{inf} \\ +s^\text{sup} \end{bmatrix}$$

Thanks to the theoretical results presented in [123, 124], the solution of problem (4.23) corresponds to the solution of problem (4.24) where we only consider the active set of constraints (\( \bar{A} \) and \( \bar{b} \)), expressed as equality constraints:

$$\min_{\tilde{s}} \| \tilde{s} - \hat{s} \|^2$$ (4.24a)

$$\bar{A}\tilde{s} = \bar{b}$$ (4.24b)

The closed-form solution of problem (4.24) is:

$$\tilde{s} = \hat{s} - \bar{A}^T \left( \bar{A}\bar{A}^T \right)^{-1} (\bar{A}\hat{s} - \bar{b})$$ (4.25)
Chapter 4. Tracking Full Upper Body Motion with Multiple Depth Sensors

Figure 4.1: A scheme of the experimental setup comprising the two depth sensors, the data acquisition processes and the main sensor fusion / motion tracking process.

and it can be further manipulated in order to obtain

\[
\begin{align*}
\tilde{A}s &= \tilde{A} \left[ \tilde{s} - \tilde{A}^T (\tilde{A} \tilde{A}^T)^{-1} (\tilde{A}s - b) \right] \\
A\tilde{s} &= A\tilde{s} - A \tilde{A}^T (A \tilde{A}^T)^{-1} (A\tilde{s} - b) \\
A\tilde{s} &= \tilde{A}s - (\tilde{A}s - b) \\
A\tilde{s} &= b
\end{align*}
\]

Consequently, to solve problem (4.24) (and, in turn, problem (4.23)) it is sufficient to apply, to each element of the original estimate \( \hat{s} \), the corresponding saturations:

\[
\forall \tilde{s}_i \in \tilde{s} \implies \tilde{s}_i = \min \left( \max (\hat{s}_i, s_{i}^{\text{inf}}), s_{i}^{\text{sup}} \right)
\]

Given this result, QP problems (4.20), (4.21) and (4.22) can be trivially solved by identifying the constraints violated by the current estimation and by applying the corresponding saturations.

4.4 Experimental Validation

The experimental setup is shown in Figure 4.1. It includes two distinct depth sensors: a Microsoft Kinect and an Asus xTion. The data acquisition stage is performed by two separate processes running on two distinct machines. Each process acquires the skeletal points coordinates from the depth sensor through the OpenNI driver [9] and converts them into human joint positions according to the inverse kinematic calculations introduced in Chapter 2 and further detailed in Appendix B. Then the acquired measures are sent to a third process that implements the tracking strategy. The current implementation guarantees that a single iteration of the algorithm is executed within 4 ms, thus allowing it to be seamlessly integrated with a real-time robot control system.

In order to validate both the sensor fusion and the motion tracking functionalities, a very simple experiment has been performed. At time \( t = 0 \) the algorithm is started
4.4. Experimental Validation

and the first depth sensor starts acquiring data. Then, at time $t = 4.85 \text{s}$, also the second depth sensor starts acquiring data and sending them to the algorithm.

Figure 4.2 shows how the motion tracking algorithm is able to merge the information acquired by the different sensors while satisfying the imposed joint position limits. Notice that data acquired from the second sensor (green solid lines) are null until the corresponding acquisition process starts ($t = 4.85 \text{s}$, as said before). On the other hand, Figures 4.3 and 4.4 show the estimation of human joint velocities and accelerations, respectively. Once again, the imposed lower and upper bounds are satisfied. Finally, the estimation of the human kinematic model parameters is displayed in Figure 4.5.

These results clearly demonstrate that the proposed motion tracking strategy presents several advantages with respect to using directly the measures acquired from the depth sensors. First of all, the estimation of human joint positions and kinematic parameters is significantly more precise (in terms of reduced impact of noise) and accurate (in terms of compensation of spikes and outliers) than the corresponding acquired measures. Moreover, the possibility of enforcing constraints increases the robustness of the estimation against measurement errors due to noise and possible interference between multiple sensors [82].

Finally, the proposed algorithm is able to provide a significantly smoother estimation of the human joint velocities than the one obtained by numerically differentiating subsequent measures acquired from the sensors. For instance, Figure 4.6 shows that our algorithm is able to estimate the linear velocity $\dot{x}$ while the numerical differentiation of the acquired values of the $x$ coordinate produces inconsistent results.
Figure 4.2: Tracking experiment - human joint positions: data acquired from depth sensor #1 (red solid line), data acquired from depth sensor #2 (green solid line), estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).
4.4. Experimental Validation

Figure 4.3: Tracking experiment - human joint velocities: estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).
Chapter 4. Tracking Full Upper Body Motion with Multiple Depth Sensors

Figure 4.4: Tracking experiment - human joint accelerations: estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).
4.4. Experimental Validation

Figure 4.5: Tracking experiment - human kinematic model parameters: data acquired from depth sensor #1 (red solid line), data acquired from depth sensor #2 (green solid line) and values estimated by the tracking algorithm (blue solid line).

Figure 4.6: Tracking experiment - linear velocity along the X-axis: numerical differentiation of data acquired from depth sensor #1 (red solid line), numerical differentiation of data acquired from depth sensor #2 (green solid line) and values estimated by the tracking algorithm (blue solid line).
Part II

Perception: Predicting Human Motion
CHAPTER 5

Human Motion Prediction: Estimating Human Walking Trajectories

5.1 Introduction

Beside motion detection and tracking, another fundamental functionality that the robot control system should perform in a HRI scenario is represented by human motion prediction. As a matter of fact, the combination of motion tracking and prediction should allow the robot control system to select the most suitable control strategy in order to avoid and/or resolve a possibly dangerous situation as soon as possible.

In this Chapter a specific kind of human motion prediction is considered: Intention Estimation (from now on “IE”). IE consists in predicting the final destinations of a human walking trajectory on the basis of a series of tracked positions and velocities. This Chapter presents a combined HDT-IE particle filtering strategy that relies on a structured representation of the robotic cell environment. This approach consists in an enhancement of the HDT algorithm previously introduced in Chapter [3]. It presents substantial advantages with respect to both the previous version and state of the art techniques:

- the prediction of the human worker’s trajectory is based on a fully validated motion model including an optimality principle;

- a single Particle Filter is used to both track the position of a detected human and to estimate his/her intention;

- our strategy is able to predict in real-time whether a human worker is heading to a predefined destination area or not, i.e. he/she is reaching an undefined goal;
Chapter 5. Human Motion Prediction: Estimating Human Walking Trajectories

- the prediction of the human worker’s trajectory is used to improve the tracking capabilities of the system.

5.2 Mapping the Robotic Cell Environment

As stated before, our HDT-IE particle filtering strategy relies on a structured map of the robotic cell. However, in a typical HRI scenario human workers moving inside the cell have to reach predefined destination areas in order to cooperate with robots. It is thus reasonable to assume that the map of the cell should not consist in the sole representation of the cell layout, but it should also include some information regarding the trajectories followed by human workers while moving from entrance zones to destination areas.

To obtain this information, several walking trajectories are computed selecting both a specific entrance and a particular destination and enforcing an obstacle avoidance constraint. The trajectories are then generalized in a probabilistic fashion in order to obtain a descriptor for each set characterized by the same entrance and by the same destination. Furthermore, a generalization of the trajectories reaching an “undefined” destination (i.e. none of the predefined goals) is derived and included in the map as well.

First of all, in order to define a map of the robotic cell environment it is necessary to identify the following elements:

- **Entrance Areas** from which human workers enter the cell;
- **Destination Areas** to which human workers are usually heading to in order to perform autonomous operations and/or cooperative tasks with the robots;
- **Destination Paths** followed by human workers from entrances to destination areas;
- **Industrial Robots** with the corresponding workspaces;
- **Generic Obstacles**.

Figure 5.1 shows the robotic cell layout derived from the experimental setup used to validate our HDT-IE Particle Filtering strategy.

In order to obtain a generalized description of the human walking trajectories for each destination area, it is convenient to use model (2.2) coupled with constraints defined by the cell layout (in terms of destination areas, paths and obstacles) and with a proper cost function to generate offline a large number of trajectories that will be generalized at a later stage.

In [97–99] the authors present a method for generating human-like walking trajectories, based on model (2.2), by solving an optimal control problem. This approach, that has been validated with respect to experimental data, relies on a cost function $J(\chi(s), \gamma)$ that minimizes both the energy dissipated during the trajectory and the distance between the current state and the goal state. More in depth:

$$J(\chi(s), \gamma) = \frac{1}{2} \int_0^S (\omega/v)^2 (1 + \gamma^T \Delta^2) ds$$  \hspace{0.5cm} (5.1)

$$\Delta^2_T = \left[ \begin{array}{c}
(x_s-x_g)^2 + (y_s-y_g)^2 \\
(\theta_s-\theta_g)^2 \\
(x_0-x_g)^2 + (y_0-y_g)^2 \\
(\theta_0-\theta_g)^2
\end{array} \right]$$  \hspace{0.5cm} (5.2)
5.2. Mapping the Robotic Cell Environment

where:

- \( s \in [0, S] \) is the natural coordinate associated to the trajectory;
- \( \chi_0 = [x_0 \ y_0 \ \theta_0]^T \) is the initial pose of the human;
- \( \chi_g = [x_g \ y_g \ \theta_g]^T \) is the goal pose;
- \( \chi(s) = [x_s \ y_s \ \theta_s]^T \) represents every intermediate pose;
- \( \gamma^T = [\gamma_1 \ \gamma_2] \) is a set of parameters estimated on the basis of experimental data (see again [97, 99] for further details).

At this point, a large number of goal poses are sampled inside the different destination areas. For each sampled goal, a human walking trajectory is computed by solving the following optimal control problem:

\[
\begin{align*}
\min_{\chi(s)} & \quad J(\chi(s), \gamma) \\
\text{subject to} & \quad (2.2) \\
\chi(0) &= \chi_0, \quad \chi(S) = \chi_g \\
\forall s \in [0, S] & \implies \text{dist}(\chi(s), O) \geq 0
\end{align*}
\]

where \( \text{dist}(\chi, O) \) is a function that measures the minimum distance between a given pose \( \chi \) and the set of obstacles \( O \) that populate the map. A subset of the resulting trajectories is displayed in Figure 5.2.

Finally, it is worth pointing out that the generality of the proposed IE methodology is not affected by the assumption of nonholonomic human motion. As a matter of fact, in order to account for holonomic motion, it would be sufficient to substitute model (2.2) and cost metric (5.1) with the kinematic model (2.4) and cost function presented in [87].

In order to obtain a single generalization/descriptor for each family of trajectories characterized by the same endpoints, the algorithm proposed in [30] has been used. Each set is transformed into a 3D dataset where datapoints are defined by natural coordinate \( s \), \( X \) coordinate \( x_s \) and \( Y \) coordinate \( y_s \). At first, a Gaussian Mixture Model (GMM) composed by 6 bivariate Gaussians is trained and further processed to obtain a Gaussian Mixture Regression (GMR). Then, a sequence of \( L \) values of the natural coordinate \( s \) is used as query points to retrieve a sequence of expected spatial distributions through the GMR. In this way every family of trajectories can be described by an average trajectory \( \mu_l \), plus uncertainty in both spatial coordinates \( \Sigma_l \):

\[
\forall l, \ 1 \leq l \leq L \quad \mu_l = \begin{pmatrix} \mu_{x,l} \\ \mu_{y,l} \end{pmatrix}, \quad \Sigma_l = \begin{pmatrix} \Sigma_{xx,l} & \Sigma_{xy,l} \\ \Sigma_{yx,l} & \Sigma_{yy,l} \end{pmatrix}
\]

Regarding undefined goals, the most convenient way to represent them is to sample, inside the robotic cell layout, a large number of points not belonging to either destination areas, destination paths or robots/obstacles. Then, these sampled points were arranged into a 2D dataset (\( X \) and \( Y \) coordinate) and finally we trained a GMM composed of 20 bivariate Gaussians, each one again characterized by an average position \( \mu \) and a covariance matrix \( \Sigma \). The obtained result is sketched in Figure 5.3.
Chapter 5. Human Motion Prediction: Estimating Human Walking Trajectories

Figure 5.1: Robotic cell layout: entrance (black solid line, marked with 'X'), destination areas (red, green, blue and yellow solid lines, marked with 'X'), destination paths (red, green, blue and yellow dashed lines) and robots/obstacles (cyan solid lines, marked with 'X').

Figure 5.2: Computed trajectories inside cell layout (red, green, blue and yellow dotted lines).
5.3. Map-based Particle Filtering

Starting from the particle filtering strategy explained in Chapter 3, a map-based particle filtering combined approach to HDT and IE has been realized by integrating the environment map previously described inside the filter. Assuming that a human worker walking inside the robotic cell will follow a trajectory sufficiently “near” to at least one of the average trajectories, it is possible to estimate his/her destination by trying to match the best particle extracted from the filter against the different GMR/GMMs. Moreover, the best particle and the trajectory descriptors can be combined to guide particles resampling in order to guarantee a good match with respect to the actual motion state of the tracked human, thus improving the tracking capabilities of the system.

5.3.1 Estimating Destination Areas’ Probabilities

In order to determine the probability that the tracked human is directed to a certain destination (included the undefined one) the best particle coordinates extracted from the filter \( \bar{q}_P = [\bar{q}_x, \bar{q}_y]^T \) must be matched with the data encoded inside the GMR/GMM included in the map.

Considering \( D \) different goals, the probability associated to \( d \)-th destination area can be estimated by summing up the probabilities corresponding to the \( L_d \) GMR/GMM points. Since each GMR/GMM element consists in a bivariate Gaussian having average \( \mu_i^d \) and covariance matrix \( \Sigma_i^d \), the following result is obtained (including final...
normalization):

\[
P_d = \sum_{l=1}^{L_d} \left( \frac{1}{2\pi \sqrt{|\Sigma^d_l|}} e^{-\frac{1}{2}(\bar{q}_P - \mu^d_l)^T (\Sigma^d_l)^{-1}(\bar{q}_P - \mu^d_l)} \right)
\]

\[
P_d \leftarrow \frac{P_d}{\sum_{d=1}^D P_d}, \quad \forall d \in [1, D]
\]

(5.4)

### 5.3.2 Particles Resampling Strategy

Knowing in real time both the estimated position of the tracked human and the probability with which he/she will reach each destination it is possible to select, for each GMR/GMM, a set of Gaussians being the nearest with respect to the best particle. By performing resampling inside this subspace of the motion model state space, we obtain particles that will guarantee a good match with respect to the actual motion state of the tracked human, since they are close to the current best particle and, at the same time, they lie on a trajectory that the human worker is following with a known probability.

For the \(d\)-th GMR/GMM, given the best particle position \(\bar{q}_P\) and its linear velocity \(\bar{q}_v\), the set \(I_d\) of the GMR points within the reach of \(\bar{q}\) in \(\Delta t\) seconds can be selected as:

\[
I_d = \left\{ (\mu^d_l, \Sigma^d_l) \mid 1 \leq l \leq L_d \land ||\mu^d_l - \bar{q}_P|| \leq (\bar{q}_v \Delta t) \right\}
\]

(5.5)

In case no points are selected from a GMR, the closest point with respect to the best particle is considered:

\[
I_d = \left\{ (\mu^d_{\bar{l}}, \Sigma^d_{\bar{l}}) \mid \bar{l} = \arg\min_{1 \leq l \leq L_d} (||\mu^d_l - \bar{q}_P||) \right\}
\]

(5.6)

At this point, the number of particles \(M_d\) that will be resampled from the \(d\)-th GMR/GMM can be determined as:

\[
M_d = \frac{M}{2D} + \frac{p_d M}{2}
\]

(5.7)

While the first half of the new particle set is equally split among the different GMR/GMM, the second one is assigned proportionally with respect to the different destination probabilities, realizing in this way a balance between exploration and exploitation. In particular, this solution provides robustness to our HDT/IE strategy since a human worker moving inside the cell without following a standard path (or switching between different destination paths) will not be lost by the tracking system, thanks to those particles sampled from the GMR/GMM not corresponding to the current estimated goal.

Once the number \(M_d\) of particles to be sampled from the \(d\)-th GMR/GMM has been determined, it is uniformly distributed over the Gaussians contained in \(I_d\):

\[
M_{I,d} = \frac{M_d}{|I_d|}
\]

(5.8)

Finally, the resampling stage takes place in the following way:

\[
q_n = \begin{cases} 
  x_n & \sim N(\mu^d_l, \Sigma^d_l) \\
  y_n & \sim N(\mu^d_l, \Sigma^d_l) \\
  \theta_n = \bar{q}_\theta, \quad v_n = \bar{q}_v, \quad \omega_n = \bar{q}_\omega
\end{cases}
\]

(5.9)
5.4. Experimental verification

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RED</td>
<td>5.820</td>
<td>0.880</td>
<td>15.12 %</td>
</tr>
<tr>
<td>GREEN</td>
<td>8.020</td>
<td>2.720</td>
<td>33.91 %</td>
</tr>
<tr>
<td>BLUE</td>
<td>5.840</td>
<td>1.780</td>
<td>30.48 %</td>
</tr>
<tr>
<td>YELLOW</td>
<td>4.360</td>
<td>0.800</td>
<td>18.34 %</td>
</tr>
</tbody>
</table>

Table 5.1: Experiment 01: Arrival time and corresponding prediction time for each tracked trajectory.

where $1 \leq d \leq D$, $l \in I_d$ and $1 \leq n \leq M_{l,d}$.

5.4 Experimental verification

The approach here presented has been experimentally validated considering four different scenarios: trajectories reaching predefined goals, trajectories ending up in the undefined area, trajectories switching between different goals and multiple simultaneous trajectories. The experimental setup includes the robotic cell layout described in Section 5.2, three industrial robots (an ABB IRB140, an ABB dual-arm prototype robot and a COMAU Smart-Six) and two AXIS 212 PTZ RGB network cameras connected via Ethernet to the workstation running the HDT/IE application.

In the following each experiment is described and the corresponding results are presented. Analogous experiments are documented in video [103], where images acquired from the cameras are elaborated online to show for each human the best particle box and the predicted destination.

5.4.1 Exp. 01 - Trajectories reaching predefined destinations

In the first experiment a human worker enters the robotic cell four times, each time reaching a different destination area. Figure 5.4 shows the four tracked trajectories while Figure 5.5 shows, for each trajectory, the probabilities corresponding to the different destination areas. Considering the previously mentioned probability threshold of 0.75, it is possible to identify, for each trajectory, the “prediction time”, i.e. the time instant from which human destination is correctly predicted. In Figure 5.4 the position of the human worker corresponding to each prediction time is highlighted.

On the other hand, Table 5.1 shows, for every experiment, the arrival time (i.e. the time required to reach the goal) and the prediction time, the last one measured both in seconds and in percentage with respect to the total time. These experimental data demonstrate the effectiveness of our approach since not only human intention is correctly predicted, but also the prediction is established well in advance with respect to the arrival time.

5.4.2 Exp. 02 - Trajectories reaching undefined destination

In the second experiment a human worker enters the cell and reaches an undefined destination (i.e. a position that does not belong to any destination area). Once again, Figure 5.4 shows the tracked trajectory (highlighting the position of the human worker corresponding to the prediction time), while Figure 5.6 shows the probabilities corresponding to the different destination areas. The prediction time is equal to 1.86 s
with respect to a total trajectory time of 3.2 s, demonstrating that the system correctly recognizes human intention even in case of UNDEFINED destinations.

5.4.3 Exp. 03 - Trajectories with switching destination

In the third experiment a human worker enters the robotic cell twice. The first time he follows the GREEN destination path, then he moves to the BLUE destination area. The second time he reaches the RED goal and continues walking until the GREEN destination area. Figure 5.7 shows the two tracked trajectories, while Figure 5.8 shows, for each trajectory, the probabilities corresponding to the different destination areas. In the first (second) case the system is able to recognize the worker’s initial intention to reach the GREEN (RED) destination. As soon as the worker exits the standard trajectory the system switches to UNDEFINED destination and in the end it recognizes that the worker is directed towards the BLUE (GREEN) goal. The positions of the human worker corresponding to the different prediction times are highlighted in Figure 5.7.

5.4.4 Exp. 04 - Multiple simultaneous trajectories

In the last experiment two human workers enter the cell, heading to the RED and BLUE destination areas, respectively. Figure 5.9 shows the two tracked trajectories (with the positions of the two human workers corresponding to the prediction times). Figure 5.10 and 5.11 show the probabilities corresponding to the different destination areas for the first and the second trajectory, respectively. The destinations of both trajectories are correctly estimated, proving that our IE strategy is effective also in the case of multiple human workers moving inside the robotic cell.
5.4. Experimental verification

Figure 5.5: Experiment 01: probabilities of the different destinations (coloured dashed lines) and threshold probability (black dashed line). From top to bottom: RED destination, GREEN destination, BLUE destination and YELLOW destination.

Figure 5.6: Experiment 02: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the tracked trajectory.
Figure 5.7: Experiment 03: first switching trajectory (black solid lines) and second one (black dashed line) with points corresponding to prediction times (red, magenta, green and blue large circles).

Figure 5.8: Experiment 03: probabilities of the different destinations (red, green, blue and yellow dashed lines), probability of the undefined goal (magenta dashed lines) and threshold probability (black dashed lines) during the execution of the tracked trajectories.
5.4. Experimental verification

Figure 5.9: Experiment 04: trajectory of the first human worker (black solid lines) and trajectory of the second human worker with points corresponding to prediction times (red and blue large circles).

Figure 5.10: Experiment 04: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the first tracked trajectory (destination = RED).

Figure 5.11: Experiment 04: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the first tracked trajectory (destination = BLUE).
CHAPTER 6

Human Motion Prediction: Swept Volumes

6.1 Introduction

As stated before, in order to guarantee human workers’ safety in a HRI setup, the motion of the human worker has to be clearly taken into account to safely adjust the trajectory of the robot. Moreover, beside predicting the final destination of the human worker moving inside the robotic cell, the control system should also be able to predict the volume occupied by the human, at least when he/she is actually sharing the workspace with the manipulator during cooperative tasks.

Given the human kinematic model (2.10) introduced in Chapter 2, it is possible to develop a simple algorithm that predicts the space occupied by a human worker within a pre-defined time span (for example the time needed by the robot to stop). This prediction consists in a series of swept volumes \[130,131\], i.e. convex polytopes that represent the entire volume that the human can reach given a specific kinematic configuration and a chosen time horizon.

To compute the swept volumes, first a reachable set is determined for each DOF composing the kinematic model and then the swept volumes are obtained by superimposing the motion of each joint, according to the reachable set previously determined.

In the following, the calculation of the reachable set for each human joint is detailed and then the algorithm computing the human swept volumes is presented. More in detail two options are presented:

- computing reachable sets and swept volumes considering joint positions and bounded joint velocities;
- computing reachable sets and swept volumes considering joint positions and velocities and bounded joint accelerations;

and, finally, a comparison between the two approaches is discussed.
Chapter 6. Human Motion Prediction: Swept Volumes

6.2 Human Kinematics Reachable Set

In order to compute the swept volumes, it is necessary to determine, for each DOF composing the human kinematic model, the reachable set on the basis of the current kinematic configuration of the human and of a pre-defined time span $T_s$. In the following, two different approaches to the calculation of the reachable set are presented. The first one relies on joint positions and bounded joint velocities, while the second one takes into account joint positions, joint velocities and bounded joint accelerations.

6.2.1 Reachable Set based on Joint Positions and Bounded Joint Velocities

Consider the kinematic model (2.2) describing human walking motion together with the following constraints on its inputs:

$$
0 \leq v \leq v_{\text{max}} \\
|\omega| \leq \omega_{\text{max}}
$$

(6.1)

In order to compute the reachable set in $T_s$ seconds, we consider every possible combination of inputs. In particular, for the case

$$
v(t) = v_{\text{max}} \\
\omega(t) = 0
$$

(6.2)

the solution of (2.2) after $T_s$ seconds is:

$$
\begin{align*}
x_{T_s} &= x_0 + v_{\text{max}} T_s \cos \theta_0 \\
y_{T_s} &= y_0 + v_{\text{max}} T_s \sin \theta_0 \\
\theta_{T_s} &= \theta_0
\end{align*}
$$

(6.3)

Similarly, for:

$$
v(t) = v_{\text{max}} \\
\omega(t) = \pm \omega_{\text{max}}
$$

(6.4)

one obtains the following solution:

$$
\begin{align*}
x_{T_s} &= x_0 \pm \frac{v_{\text{max}}}{\omega_{\text{max}}} [\sin (\theta_0 \pm \omega_{\text{max}} T_s) - \sin \theta_0] \\
y_{T_s} &= y_0 \pm \frac{v_{\text{max}}}{\omega_{\text{max}}} [\cos (\theta_0 \pm \omega_{\text{max}} T_s) - \cos \theta_0] \\
\theta_{T_s} &= \theta_0 \pm T_s \omega_{\text{max}}
\end{align*}
$$

(6.5)

Therefore, the reachable set of the unicycle can be conservatively represented by a circular sector of radius $R$, spanning an angle $\beta$ symmetrically with respect to the current direction of motion $\theta_0$ (see Figure 6.1):

$$
R = v_{\text{max}} T_s \\
\beta = 2 \cos^{-1} \left( \frac{\sin (\omega_{\text{max}} T_s)}{\sqrt{2 - 2 \cos (\omega_{\text{max}} T_s)}} \right)
$$

(6.6)
6.2. Human Kinematics Reachable Set

Thanks to this approximation, the human walking model can be described as a composition of two independent degrees of freedom: a rotational one followed by a translational one. The corresponding reachable set follows immediately from this approximation:

\[
x_{T_s} = \begin{bmatrix} x_{T_s}^x \\ x_{T_s}^y \\ \theta_{T_s} \\ \theta_{T_s}^+ \end{bmatrix}
= \begin{bmatrix} x_0 \\ x_0 + R \cos \theta_0 \\ \theta_0 - \beta \\ \theta_0 + \beta \end{bmatrix}
\]

\[
y_{T_s} = \begin{bmatrix} y_{T_s}^x \\ y_{T_s}^y \\ \theta_{T_s} \\ \theta_{T_s}^+ \end{bmatrix}
= \begin{bmatrix} y_0 \\ y_0 + R \sin \theta_0 \\ \theta_0 - \beta \\ \theta_0 + \beta \end{bmatrix}
\]

\[
\theta_{T_s} = \begin{bmatrix} \theta_{T_s}^- \\ \theta_{T_s}^+ \end{bmatrix}
= \begin{bmatrix} \theta_0 \\ \theta_0 - \beta \end{bmatrix}
\]

Figure 6.1: Unicycle reachable set (dashed) and its convex approximation (solid bold)

Finally, if we consider kinematic model (2.4), that accounts for lateral walking, we need to introduce the following constraint on the orthogonal velocity:

\[-v_{\perp}^{\text{max}} \leq v_{\perp} \leq v_{\perp}^{\text{max}} \]

Consequently, the \( x_{T_s} \) and \( y_{T_s} \) components of the reachable set determined in (6.7) can be modified as follows:

\[
x_{T_s} \leftarrow x_{T_s} + \begin{bmatrix} -v_{\perp}^{\text{min}} T_s \sin \theta_0 \\ -v_{\perp}^{\text{max}} T_s \sin \theta_0 \end{bmatrix}
\]

\[
y_{T_s} \leftarrow y_{T_s} + \begin{bmatrix} v_{\perp}^{\text{min}} T_s \cos \theta_0 \\ v_{\perp}^{\text{max}} T_s \cos \theta_0 \end{bmatrix}
\]

As for the kinematic model of the human arm (2.6), its workspace is clearly limited due to some intrinsic limitations in the gleno-humeral joint (shoulder) as well as in the elbow. Differently from robots, however, these limits are coupled, as described e.g. in [75]. In particular, the range of motion of the human arm is limited to the region identified by the following constraints:

\[-9^\circ \leq \alpha_1 \leq 160^\circ \]
\[
\begin{align*}
- 43^\circ + \frac{\alpha_1}{3} & \leq \alpha_2 \leq 153^\circ - \frac{\alpha_1}{6} \\
- 90^\circ + \frac{7\alpha_1}{9} - \frac{\alpha_2}{9} + \frac{2\alpha_1\alpha_2}{810} & \leq \alpha_3 \leq 60^\circ + \frac{4\alpha_1}{9} - \frac{5\alpha_2}{9} + \frac{5\alpha_1\alpha_2}{810} \\
20^\circ & \leq \alpha_4 \leq 180^\circ \\
- 30^\circ & \leq \rho \leq 90^\circ 
\end{align*}
\] (6.13b)

where \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \) are the arm joint angles expressed in degrees and \( \rho \) is the torso bending angle also expressed in degrees.

By considering the following bounds on arm angles velocities:
\[
\begin{align*}
- \dot{\alpha}_{\text{max}} & \leq \dot{\alpha} \leq + \dot{\alpha}_{\text{max}} \\
- \dot{\rho}_{\text{max}} & \leq \dot{\rho} \leq + \dot{\rho}_{\text{max}}
\end{align*}
\] (6.14)

the reachable set in \( T_s \) seconds can be determined by simply computing:
\[
\begin{align*}
\alpha_{T_s}^- = \left[ \alpha_0 - T_s \dot{\alpha}_{\text{max}} \right] \\
\alpha_{T_s}^+ = \left[ \alpha_0 + T_s \dot{\alpha}_{\text{max}} \right] \\
\rho_{T_s}^- = \left[ \rho_0 - T_s \dot{\rho}_{\text{max}} \right] \\
\rho_{T_s}^+ = \left[ \rho_0 + T_s \dot{\rho}_{\text{max}} \right]
\end{align*}
\] (6.15)

and by limiting the result to the region inside the joint limits, previously introduced in (6.13).

### 6.2.2 Reachable Set based on Joint Positions and Velocities

A more accurate calculation of the swept volumes can be obtained by computing the reachable set of each DOF of the human kinematic model on the basis of both joint positions and velocities and by considering fixed bounds on joint accelerations.

For what concerns walking kinematics, consider a generic starting configuration:
\[
\begin{align*}
x(0) = x_0 & \quad y(0) = y_0 & \quad \theta(0) = \theta_0 \\
v(0) = v_0 & \quad \omega(0) = \omega_0 & \quad v_\perp(0) = v_{\perp,0}
\end{align*}
\] (6.17)

along with fixed lower and upper bounds on linear and angular accelerations:
\[
\begin{align*}
v^{\text{inf}} & \leq \ddot{v} \leq v^{\text{sup}} \\
\omega^{\text{inf}} & \leq \ddot{\omega} \leq \omega^{\text{sup}} \\
v^{\perp,\text{inf}} & \leq \ddot{v}_\perp \leq v^{\perp,\text{sup}}
\end{align*}
\] (6.19)

On the basis of these constraints, it is possible to compute the minimum and maximum velocities that a walking human can reach within \( T_s \) seconds:
\[
\begin{align*}
v_{\text{min}} & = 0 \\
v_{\text{max}} & = \min \left\{ v_0 + \dot{v}_{\text{sup}} \cdot T_s, \ v_{\text{sup}} \right\} \\
\omega_{\text{min}} & = \max \left\{ \omega_0 + \dot{\omega}_{\text{inf}} \cdot T_s, \ \omega_{\text{inf}} \right\} \\
\omega_{\text{max}} & = \min \left\{ \omega_0 + \dot{\omega}_{\text{sup}} \cdot T_s, \ \omega_{\text{sup}} \right\} \\
v_{\perp,\text{min}} & = \max \left\{ v_{\text{bot},0} + v^{\perp,\text{inf}} \cdot T_s, \ v^{\perp,\text{inf}} \right\} \\
v_{\perp,\text{max}} & = \min \left\{ v_{\text{bot},0} + v^{\perp,\text{sup}} \cdot T_s, \ v^{\perp,\text{sup}} \right\}
\end{align*}
\] (6.20)
6.2. Human Kinematics Reachable Set

As already presented in the previous Section, a conservative approximation of the human walking $T_s$-reachable set is determined by considering every possible combination of the minimum and maximum velocity inputs. Since the minimum and maximum angular velocities $\omega_{\text{min}}$ and $\omega_{\text{max}}$ are not necessarily equal in modulus, the reachable set of the unicycle can be conservatively represented by two distinct circular sectors of radius $R = v_{\text{max}} T_s$, spanning two different angles $\beta_{\text{min}}$ and $\beta_{\text{max}}$ with respect to the original direction of motion $\theta_0$, as shown in Fig. 6.2. In particular:

$$\beta_{\text{min}} = 2 \cos^{-1} \left( \frac{\sin (\omega_{\text{min}} T_s)}{\sqrt{2 - 2 \cos (\omega_{\text{min}} T_s)}} \right)$$  \hspace{1cm} (6.26)$$

$$\beta_{\text{max}} = 2 \cos^{-1} \left( \frac{\sin (\omega_{\text{max}} T_s)}{\sqrt{2 - 2 \cos (\omega_{\text{max}} T_s)}} \right)$$  \hspace{1cm} (6.27)$$

Finally, the reachable set can be determined as:

$$x_{T_s} = \begin{bmatrix} x_{T_s}^- \n x_{T_s}^+ \end{bmatrix} = \begin{bmatrix} x_0 \\ x_0 + v_{\text{max}} T_s \cos \theta_0 \end{bmatrix} + \begin{bmatrix} -v_{\text{max}} T_s \sin \theta_0 \\ v_{\text{max}} T_s \sin \theta_0 \end{bmatrix}$$  \hspace{1cm} (6.28)$$

$$y_{T_s} = \begin{bmatrix} y_{T_s}^- \n y_{T_s}^+ \end{bmatrix} = \begin{bmatrix} y_0 \\ y_0 + v_{\text{max}} T_s \sin \theta_0 \end{bmatrix} + \begin{bmatrix} v_{\text{min}} T_s \cos \theta_0 \\ v_{\text{max}} T_s \cos \theta_0 \end{bmatrix}$$  \hspace{1cm} (6.29)$$

$$\theta_{T_s} = \begin{bmatrix} \theta_{T_s}^- \n \theta_{T_s}^+ \end{bmatrix} = \begin{bmatrix} \theta_0 + \beta_{\text{min}} \\ \theta_0 + \beta_{\text{max}} \end{bmatrix}$$  \hspace{1cm} (6.30)$$

Moving from the walking kinematics to the upper limbs, the reachable set in $T_s$ seconds

$$\alpha_{T_s} = \begin{bmatrix} \alpha_{T_s}^- \n \alpha_{T_s}^+ \end{bmatrix}$$

can be computed according to Algorithm 2 on the basis of joint positions $\alpha$, joint velocities $\dot{\alpha}$ and fixed acceleration bounds $\ddot{\alpha}_{\text{inf}}$ and $\ddot{\alpha}_{\text{sup}}$.

Identical considerations lead to the calculation of $\rho_{T_s} = \begin{bmatrix} \rho_{T_s}^- \n \rho_{T_s}^+ \end{bmatrix}$. Once again, we limit the obtained result to the region inside the joint angle limits, previously introduced in (6.13).
Algorithm 2 Human arm \( T_s \)-reachable set \( \alpha_{T_s} \)

1: for all \( \alpha_i \in \alpha \) do
2: if \( \dot{\alpha}_i + \dot{\alpha}_i^{sup} T_s \leq \dot{\alpha}_i^{sup} \) then
3: \( \alpha_{T_s,i}^+ = \alpha_i + \dot{\alpha}_i T_s + \frac{1}{2} \dot{\alpha}_i^{sup} T_s \)
4: else
5: \( T^* = (\dot{\alpha}_i^{sup} - \dot{\alpha}_i) / \dot{\alpha}_i^{sup} \)
6: \( \alpha_{T_s,i}^+ = \alpha_i + \dot{\alpha}_i T_s + \frac{1}{2} \dot{\alpha}_i^{sup} T_s^* + \dot{\alpha}_i^{sup} (T_s - T^*) \)
7: end if
8: if \( \dot{\alpha}_i + \dot{\alpha}_i^{inf} T_s \geq \dot{\alpha}_i^{inf} \) then
9: \( \alpha_{T_s,i}^- = \alpha_i + \dot{\alpha}_i T_s + \frac{1}{2} \dot{\alpha}_i^{inf} T_s \)
10: else
11: \( T^* = (\dot{\alpha}_i^{inf} - \dot{\alpha}_i) / \dot{\alpha}_i^{inf} \)
12: \( \alpha_{T_s,i}^- = \alpha_i + \dot{\alpha}_i T_s + \frac{1}{2} \dot{\alpha}_i^{inf} T_s^* + \dot{\alpha}_i^{inf} (T_s - T^*) \)
13: end if
14: end for

6.3 Human Swept Volumes Calculation

In the following we present a method to predict the occupancy of the human silhouette in terms of a series of convex polytopes, using the computation of the reachable sets for each DOF. Assume a given configuration of the human upper body:

\[
p = \begin{bmatrix} x & y & \theta & \rho & \alpha^{right} & \alpha^{left} \end{bmatrix}^T
\]

\[
\dot{p} = \begin{bmatrix} \dot{x} & \dot{y} & \omega & \dot{\rho} & \dot{\alpha}^{right} & \dot{\alpha}^{left} \end{bmatrix}^T
\]

(6.32)

Once we determine, for each DOF, the \( T_s \)-reachable set lower and upper bounds

\[
p_{T_s}^- = \begin{bmatrix} x_{T_s}^- & y_{T_s}^- & \theta_{T_s}^- & \rho_{T_s}^- & \alpha_{T_s}^{left}^- & \alpha_{T_s}^{right}^- \end{bmatrix}^T
\]

(6.33)

\[
p_{T_s}^+ = \begin{bmatrix} x_{T_s}^+ & y_{T_s}^+ & \theta_{T_s}^+ & \rho_{T_s}^+ & \alpha_{T_s}^{left}^+ & \alpha_{T_s}^{right}^+ \end{bmatrix}^T
\]

(6.34)

the problem of computing and representing the human occupancy can be solved by mapping the following inequalities into a corresponding region inside the 3D Cartesian space:

\[
p_{T_s}^- \leq p \leq p_{T_s}^+
\]

(6.35)

6.3.1 Swept Volumes of Convex Objects

Given a generic 3-dimensional convex object \( O \), there are two different swept volumes that can be computed: a translational swept volume and a rotational one, depending on the kind of motion \( O \) is subjected to. If we consider a prismatic joint, the translational swept volume of \( O \) can be computed by applying the corresponding translation to each point belonging to \( O \) and by determining the convex hull of the resulting points, as shown in Figure 6.3a.

In turn, for a rotational DOF, we exploit one of the methods described in [130]. In particular, when applying a rotation to each point belonging to \( O \), we obtain a circular arc. A possible way to approximate this arc with a finite number of points is to construct a triangle as it is shown in Figure 6.4. Therefore, the rotational swept volume of \( O \) can be easily obtained as the convex hull of the vertices of all the triangles, see Figure 6.3b.
6.3. Human Swept Volumes Calculation

Figure 6.3: Figure 6.3a translational swept volume of convex object. Figure 6.3b rotational swept volume of convex object. The corresponding convex hulls are also highlighted.

Figure 6.4: Triangular approximation of a circular arc.

6.3.2 Human Swept Volumes Calculation Algorithm

Knowing how to determine the translational and rotational swept volumes for a generic set of points, the prediction of human occupancy can be determined by computing a specific swept volume for each limb:

- head - HD;
- thorax - THX;
- upper left arm (from shoulder to elbow) - ULA;
- lower left arm (from elbow to wrist) - LLA;
- upper right arm - URA;
- lower right arm - LRA.

Assume that a convex object $V_l$ representing the $l$-th limb is given by means of its vertices set. Then, referring to the human kinematic model (2.10), a list of DOFs from the current limb to the world frame can be arranged.
Chapter 6. Human Motion Prediction: Swept Volumes

Algorithm 3 Swept Volume Calculation

1: \( Sv \leftarrow \emptyset; \)
2: \( L = \{ HD, THX, ULA, LLA, URA, LRA \}; \)
3: for all \( l \in L \) do
4: \( \mathcal{P} = \text{kinematicChain}(l, p); \)
5: \( \mathcal{V}_l \leftarrow \text{InitSweptVolume}(); \)
6: if \( \text{isPrismatic}(p_1) \) then
7: \( \mathcal{V}_l \leftarrow \text{sweepLin}(\mathcal{V}_l, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+); \)
8: else
9: \( \mathcal{V}_l \leftarrow \text{sweepRot}(\mathcal{V}_l, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+); \)
10: end if
11: for all \( p_i \in \mathcal{P} \) do
12: if \( p_i \neq p_1 \) then
13: \( \mathcal{V}_l \leftarrow A_{i-1}^+\cdot \mathcal{V}_l = \left\{ A_{i-1}^+ \cdot r_i^l | r_i^l \in \mathcal{V}_l \right\}; \)
14: if \( \text{isPrismatic}(p_i) \) then
15: \( \mathcal{V}_l \leftarrow \text{sweepLin}(\mathcal{V}_l, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+); \)
16: else
17: \( \mathcal{V}_l \leftarrow \text{sweepRot}(\mathcal{V}_l, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+); \)
18: end if
19: end if
20: end for
21: \( Sv \leftarrow Sv \cup \{ \mathcal{V}_l \}; \)
22: end for

Consequently, the swept volume of the \( l \)-th limb can be determined by iteratively applying the proper sweeping strategy to \( \mathcal{V}_l \) for each DOF connecting the \( l \)-th limb to the world-base frame. Each sweeping operation is calculated on the basis of the upper and lower bounds \( p_{T_{l,i}}^- \) and \( p_{T_{l,i}}^+ \) previously computed. A pseudo-code version of this procedure is sketched in Algorithm 3 where:

- \text{kinematicChain}(l, p) is a function that computes \( \mathcal{P} \), i.e. the set of DOFs connecting the limb to the world-base frame. The set is ordered starting from the limb and going backwards along the kinematic chain until the world-base frame is reached;

- \text{InitSweptVolume}() is a function that initializes a swept volume by including into the set the endpoints of the link corresponding to the first DOF;

- \text{isPrismatic}(p_i) is a logical predicate that is true if its argument is a prismatic DOF and false otherwise;

- \( p_{T_{l,i}}^- \) and \( p_{T_{l,i}}^+ \) are, respectively, the lower and upper bound of the \( i \)-th DOF selected by \text{kinematicChain};

- \text{sweepLin}(\mathcal{V}, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+) computes the translational swept volume of the set of points \( \mathcal{V} \), given the DOF bounds;

- \text{sweepRot}(\mathcal{V}, \ p_{T_{l,i}}^-, p_{T_{l,i}}^+) computes the rotational swept volume of the set of points \( \mathcal{V} \), given the DOF bounds;
6.3. Human Swept Volumes Calculation

- $A_{i-1}^i$ is the linear transformation matrix from frame $i$ to frame $i-1$, with respect to the output of $\text{kinematicChain}$;
- $r_i^j$ is the $j$-th vertex of the $l$-th swept volume $\mathcal{V}_l$.

Finally, in order to account for the dimensions of the different human body parts, a radius parameter $r$ is introduced and each convex swept volume $\mathcal{V}$ is augmented by computing the Minkowski sum $\oplus$ of its convex hull and a sphere of radius $r$:

$$\forall \mathcal{V}_l \in \mathcal{SV} \implies \mathcal{V}_l^r = \text{convhull}(\mathcal{V}_l) \oplus \{b \in \mathbb{R}^3 : ||b|| \leq r\} \quad (6.36)$$

It is worth mentioning that the augmented swept volumes $\mathcal{V}_l^r$ are used only for visualization purposes. As a matter of fact, according to the formulation of the safety constraints given in Chapter 7, it is possible to incorporate the radius $r$ inside the clearance parameter $d$ of equation (7.24) and to state the safety constraints separately for each swept volume $\mathcal{V}_l$.

6.3.3 Experimental Validation and Discussion

A brief comparison of the two possible approaches for the calculation of the swept volumes is here presented. Given the following kinematic configuration:

$$p = [0 \ m, \ 0 \ m, \ +\frac{1}{2} \pi \ \text{rad}, \ 0 \ \text{rad}, \ 0 \ \text{rad},
\quad 0 \ \text{rad}, \ +\frac{1}{2} \pi \ \text{rad}, \ 0 \ \text{rad}, \ 0 \ \text{rad}, \ +\frac{1}{2} \pi \ \text{rad}]^T$$

$$\dot{p} = [+0.5 \ \frac{m}{s}, \ 0 \ \frac{m}{s}, \ +0.1 \ \frac{\text{rad}}{s}, \ +0.1 \ \frac{\text{rad}}{s}, \ +0.2 \ \frac{\text{rad}}{s}, \ +0.2 \ \frac{\text{rad}}{s},
\quad +0.2 \ \frac{\text{rad}}{s}, \ +0.2 \ \frac{\text{rad}}{s}, \ +0.2 \ \frac{\text{rad}}{s}, \ +0.2 \ \frac{\text{rad}}{s}]^T$$

the following bounds on human joint velocities and accelerations:

$$\dot{p}^{\text{inf}} = [0 \ \frac{m}{s}, \ -0.8 \ \frac{m}{s}, \ -\frac{1}{2} \pi \ \frac{\text{rad}}{s}, \ -0.1 \ \frac{\text{rad}}{s}, \ -0.5 \ \frac{\text{rad}}{s}, \ -0.5 \ \frac{\text{rad}}{s},$$

$$\quad -0.5 \ \frac{\text{rad}}{s}, \ -\frac{3}{10} \ \pi \ \frac{\text{rad}}{s}, \ -0.5 \ \frac{\text{rad}}{s}, \ -0.5 \ \frac{\text{rad}}{s}, \ -0.5 \ \frac{\text{rad}}{s}, \ -\frac{3}{10} \ \pi \ \frac{\text{rad}}{s}]^T$$

$$\dot{p}^{\text{sup}} = [+0.8 \ \frac{m}{s}, \ +0.8 \ \frac{m}{s}, \ +\frac{1}{2} \pi \ \frac{\text{rad}}{s}, \ +0.1 \ \frac{\text{rad}}{s}, \ +0.5 \ \frac{\text{rad}}{s}, \ +0.5 \ \frac{\text{rad}}{s},
\quad +0.5 \ \frac{\text{rad}}{s}, \ +\frac{3}{10} \ \pi \ \frac{\text{rad}}{s}, \ +0.5 \ \frac{\text{rad}}{s}, \ +\frac{3}{10} \ \pi \ \frac{\text{rad}}{s}]^T$$

$$\ddot{p}^{\text{inf}} = [-0.1 \ \frac{m}{s^2}, \ -0.1 \ \frac{m}{s^2}, \ +\frac{1}{2} \pi \ \frac{\text{rad}}{s^2}, \ -\frac{1}{20} \ \pi \ \frac{\text{rad}}{s^2}, \ -0.25 \ \frac{\text{rad}}{s^2}, \ -0.25 \ \frac{\text{rad}}{s^2},$$

$$\quad -0.25 \ \frac{\text{rad}}{s^2}, \ -\frac{3}{20} \ \pi \ \frac{\text{rad}}{s^2}, \ -0.25 \ \frac{\text{rad}}{s^2}, \ -0.25 \ \frac{\text{rad}}{s^2}, \ -\frac{3}{20} \ \pi \ \frac{\text{rad}}{s^2}]^T$$

$$\ddot{p}^{\text{sup}} = [+0.1 \ \frac{m}{s^2}, \ +0.1 \ \frac{m}{s^2}, \ +\frac{1}{2} \pi \ \frac{\text{rad}}{s^2}, \ +\frac{1}{20} \ \pi \ \frac{\text{rad}}{s^2}, \ +0.25 \ \frac{\text{rad}}{s^2}, \ +0.25 \ \frac{\text{rad}}{s^2},$$

$$\quad +0.25 \ \frac{\text{rad}}{s^2}, \ +\frac{3}{20} \ \pi \ \frac{\text{rad}}{s^2}, \ +0.25 \ \frac{\text{rad}}{s^2}, \ +0.25 \ \frac{\text{rad}}{s^2}, \ +\frac{3}{20} \ \pi \ \frac{\text{rad}}{s^2}]^T$$

and $T_s = 0.5 \ s$, swept volumes have been computed following the two approaches previously introduced.

As it can be seen in Figure 6.5, the algorithm based on joint positions and bounded joint velocities produces very large swept volumes, thus entailing a strongly conservative prediction of human occupancy. On the other hand, the second approach takes into account joint positions and velocities and considers fixed upper and lower bounds on joint accelerations, thus obtaining a considerably less conservative prediction of human occupancy, as it can be seen in Figure 6.6.
Chapter 6. Human Motion Prediction: Swept Volumes

Figure 6.5: Example of swept volumes computed on the basis of joint positions and fixed bounds on joint velocities

Figure 6.6: Example of swept volumes computed on the basis of joint positions and velocities and fixed bounds on joint accelerations.

Finally, in order to validate the motion prediction strategy, we consider the same kinematic configuration and the same bounds on human joint velocities and we compute the corresponding swept volumes using the less conservative approach (i.e. the one based on joint positions and joint velocities). Figure 6.7 shows that the prediction of occupancy computed at time $T = 0$ contains the actual kinematic configuration of the moving human until $T = T_s$ and only after $T_s$ the human skeletal approximation exits the swept volumes.
6.3. Human Swept Volumes Calculation

Figure 6.7: Graphic reconstruction of the human kinematic configuration versus the human swept volumes computed at time $T = 0$. First row: human kinematic configuration at time $T = 0$. Second row: human kinematic configuration at time $T = \frac{1}{2}T_s$. Third row: human kinematic configuration at time $T = T_s$. Fourth row: human kinematic configuration at time $T = \frac{3}{2}T_s$. For each row, from left to right, 3D view, projection in the $X$-$Z$ plane and projection in the $X$-$Y$ plane.
Part III

Control: Safety-Oriented Trajectory Generation Algorithms
CHAPTER 7

Safety Constraints

7.1 Introduction

In traditional industrial environments the presence of physical barriers inherently guarantees human workers’ safety by limiting direct man-machine interaction. On the other hand, in a generic HRI setup there are no fences that separate the manipulator workspace from the one dedicated to human workers. If we consider an industrial manipulator performing a task in the presence of both static and dynamic obstacles (for instance, human workers moving inside the robot workspace), a problem that can easily arise is represented by possible collisions between the robot itself and these obstacles. In order to safely accomplish the task by avoiding such collisions, the robot control system must be able to detect the presence of such obstacles, to monitor the distance between them and the manipulator and to adapt the robot velocity accordingly.

Figure 7.1 shows the relationship between production (in terms of robot velocity) and safety. As the distance between the robot and the human becomes smaller, the velocity of the robot should be reduced (e.g. according to the minimum distance criterion, see [1]), thus decreasing the productivity of the robot. On the other hand, even in case of a reduced separation distance, the robot should continue its task if its velocity is oriented so that the distance with the human operator will increase.

To achieve a fruitful trade-off between safety and productivity it is possible to consider the first as a hard constraint, in which respect the latter could be somehow maximized. In other words, safety constraints must be defined in order to ensure that the entire kinematic chain of the manipulator performs a collision-free motion during task execution. These constraints must necessarily take into account the kinematic configuration of the manipulator, known obstacles positions and their geometry.

In the following, some background work on the formalization of the safety constraints is reported and then two different extensions of the original formulation are
7.2 Background on Safety Constraints

We here report some background material on the derivation of these safety constraints for a point obstacle as introduced in [136,138]. Consider Fig. 7.2 which represents a point obstacle $r_{\text{obst}}$ as well as a generic robotic link, whose endpoints are at positions $r_a$ and $r_b$.

Figure 7.2: A rigid beam representing one link and a point-shaped obstacle.

At all time, the robot trajectory must obey the following safety requirement expressed as an inequality among the robot velocity and its distance from an obstacle:

$$\text{velocity} \cdot T_s \leq \text{distance}$$  \hspace{1cm} (7.1)

where the braking time $T_s$ possibly depends on the robot payload [1]. For a generic point $r_s$ on the robot link, with velocity $v_s$, we have:

$$v_s^T \frac{r_{\text{obst}} - r_s}{\|r_{\text{obst}} - r_s\|} T_s \leq \|r_{\text{obst}} - r_s\|$$  \hspace{1cm} (7.2)

where $v_s^T ( (r_{\text{obst}} - r_s) / \|r_{\text{obst}} - r_s\| )$ represents the projection of $v_s$ onto the normalized segment connecting $r_s$ to $r_{\text{obst}}$. This constraint can be further arranged as

$$v_s^T (r_{\text{obst}} - r_s) T_s \leq \|r_{\text{obst}} - r_s\|^2$$  \hspace{1cm} (7.3)
7.2. Background on Safety Constraints

Assume now the following parameterization of the link in terms of position and velocity of its end points

\[ r_s = r_a + s (r_b - r_a) \quad \forall s \in [0, 1] \]

(7.4)

In order to enforce the safety constraints, we require the inequality in (7.3) to be satisfied for all \( s \in [0, 1] \). The left hand side becomes

\[
\begin{align*}
\dot{\mathbf{v}}_s^T (\mathbf{r}_{\text{obst}} - r_s) &= \mathbf{v}_a^T (\mathbf{r}_{\text{obst}} - r_a) - s (\mathbf{v}_b - \mathbf{v}_a)^T (\mathbf{r}_{\text{obst}} - r_a) \\
&\quad - s \mathbf{v}_a^T (r_b - r_a) + s^2 (\mathbf{v}_b - \mathbf{v}_a)^T (r_b - r_a) = 0
\end{align*}
\]

(7.5)

As a result, the set of inequalities describing the safety constraints can be written as follows

\[ \alpha + \beta s \leq g(s), \forall s \in [0, 1] \]

(7.6)

where

\[
\begin{align*}
\alpha &= T_s \mathbf{v}_a^T (\mathbf{r}_{\text{obst}} - r_a) \\
\beta &= T_s (\mathbf{v}_b - \mathbf{v}_a)^T (\mathbf{r}_{\text{obst}} - r_a) - T_s \mathbf{v}_a^T (r_b - r_a) \\
g(s) &= \|\mathbf{r}_{\text{obst}} - r_s\|^2
\end{align*}
\]

(7.7)

Since the left hand side is a linear function in \( s \), it is possible to write the following sufficient condition for the safety constraint (7.6) to be satisfied

\[ \max \{\alpha, \alpha + \beta\} \leq \min_s g(s) \]

(7.8)

where the term

\[ \min_s g(s) = \min_s \|\mathbf{r}_{\text{obst}} - r_s\|^2 \]

(7.9)

represents the distance between the point-shaped obstacle \( \mathbf{r}_{\text{obst}} \) and the segment from \( \mathbf{r}_a \) to \( \mathbf{r}_b \). Finally, we obtain the following pair of inequalities

\[ \alpha = T_s (\mathbf{r}_{\text{obst}} - r_a)^T \mathbf{v}_a \leq \min_s g(s) \]

(7.10)

\[ \alpha + \beta = T_s (\mathbf{r}_{\text{obst}} - r_a)^T \mathbf{v}_b - T_s (r_b - r_a)^T \mathbf{v}_a \leq \min_s g(s) \]

Summarizing, the minimum separation distance criterion can be written in matrix form as:

\[ T_s \mathbf{E} \dot{\mathbf{q}} \leq \mathbf{f} \]

(7.11)

where

\[
\mathbf{E} = \begin{bmatrix} (\mathbf{r}_{\text{obst}} - r_a)^T \mathbf{J}_a \\ (\mathbf{r}_{\text{obst}} - r_a)^T \mathbf{J}_b - (r_b - r_a)^T \mathbf{J}_a \end{bmatrix}
\]

(7.12)

\[ \mathbf{f} = \min_s \|\mathbf{r}_{\text{obst}} - r_s\|^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \]

\( \mathbf{J}_a \) and \( \mathbf{J}_b \) are position Jacobians of the two link end points.
Chapter 7. Safety Constraints

The set of inequalities in (7.11) represents an easy way to check whether the current robot state of motion satisfies the minimum separation distance or, in other terms, if the current robot velocity is sufficiently low to allow the robot to stop before a collision occurs. As an example, Fig. 7.3 shows the regions to be avoided around an industrial robot in position $q$, with nominal velocities $\dot{q}_1$ and $\dot{q}_2$:

$$q = [20, -20, 40, 0, 0]^T \text{ deg}$$
$$\dot{q}_1 = [100, 20, 50, 0, 10]^T \text{ deg/s}$$
$$\dot{q}_2 = \frac{1}{2} \dot{q}_1$$

Figure 7.3: Regions to be avoided around an industrial robot in motion with nominal velocity $\dot{q}_1$ (green) and $\dot{q}_2$ (light green).

7.3 Accounting for Uncertainty in Safety Constraints

A first extension of the mathematical formulation of the safety constraints has been realized in order to As a matter of fact, in real world scenarios the possibility to account for uncertainties of various kind is of utmost relevance.

Two major sources of uncertainty have been considered. The former is represented by measurement noise or inaccuracies affecting the sensors, while the latter consists in the approximations on the model for both the robot geometry and the obstacle(s) dimension. As a matter of fact the beam approximation does not allow to take into consideration the dimension of the different links, while the point-shaped model of the obstacle completely ignores the actual dimension of the obstacle itself.

The easiest way to account for these types of uncertainty is to introduce a clearance parameter $\Delta$ to allow the robot to stop at a certain distance (strictly greater than zero) from the obstacle. In other words, inequality (7.1), can be rewritten as follows:

$$velocity \cdot T_s \leq \max (0, \text{distance} - \Delta) \quad (7.13)$$
7.3. Accounting for Uncertainty in Safety Constraints

From a geometrical point-of-view, reducing the point-to-segment distance of a quantity \( \Delta \) is equivalent to either computing the distance between the segment representing the link and a sphere of radius \( \Delta \) centered in \( r_{\text{obst}} \) or to evaluating the distance between point \( r_{\text{obst}} \) and a capsule of radius \( \Delta \) built around the robot link, as shown in Fig. 7.4.

![Diagram of a rigid beam with obstacle and dual interpretation of the clearance parameter \( \Delta \).](image)

Figure 7.4: Rigid beam with obstacle and dual interpretation of the clearance parameter \( \Delta \).

Therefore, the clearance parameter \( \Delta \) can simultaneously be used to:

- take into account the size of the links of the robot;
- take into account the size of obstacle;
- account for measurement uncertainty;
- further denote an actual clearance distance.

It is worth noting that the clearance parameter \( \Delta \) could also be considered as a way to take into account the velocity of a moving obstacle, but in the case of human workers cooperating with the robot a better result is achieved by considering as obstacles the human swept volumes introduced in Chapter 6.

In order to formalize the safety constraints inequalities while taking the clearance parameter \( \Delta \) into account, we initially re-wrote inequalities (7.2) and (7.3) in the following way:

\[
\begin{align*}
    v_s^T \frac{r_{\text{obst}} - r_s}{\|r_{\text{obst}} - r_s\|} T_s &\leq \max (0, \|r_{\text{obst}} - r_s\| - \Delta) \\
    v_s^T (r_{\text{obst}} - r_s) T_s &\leq \max (0, \|r_{\text{obst}} - r_s\|^2 - \Delta \|r_{\text{obst}} - r_s\|)
\end{align*}
\]  

(7.14)

Moreover, notice that for the right hand side

\[
[\max (0, \|r_{\text{obst}} - r_s\| - \Delta)]^2 \leq \max (0, \|r_{\text{obst}} - r_s\|^2 - \Delta \|r_{\text{obst}} - r_s\|)
\]  

(7.15)

Consequently, function \( g(s) \) can be re-defined as

\[
g(s) = [\max (0, \|r_{\text{obst}} - r_s\| - \Delta)]^2
\]  

(7.16)

thus yielding:

\[
\min_s g(s) = [\max (0, \min_s \|r_{\text{obst}} - r_s\| - \Delta)]^2
\]  

(7.17)
where the term \( \min_s \| r_{\text{obst}} - r_s \| - \Delta \) represents, when it is positive, the distance between a sphere of radius \( \Delta \) centered in \( r_{\text{obst}} \) and the segment from \( r_a \) to \( r_b \). Finally, the right-hand side of the inequality (7.11) can be re-written as:

\[
f = \left[ \max \left(0, \min_s \| r_{\text{obst}} - r_s \| - \Delta \right) \right]^2 \left[ \begin{array}{c} 1 \\ 1 \end{array} \right]
\]

(7.18)

### 7.4 Safety Constraints for arbitrarily-shaped convex obstacles

So far, safety constraints have been formulated by considering only point-shaped obstacles. In order to account for more complex obstacles, like for instance work-pieces, tools or human workers cooperating with the robot, the mathematical formalization of safety constraints must be extended to the case of obstacles having more complex geometry. Consider for instance a generic polytopic obstacle \( O \) as shown in Fig. 7.5.

![Figure 7.5: A generic polytopic (convex) obstacle.](image)

The constraints to be enforced for such an obstacle can be written as follows

\[
T_s E (r_{\text{obst}}) \dot{q} \leq f (r_{\text{obst}}), \forall r_{\text{obst}} \in O
\]

(7.19)

The number of constraints to be enforced at run time is conceptually infinite, i.e. one per each point belonging to \( O \). However, some geometrical properties of the obstacle can be exploited in order to make the problem tractable.

A sufficient condition for (7.19) to be satisfied for all points \( r_{\text{obst}} \in O \) is

\[
T_s E (r_{\text{obst}}) \dot{q} \leq d \left[ \begin{array}{c} 1 \\ 1 \end{array} \right], \forall r_{\text{obst}} \in O
\]

(7.20)

where the right hand side term

\[
d = \min_{r_{\text{obst}} \in O} \| f (r_{\text{obst}}) \|_\infty
\]

(7.21)

represents the minimum distance between the link of the robot and the polytopic obstacle \( O \) and can be easily computed using the GJK algorithm [48]. Moreover, notice that the left hand side term is linear with respect to the parameter \( r_{\text{obst}} \in O \). Therefore the safety constraints regarding the pair link-obstacle can be written as follows

\[
T_s (r_{\text{obst}}^T E_0 + E_1) \dot{q} \leq d, \forall r_{\text{obst}} \in O
\]

(7.22)
7.4. Safety Constraints for arbitrarily-shaped convex obstacles

Since obstacle $\mathcal{O}$ is convex, every point belonging to $\mathcal{O}$ can be expressed as a linear combination of the vertices of the polytope representing $\mathcal{O}$:

$$\text{vert}(\mathcal{O}) = \{ \mathbf{r}_{\text{obst}}^v1, \mathbf{r}_{\text{obst}}^v2, \ldots, \mathbf{r}_{\text{obst}}^vN \}$$

$$\forall \mathbf{r}_{\text{obst}} \in \mathcal{O} \implies \exists a^1, a^2, \ldots, a^N \in \mathbb{R} | \sum_{i=1}^{N} a^i \cdot \mathbf{r}_{\text{obst}}^v_i = \mathbf{r}_{\text{obst}}$$

Consequently, the constraints associated to the generic point $\mathbf{r}_{\text{obst}}$ can be re-written substituting $\mathbf{r}_{\text{obst}}$ with a linear combination of the elements of $\text{vert}(\mathcal{O})$:

$$T_s \left( \sum_{i=1}^{N} a^i \cdot \mathbf{r}_{\text{obst}}^v_i \right)^T \mathbf{E}_0 + \mathbf{E}_1 \dot{q} \leq d, \forall \mathbf{r}_{\text{obst}} \in \mathcal{O} \quad (7.23)$$

thus obtaining a formulation that is linear with respect to the vertices of $\mathcal{O}$. Therefore, the set of constraints $$(7.22)$$ (which actually consists of an infinite number of scalar inequalities) can be equivalently expressed by formulating a finite number of inequalities that take into account only the elements of $\text{vert}(\mathcal{O})$:

$$T_s \left( \mathbf{r}_{\text{obst}}^T \mathbf{E}_0 + \mathbf{E}_1 \right) \dot{q} \leq d, \forall \mathbf{r}_{\text{obst}} \in \text{vert}(\mathcal{O}) \quad (7.24)$$
CHAPTER 8

Safety-oriented control strategies and trajectory generation algorithms

8.1 Introduction

As already stated in the previous chapters, one of the new challenges introduced by HRI consists in the possibility to satisfy at the same time both safety and productivity requirements. In other words, the human worker should not be harmed by the robot (see [49, 53]), but also production constraints, like for instance pre-programmed execution paths, should not be violated, in order to preserve the manipulator’s productivity.

This chapter presents two distinct trajectory generation algorithms for safe HRI both based on two fundamental elements:

- the motion and/or occupancy prediction strategies described in Chapter 6;
- the idea and the corresponding formalization of the Safety Constraints introduced in Chapter 7.

The first strategy consists in a kinematic scaling trajectory that adapts the robot speed along a pre-programmed path in such a way that the manipulator completely stops before colliding with calibrated obstacles and/or human workers moving inside the robotic cell. In this way a fruitful trade-off can be achieved between safety requirements and the manipulator productivity, especially in situations that require the pre-programmed path not to be modified and/or relaxed during task execution.

On the other hand the second approach consists in a point-to-point trajectory generation algorithm that takes as input a series of way points and produces a trajectory that is collision free with respect to obstacles and human workers. Differently from the previous one, this algorithm indeed allows the manipulator to deviate from the pre-
programmed trajectory in order to enforce the safety constraints, while minimizing this deviation and preserving the original task specification as much as possible.

Finally, a control architecture is proposed, that integrates the mentioned trajectory generation algorithms with the motion prediction strategy described in Chapter 5. In particular, the information regarding the estimated destination of a human walking trajectory is processed by a supervising Finite State Machine (FSM) that activates and/or deactivates the constraint-based reactive trajectory generation functionality.

8.2 Kinematic Scaling Algorithm

In order to solve the safety-aware trajectory motion planning problem, a kinematic scaling algorithm is proposed, whose block scheme is sketched in Figure 8.1. The algorithm suitably scales a pre-planned trajectory in time in order to guarantee that the robot completely stops before colliding with a generic obstacle \( O \). The compatibility of the trajectory scaling technique with respect to a closed-ended industrial controller is also discussed.

Assume the following well-known parametrization of the task with respect to time:

\[
\begin{align*}
\dot{x}(\tau) &= \frac{\partial x}{\partial \tau} \\
(8.1)
\end{align*}
\]

where \( \tau \) is the time variable and \( x(\cdot) \) is a differentiable task function specifying the desired trajectory. Given a specific value of \( \tau \) is it possible to “evaluate the motion primitives”, i.e. to determine the values of both \( x(\tau) \) and \( x'(\tau) \). Let

\[
\delta \in [0, 1] \\
(8.2)
\]

be a scalar quantity adopted to kinematically scale the trajectory in time. More in depth:

- \( \delta = 1 \implies \) nominal trajectory, i.e. path executed at programmed speed;
- \( \delta = 0 \implies \) the robot stops;

Similarly to the concept originally developed in [50], we introduce the following Linear Programming (LP) optimization problem:

\[
\max_{\delta, \dot{q}} \delta \\
(8.3a)
\]
8.2. Kinematic Scaling Algorithm

\[ T_sE(r_{obst}) \dot{q}\delta \leq d_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \forall r_{obst} \in \Omega \]  
(8.3b)

\[ J(q) \dot{q} = \delta x' \]  
(8.3c)

\[ 0 \leq \delta \leq 1 \]  
(8.3d)

where \( J(q) \) is the manipulator Jacobian matrix, \( T_s \) is the manipulator stopping time (see [38]), and the equality constraint (8.3c) guarantees that the robot does not deviate from the pre-programmed path. Moreover, notice that each pair link-obstacle is accounted for in the inequalities (8.3b), that correspond to the safety constraints already introduce in Chapter 7. It is worth noting that, since the prediction of human occupancy introduced in Chapter 6 consists in a set of convex volumes, each swept volume can be treated as a single convex obstacle when formulating the safety constraints.

As a matter of fact, the solution of the LP problem tends to maximize the throughput of the robot, while being consistent with safety requirements, thus resulting in a trade-off between safety and productivity. Notice that the problem always has a trivial solution:

\[ \delta = 0 \]

which guarantees its solvability in realistic applications.

In order to avoid chattering behaviour of variable \( \delta \) that would result in multiple activations and suspensions of the task, a hysterisis has been implemented: once \( \delta \) is set to zero, the output of the trajectory scaling algorithm is forced to zero, until the minimum distance exceeds a predefined threshold. Finally, the computed value of \( \delta \) is used to perform the update of the time variable \( \tau \):

\[ \tau_{k+1} = \tau_k + \Delta t \cdot \delta \]  
(8.4)

where \( \Delta t \) is the control algorithm cycle-time.

8.2.1 Experimental Validation

In the following a relevant case study is described and experimental results demonstrating the effectiveness of the proposed algorithm are presented. The robot has to accomplish a simple pick and place task while cooperating with the human worker by scaling the Tool Center Point velocity without disrupting the pre-programmed path.

Experimental Setup

The experimental setup is sketched in Figure 8.2. It consists of:

- ABB IRB 140 robot: a 6 axes industrial robot with 6 kg maximum payload. Its position controlled by an industrial ABB IRC 5 controller and programmed through the proprietary RAPID language. RAPID instructions allow to specify the programmed speed along the given path (see again [136]);
- Microsoft Kinect: a RGBD motion sensing camera developed by Microsoft. Kinect is used to detect the presence of a human worker and it allows to determine his/her kinematic configuration with respect to the full human kinematic model (2.10) introduced in Chapter 2.
Chapter 8. Safety-oriented control strategies and trajectory generation algorithms

- External PC: a real-time PC that acquires data from the Kinect sensor, computes the human swept volumes (using the approach based on joint positions and bounded joint velocities), acquires from the robot controller the kinematic configuration of the manipulator and evaluates the kinematic scaling parameter $\delta$ according to the distance between the swept volumes and the manipulator itself. The bi-directional communication between the External PC and the IRC5 robot controller is implemented with standard TCP/IP sockets.

It is worth noticing that this control strategy can be applied independently from the presence of an “open” controller that allows joint position and velocity references to be overridden. As a matter of fact, modern industrial controllers (like the ABB IRC5) have dedicated functionalities to implement kinematic scaling of pre-planned trajectories in real-time. In this case, the proposed methodology completely exploits the available functionalities (in particular kinematic inversion and online trajectory scaling) of the proprietary industrial controller while the only requirement is the presence of a real-time interface through which the External PC acquires joint position and velocity measures.

![Diagram representing the experimental setup](image)

**Figure 8.2:** Diagram representing the experimental setup

**Experimental Results**

In order to validate our trajectory scaling strategy the robot has been assigned a typical pick and place task consisting in picking the compact discs stored in the CD rack and releasing them into a green box. During task execution a human worker approaches the manipulator in order to substitute the green box containing the CDs released by the robot with an empty one, see Figure 8.3.

Figure 8.5 shows a graphical reconstruction of the human swept volumes computed during the experiment. On the other hand, Figure 8.4 clearly demonstrates the effectiveness of our approach: if the predicted human occupancy is far away from the manipulator, the kinematic scaling parameter $\delta$ is equal to 1 and the robot executes the...
Figure 8.3: Picture taking during the experiment showing the interaction between the human worker and the manipulator.

Figure 8.4: Validation Experiment: the first plot represents speed scaling (solid blue) and minimum distance (solid green), while the second plot shows robot Tool Center Point velocity (solid red).
	nominal trajectory. Whenever the human worker approaches the robot, the distance between his/her predicted occupancy and the manipulator rapidly decreases and consequently $\delta$ is lowered until it reaches zero. Once again Figure 8.4 shows that when the kinematic scaling parameter is equal to zero the manipulator completely stops allowing the human worker to safely interact with it (i.e. by substituting the green box). As
Chapter 8. Safety-oriented control strategies and trajectory generation algorithms

Figure 8.5: From top left to bottom right: graphical reconstruction of the human swept volumes computed during the experiment along with the robot.

mentioned before, $\delta$ is set back to 1 and the robot resumes the nominal trajectory as soon as the distance between the predicted human occupancy and the robot overcomes the predefined threshold. Video [105] integrates the experiment description.

8.3 Point-to-Point Trajectory Generation Algorithm

In the following, we present the trajectory generation algorithm that modifies the pre-programmed robot trajectory in order to ensure safe HRI. This algorithm consists of a QP problem which is solved at every time step in order to compute joint reference accelerations as control variables:

$$u_k = \ddot{q}_{k}^{ref} \quad (8.5)$$

Then, position and velocity references $q^{ref}_{k+1}$ and $\dot{q}^{ref}_{k+1}$, respectively, are updated on the basis of $u_k$ and sent to the lower level axis controller.

The QP problem is formulated as a tracking problem so that the manipulator tries to follow the pre-programmed trajectory as much as possible while respecting the safety constraints. Moreover, we consider also joint position, velocity and accelerations saturations and task space velocity limits, e.g. according to [5].

The complete QP problem is reported in (8.6a).

$$\min_{u_k} \left( \|\dot{x}_{k+1} - \dot{x}_{k+1}^{ref}\|_{Q_s}^2 + \|x_{k+1} - x_{k+1}^{ref}\|_{Q_p}^2 \right) \quad (8.6a)$$
8.3. Point-to-Point Trajectory Generation Algorithm

\[ x_{k+1} = x_k + \Delta t J_k \dot{q}_k + \frac{\Delta t^2}{2} \left( J_k \ddot{q}_k + J_k u_k \right) \]  
(8.6b)

\[ \dot{x}_{k+1} = J_k \dot{q}_k + \Delta t \left( J_k \ddot{q}_k + J_k u_k \right) \]  
(8.6c)

\[ q_{\text{inf}} \leq q_k + \Delta t \dot{q}_k + \frac{1}{2} \Delta t^2 u_k \leq q_{\text{sup}} \]  
(8.6d)

\[ \dot{q}_{\text{inf}} \leq \dot{q}_k + \Delta t u_k \leq \dot{q}^{\text{sup}} \]  
(8.6e)

\[ u_{\text{inf}} \leq u_k \leq u^{\text{sup}} \]  
(8.6f)

\[ -\ddot{x}^{\text{max}} \leq \dot{x}_{k+1} \leq \ddot{x}^{\text{max}} \]  
(8.6g)

\[ -\dddot{x}^{\text{max}} \leq J_k \ddot{q}_k + J_k u_k \leq \dddot{x}^{\text{max}} \]  
(8.6h)

\[ \forall V_i \in SV, \forall r^j_i \in V_i \implies E_k (r^j_i) (\dot{q}_k + \Delta t u_k) \leq f_k (r^j_i) \]  
(8.6i)

where:

- \( x_k^{\text{ref}} \) and \( \dot{x}_k^{\text{ref}} \) are the pre-programmed robot task-space position and velocity at time step \( k \);
- \( x_k \) and \( \dot{x}_k \) are the actual robot task-space position and velocity at time step \( k \);
- \( Q_p \) and \( Q_v \) are diagonal matrices acting as weights for the task-space position and velocity error inside the tracking cost function;
- \( q_k \) and \( \dot{q}_k \) are the actual robot joint-space position and velocity at time step \( k \);
- \( J_k \) stands for \( J (q_k) \);
- \( \Delta t \) is the robot controller sample time.

The cost function (8.6a) is meant to guarantee best tracking performance of the pre-programmed trajectory in terms of displacement and velocity. Constraints in (8.6b) and (8.6c) represent a second order Taylor approximation of the forward kinematics of the manipulator, while those in (8.6d), (8.6e) and (8.6f) implement joint position, velocity and acceleration limits, respectively, as well as (8.6h) and (8.6g) represent Cartesian acceleration and velocity limits. Finally, as discussed in Chapter 7, the safety constraints (8.6i) ensure that the robot is always able to stop before colliding with an obstacle or before entering the predicted occupancy of the human worker (see again Chapter 6).

Finally, position and velocity references at the next time step are obtained as follows:

\[ \dot{q}_k^{\text{ref}} = \dot{q}_k^{\text{ref}} + \Delta t u_k \]  
(8.7)

\[ q_k^{\text{ref}} = q_k^{\text{ref}} + \Delta t \dot{q}_k^{\text{ref}} + \frac{\Delta t^2}{2} u_k \]  
(8.8)

and sent to the lower level axis controller.
Chapter 8. Safety-oriented control strategies and trajectory generation algorithms

8.3.1 Experimental Validation

In this Section we describe a relevant case study and we present the result obtained by applying our proposed approach. The robot has to accomplish a simple pick and place task while cooperating with the human worker.

Experimental Setup

For the implementation and validation of the proposed trajectory generation algorithm, we here consider the experimental setup sketched in Figure 8.6. It consists of:

- ABB IRB 140 robot: a 6 axes industrial robot with 6 kg maximum payload. Its position is controlled by an industrial ABB IRC 5 controller connected to an External PC through an Ethernet-based interface;
- Microsoft Kinect and ASUS Xtion: two RGBD motion sensing cameras equipped with the OpenNI drivers [9] used to detect the presence of a human worker inside the robotic cell and to perform skeletal tracking;
- External PC #1: a workstation responsible to run the sensor fusion algorithm described in Chapter 4 and to compute the human swept volumes (using the approach based on joint positions, joint velocities, and bounded joint accelerations). This PC also stores the parameters needed to compute the swept volumes: position, velocity and accelerations upper and lower bounds along with the braking time $T_s$;
- External PC #2: a real-time PC that runs the trajectory generation algorithm on the basis of both the kinematic configuration of the robot and the predicted occupancy of the human worker. It is interfaced to External PC #1 with a standard Ethernet interface and it is connected to the IRC5 controller through a real-time Ethernet-based interface (see [20] for further details). It runs under Linux OS with the Xenomai patch, that makes it a hard real-time system. Using the real-time Ethernet-based interface, it is possible to develop and execute a control algorithm, named “External Controller”, that communicates in real-time with the IRC5 controller at a frequency of 250 Hz. Full-duplex real-time communication allows the External Controller to acquire data regarding the kinematic configuration of the manipulator and to override the joint position and velocity reference signals the IRC5 sends to the low-level joint controllers.

Notice that, differently from the previous case, this control strategy can be applied only in presence of an open controller that allows joint position and velocity references to be overridden.

Experimental Results

The point-to-point trajectory generation algorithm here proposed has been validated considering a typical pick-and-place scenario. The robot picks some metal pieces from a rack and deposits them inside a plastic box.

A first experiment is performed to validate the trajectory generation algorithm in presence of known static obstacles. The robot executes the task twice. During the first run no obstacle is located inside the manipulator workspace, while during the second
8.3. Point-to-Point Trajectory Generation Algorithm

Figure 8.6: Experimental setup - deployment diagram

one a plastic box is placed in the zone between the metal pieces rack and the deposit box, as it can be seen in Figure 8.7.

Figure 8.7: Experiment #1 - setup: the metal pieces placed on the rack and the deposit box are highlighted in green, the depth sensors in red and the known obstacle in yellow.

Figure 8.8 shows a graphical reconstruction of the two end-effector trajectories: the nominal one (in blue) and the one followed during the second execution (i.e. in presence of the obstacle, in red). As expected, the robot is able to accomplish the task without colliding with the obstacle thanks to the possibility to deviate from the nominal path. Moreover, Figure 8.9 demonstrates that as the minimum distance between the robot and the obstacle decreases, the control algorithm reduces the manipulator speed allowing it to keep executing the task while satisfying the safety constraints. Clearly, this is also the main reason why the time needed to complete the task increases with respect to the scenario without the obstacle.

A second experiment involving a human operator is here described. The robot executes the same pick-and-place under the supervision of a human worker standing inside its workspace, as depicted in Figure 8.10.

Once again, the robot accomplishes the task even though it has to deviate from the nominal trajectory in order to avoid collisions with the human worker. The graphical reconstruction of the two trajectories is shown in Figure 8.11. On the other hand,
Figure 8.8: Experiment #1. Graphic reconstruction of the nominal trajectory (blue line) and of the second trajectory (red line) followed in presence of the obstacle (light blue box). From the top left corner: 3D visualization, projection in the $X$-$Y$ plane, projection in the $X$-$Z$ plane and projection in the $Y$-$Z$ plane.

Figure 8.9: Experiment #1. Top plot: nominal end-effector Cartesian velocity ($\text{m/s}$, blue solid line). Bottom plot: end-effector Cartesian velocity in presence of the obstacle ($\text{m/s}$, blue solid line) versus the minimum robot-obstacle distance ($\text{m}$ dashed red line).

Figure 8.12 shows that, in order to satisfy the safety constraints, the control algorithm adapts the manipulator speed whenever it gets too close to the human.
8.4 Integrating prediction of human walking trajectories inside the Safety Controller

In this final section a control architecture is proposed that integrates the constraint-based reactive trajectory generation functionality with the motion prediction strategy described in Chapter 5.

As a matter of fact, both the kinematic scaling strategy and the point-to-point trajectory generation algorithm do rely on the prediction of human occupancy, but they both ignore another relevant information: the destination area that the human worker is reaching while walking inside the cell. Moreover, in structured industrial environment, HRI usually takes place in well-defined areas that are located nearby the robot or directly inside its workspace, while other specific areas are dedicated to the execution of manual task performed by the human operator on his/her own (see [19, 109] for instance). Consequently, it is reasonable to assume that the constraint-based reactive trajectory generation functionality should be activated only when the human is actually cooperating with the robot.

The proposed combined control architecture is sketched in Figure 8.13. At first, the information acquired by all the available RGB and RGB-D sensors is used to estimate the final destination of the worker’s walking trajectory (block “Intention Estimation”) and to predict his/her spatial occupancy (block “Swept Volumes”).

Then, the information regarding the estimated destination area is sent to a FSM that acts as a high-level supervisor. In particular, the FSM is composed of two states:

- **Human-Robot Coexistence**: the human worker executes a task that is completely independent from the one performed by the robot and that is carried out at a destination area which is outside (of far from) the manipulator workspace;

- **Human-Robot Interaction**: the human worker and the robot execute a cooperative task that requires the human operator to reach a destination area that is close (or directly inside) the manipulator workspace.
Chapter 8. Safety-oriented control strategies and trajectory generation algorithms

Figure 8.11: Experiment #2. Graphic reconstruction of the nominal trajectory (blue line) and of the second trajectory (red line) followed in presence of the human worker. From the top left corner: 3D visualization, projection in the $X$-$Y$ plane, projection in the $X$-$Z$ plane and projection in the $Y$-$Z$ plane.

Figure 8.12: Experiment #2. Top plot: nominal end-effector Cartesian velocity ($\text{[m/s]}$, blue solid line). Bottom plot: end-effector Cartesian velocity in presence of the human worker ($\text{[m/s]}$, blue solid line) versus the minimum human-robot distance ($\text{[m]}$ dashed red line).
8.4. Integrating prediction of human walking trajectories inside the Safety Controller

Each destination area is classified inside the FSM as either a coexistence area or an interaction area. Consequently, whenever the Intention Estimation block recognizes a specific destination, the FSM receives this information and possibly performs a state transition. More in depth, if the output of the Intention Estimation block corresponds to a coexistence area, the FSM will switch to the Coexistence state and viceversa.

The output of the FSM is a boolean variable, labelled “control flag” in Figure 8.13 that activates and deactivates the constraint-based reactive trajectory generation functionality. In other words, whenever the FSM is in the coexistence state, the flag deactivates the reactive algorithm and the joint positions and velocity references sent to the robot will be computed by the “Regular Task Execution” block that simply performs inverse kinematics on the basis of the pre-programmed trajectory. On the other hand, if the FSM is in the interaction state, the flag activates the “Constraint-based Task Execution” block that executes one of the two reactive algorithms previously introduced on the basis of the predicted human occupancy.

As a matter of fact this solution allows to maximize the robot productivity since nominal task execution is always preserved unless the supervising system recognizes that the human worker is going to perform some kind of cooperative tasks that requires the safety-oriented reactive algorithm to be activated. In the following an experiment is presented that shows the effectiveness of the proposed control architecture.

![Block scheme of the Safety Controller endowed with the Intention Estimation functionality.](figure8.13.png)

**Figure 8.13:** Block scheme of the Safety Controller endowed with the Intention Estimation functionality.
CHAPTER 9

Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

9.1 Introduction

While being able to offer fast and accurate task execution in various industrial fields, industrial robots are inherently characterized by a low degree of adaptability to rapidly changing task specifications. The fact that programming an industrial manipulator is a complex and time-consuming activity represents one of the main weaknesses of today’s industrial robotic systems, since it is preventing manipulators from being massively used in SMEs, where small size production and rapidly changing product features request the highest production flexibility.

The introduction of Lead-Through Programming (from now on “LTP”) approaches [18, 96] has definitely helped increasing the speed and reducing the complexity of the programming phase, by allowing the human operator to manually guide the robot in order to teach new positions. Nevertheless, these solutions present some drawbacks as well. Not only lead-through programming cannot guarantee the same level of accuracy and customization achievable with a teach pendant-based programming, but it can be also potentially unsafe since it requires physical Human-Robot Interaction (pHRI).

Although the possibility of constraining the movement of the robot during LTP has been addressed in the field of surgical robotics [60, 61, 84], the proposed approaches consist in simply defining a safety envelope from which the manipulator end-effector cannot exit, while it is guided by the human. Only at a later stage [76], the formalization of more generic constraints for lead-through programming of surgical robots was introduced.

Furthermore LTP relies on dedicated hardware, i.e. force/torques sensors. Although recently various manipulators have been designed with built-in force/torques sensors
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

and/or compliant joints \([35, 57, 95, 120, 121]\), the great majority of industrial manipulators is not inherently equipped with hardware that enables LTP. Adding a force/torque sensor to a standard industrial robot is a rather expensive and difficult operation.

In order to overcome this particular limitation, several approaches to the problem of sensorless detection of human-robot physical contact have been proposed in the literature, mainly based on fault detection and isolation algorithms fed with motor torque (or alternatively motor current) measurements \([128]\). As an example, the ABB RobotWare Software running on the IRC5 industrial controller includes this kind of feature \([116]\), while \([47]\) presents a safe collision detection and reaction strategy developed on an industrial manipulator with a closed control architecture.

Nevertheless, sensorless collision detection is not sufficient to completely overcome the need for dedicated hardware, since real-time estimation of the values achieved by interaction forces applied by the human operator to the manipulator (and not just their occurrence) is obviously necessary to perform sensorless LTP.

The problem of sensorless real-time estimation of external forces (or torques due to these forces) has been addressed in the literature, as well. The most relevant proposed solutions (like for instance \([34, 36, 40, 50, 80, 81]\)) rely on the calculation of residuals based on the manipulator generalized momentum. Typically estimation of external torques is used as input for impedance and/or admittance control strategies, like for instance in \([118, 119, 134]\). Other strategies for contact force estimation based on both friction estimation and detuning of the low-level joint control loops, have been proposed in \([77, 125, 126]\).

While several force/torque estimation strategies have been developed in order to avoid the use of dedicated sensors, the possibility to apply these strategies in the field of LTP has not been thoroughly explored yet. As a matter of fact, a complete and well-established framework for sensorless, accurate and safe LTP of a typical industrial manipulator in a structured environment is still missing. This possibility to teach an industrial robot by manually driving it without the need of dedicated hardware and taking into account motion accuracy and, above all, operator’s safety, definitely represents an interesting enhancement in both the fields of robot programming and pHRI.

In this sense, the LTP strategy here introduced combines the positive aspects of traditional teach pendant-based techniques (accuracy, high level of customization, safety and no need for additional hardware) with the advantages brought by lead-through techniques (reduced programming time and user friendliness), while mitigating their respective limitations.

The key innovative features of this LTP strategy can be summarized as follows:

- **Accuracy**: a voting system and a Finite State Machine (FSM) work together to select the largest Cartesian component of the forces/moments applied by the operator, thus allowing him/her to modify only one operational space degree of freedom at a time;

- **Safety**: the output of the admittance filter is processed by an optimization stage in order to satisfy actuation bounds, safety-related limits on operational space TCP velocity and avoidance of known obstacles. Moreover, a redundancy resolution algorithm ensures that the entire manipulator kinematic chain does not collide with workspace objects.
9.2 Manipulator Dynamic Model

The manipulator dynamic model is expressed, in Euler-Lagrange formulation, by the following equation:

\[
B(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) + F(\dot{q}) = \tau + \tau_{\text{ext}} \tag{9.1}
\]

where \(q, \dot{q}, \ddot{q}\) represent joint positions, velocities and accelerations, respectively, \(g\) is the gravity acceleration vector, \(B(q), C(q, \dot{q})\) and \(G(q)\) represent the inertia matrix, the Coriolis-centrifugal matrix and the gravitational vector, term respectively, \(F(\dot{q})\) is the function modelling friction torques, \(\tau\) denotes motor torques. \(\tau_{\text{ext}}\) represents the external interaction torques due to forces/moments \(\mu\) applied to the manipulator, given by the following relation:

\[
\tau_{\text{ext}} = J_C(q)^T \mu \tag{9.2}
\]

where \(J_C(q)\) is the Jacobian associated to the contact point \(C\).

DH kinematic parameters and masses of the links, gear ratios, CoG positions, inertia tensors and motor inertias are assumed to be known a priori from manufacturer data-sheets (Section 9.4.1 provides validation results that confirm the accuracy of these parameters for the given case). Standard state-of-the-art identification techniques exist in case such dynamic parameters are not known. Some considerations on the model and related identification for friction will be given in the following. Denoting with \(m\) the number of links composing the manipulator and considering the \(i\)-th link (with \(i \in \{1, \ldots, m\}\)), the identified friction model can be expressed in the following way:

\[
F_i(\dot{q}_i) = \begin{cases}
F_{vi}^- (\dot{q}_i - 0.01) - f_{si}^- & : \dot{q}_i \leq -0.01 \\
f_{si}^- (\dot{q}_i/0.01) & : -0.01 < \dot{q}_i \leq 0 \\
f_{si}^+ (\dot{q}_i/0.01) & : 0 < \dot{q}_i \leq 0.01 \\
F_{vi}^+ (\dot{q}_i + 0.01) + f_{si}^+ & : \dot{q}_i > 0.01
\end{cases} \tag{9.3}
\]
where $F^+_{vi}$ and $F^-_{vi}$ are the viscous friction coefficients for positive and negative joint velocities, respectively, while $f^+_{si}$ and $f^-_{si}$ are the static friction coefficients, again for positive and negative joint velocities. Notice that a linear approximation of the discontinuous static friction is adopted for joint velocities in a range

$$|\dot{q}_i| \leq 0.01$$

(9.4)

The friction model is linear with respect to the parameters. Consequently, all friction coefficients can be identified through the non-negative least squares minimization procedure described in the following.

An identification dataset consisting of $q$, $\dot{q}$ and $\tau$ is acquired by making the manipulator execute an ad-hoc program. Joint velocities and motor torques are filtered using a moving average filter. Joint accelerations $\ddot{q}$ are obtained differentiating filtered joint velocities, thus obtaining the following filtered measurements: $\dot{q}^{flt}$, $\ddot{q}^{flt}$, and $\tau^{flt}$.

For every time step $k$ the regression matrix $A_k$ can be expressed as a function of filtered joint velocities $\dot{q}^{flt}_k$, while the vector of measurements in the absence of torques due to external forces ($\tau_{ext} = 0$) is defined by:

$$Y_k = F(\dot{q}^{flt}_k) = \tau^{flt}_k - B(q_k)\ddot{q}^{flt}_k + C(q_k, \dot{q}^{flt}_k)\dot{q}^{flt}_k - G(q_k)$$

(9.5)

In order to minimize relative errors rather than absolute errors, each row of the regression matrix and of the measurement vector is divided by the maximum torque of the corresponding link. Stacking vertically the regression matrix and the measurement vector for every time instant we obtain:

$$Y = A \Pi$$

(9.6)

where $\Pi$ contains the friction parameters $F^+_{vi}$, $F^-_{vi}$, $f^+_{si}$ and $f^-_{si}$.

Finally, identification is performed via Least Square error minimization with non-negativity constraints on both viscous and static friction coefficients.

Although it is outside the scope of this work, it would be possible to consider more sophisticated friction models in order to improve the accuracy of our dynamic model, like for instance the friction observer proposed in [122] or the dithering feed-forward torque approach described in [127].

### 9.3 Lead-through Programming Algorithm

As mentioned before, the proposed LTP strategy represents a significant improvement with respect to state-of-the-art robot programming techniques not only because it overcomes the need for additional hardware, but also as it combines user-friendliness with motion accuracy, safety features and with the possibility of customizing robot movements by imposing different kinds of constraints. In the following a high level description of the proposed programming method is reported and each fundamental building block is thoroughly detailed.

First of all, considering the block scheme depicted in Figure 9.2 in order to perform sensorless LTP, applied external forces and moments need to be estimated on the basis
9.3. Lead-through Programming Algorithm

Figure 9.2: Block diagram of the proposed algorithm for sensorless, accurate and safe lead-through programming.

of all the information that can be directly acquired from the robot: joint positions, joint velocities and motor torques. Then, to achieve accuracy of the enforced motion, it is necessary to identify the largest Cartesian component of the estimated forces/moments in order to select the proper direction. To achieve this goal a voting system is introduced together with a Finite State Machine that outputs a projection of the estimated forces/moments according to the output of the voting system itself.

At this point, projected external forces/moments are transformed into operational space reference positions and velocities by using an admittance filtering techniques. Finally, the constraint-based controller computes the new reference values for joint positions and velocities by minimizing the error with respect to the output of the admittance filters while obeying to different types of constraints: actuation bounds, safety-inspired limits on operational space TCP velocity, and avoidance of known obstacles.

9.3.1 Online External Forces/Moments Estimation

In order to estimate the external forces/moments $\mu$ applied to the manipulator it is necessary to compute (or to estimate) in real-time the external torques $\tau_{ext}$. Unfortunately, while the Euler-Lagrange formulation of the dynamic model given in equation (9.1) can be successfully exploited to perform offline identification of unknown dynamic parameters, it is not possible to rely on this formulation for real-time estimation of interaction forces/moments, since accurate online computation of joint accelerations by numerical differentiation of joint velocities (or either from double numerical differentiation of joint positions) is not feasible.

Considering the generalized moments $p = B(q) \dot{q}$ and their derivative:

$$\dot{p} = C(q, \dot{q})^T \dot{q} - G(q) - F(\dot{q}) + \tau + \tau_{ext}$$  \hspace{1cm} (9.7)

it is possible to define the residuals \[36\] at time $t_k$ as

$$r(t_k) = K[p(0) + \int_0^{t_k} \left( \tau + C(q, \dot{q})^T \dot{q} - G(q) - F(\dot{q}) + r(t) \right) dt]$$  \hspace{1cm} (9.8)
where \( K \) is a diagonal positive definite matrix. By computing (9.8), one obtains a first order stable linear relationship between the external torques \( \tau_{ext} \) and the defined residual vector
\[
\dot{r} = K (\tau_{ext} - r)
\]  
(9.9)

Assuming all interaction forces/moments \( \mu \) are applied to the TCP frame, the following equation
\[
\mu = \left( J (q)^T \right) ^\dagger r
\]  
(9.10)
can be finally adopted to estimate the applied force/momentum \( \mu \), where \( J \) represents the Jacobian of the TCP frame, while the symbol \( A^\dagger \) stands for the Moore-Penrose pseudo-inverse of matrix \( A \). By exploiting the knowledge of the dynamic model and the measurement of the applied motor torques \( \tau \), it is possible to achieve an estimate, at time \( t_k \), of the applied interaction forces/moments \( \mu \) through (9.9) and (9.10).

For real-time implementation, the observer of the forces/moments is discretized in the following way:
\[
\begin{align*}
v_0 &= 0 \\
p_k &= B (q_k) \dot{q}_k \\
r_k &= (I + \Delta t K)^{-1} K [p_k - p_0 - \dot{p}_k \Delta t] \\
v_{k+1} &= v_k + (\dot{p}_k + r_k) \Delta t \\
\mu_k &= \left( J (q_k)^T \right) ^\dagger r_k
\end{align*}
\]  
(9.11)

where \( v_k \) is the state variable of the observer at time \( t_k \) and \( \Delta t \) is the discrete time step.

### 9.3.2 Voting System and Finite State Machine

The voting system and the Finite State Machine (FSM) represented in Figure 9.2 play a crucial role in the context of the proposed LTP strategy, since they allow to enforce accurate end-effector motion by selecting both the largest Cartesian component of the applied force/momentum and the corresponding Cartesian direction.

We assume that the human operator wants to move/rotate the end-effector along/around a certain Cartesian direction, while applying a force/momentum approximately directed along/around such direction. Therefore we identify the largest Cartesian component of the applied force/momentum and we force the manipulator end-effector to move/rotate exclusively along/around the corresponding Cartesian axis.

More in depth, the voting system consists in a circular buffer that collects votes. These votes are assigned to the three Cartesian axis \( X \), \( Y \) and \( Z \) on the basis of two distinct inputs: the estimated applied forces/moments \( \mu \) and a boolean flag, named “orientation”, that can be directly set by the human operator in order to enforce either translational or rotational motion. At each iteration, if \( \mu \) is equal to the null vector, a blank vote is cast and inserted into the buffer. Otherwise, if \( \text{orientation} \) is equal to 0, a vote is assigned to the axis corresponding to the largest component of the estimated force, while if \( \text{orientation} \) is set to 1, the vote is cast for the direction corresponding to the largest component of the applied momentum.

When a significantly large number of votes is assigned to the same Cartesian axis, we can assume that the operator wants to enforce a translational/rotational motion.
9.3. Lead-through Programming Algorithm

along/around that direction. Consequently, the voting system outputs a discrete variable, named “majority”, that can assume four different values: “null”, “X”, “Y”, “Z”. Obviously each value corresponds to a Cartesian axis, a part from null that means that either no force/momentum is exerted on the robot or that no Cartesian direction is assigned a sufficiently large number of votes.

More in detail, whenever majority is equal to null and the number of votes assigned to a Cartesian direction reaches a threshold value of maj_set votes, the value of majority is set to the corresponding axis. On the other hand, if majority is not equal to null and the number of votes assigned to the corresponding Cartesian axis decreases below the threshold value of maj_reset votes, majority is reset to null. Both maj_set and maj_reset are parametric thresholds that can be set by the human operator. Finally, whenever the operator modifies the value of the orientation flag, the buffer is emptied and majority is set to null.

Both orientation and majority act as input to the FSM, whose state-transition graph is represented in Figure 9.3. On the basis of its internal state, the FSM computes a specific value of the unit projection vector \( n \) (as reported in Figure 9.3), thus determining which component of the estimated applied force/momentum will be passed to the next stages of the programming algorithm.

Then the estimated applied forces/moments \( \mu \) are projected onto \( n \) to obtain a scalar quantity \( \mu^{prj} \):

\[
\mu^{prj} = n^T \mu
\]  

(9.12)

Clearly, both in State #0 and in State #1 the resulting projection vector zeroes all the estimated forces/moments, either stopping the manipulator or keeping it still until a new value of majority is set by the voting system. Otherwise, by sending both \( \mu^{prj} \) and \( n \) to the admittance filtering stage of the algorithm, it is possible to obtain an accurate translational/rotational motion of the end-effector along/around only one Cartesian axis at a time.

Necessarily, whenever the FSM is in State #1, and the operator wants to move the end-effector along a specific direction, the voting system has to wait for a new value of majority to be set before the FSM can switch to the desired State, thus introducing some delay between the human action and the robot reaction. This delay depends on the length of the buffer included in the voting system. During validation experiments we tested different buffer lengths: 100, 75 and 50 votes. Given these dimensions, the minimum time needed to establish an absolute majority among the votes is, respectively, 0.20 s, 0.15 s and 0.10 s.

It is worth noting that this delay is not sufficiently large to influence the human operator’s behaviour (see video [102] for a practical example). Moreover, the maximum delay only occurs when the operator wants to change the direction of motion and a new majority has to be established.

9.3.3 Admittance Filtering

The previously computed projection of the estimated applied forces/moments \( \mu^{prj} \) is processed by an admittance filtering stage in order to convert forces into linear velocities and moments into angular velocities. These, in turn, modify the end-effector operational space configuration \( \sigma \), defined as the output of the forward kinematics of
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

Figure 9.3: FSM state transition diagram.

the manipulator:

$$\mathbf{\sigma} = \mathbf{\sigma}(\mathbf{q}) = [x, y, z, \phi, \theta, \psi]^T$$ (9.13)

where $x$, $y$ and $z$ represent the end-effector position coordinates in the Cartesian space, while

$$\mathbf{\Phi} = [\phi, \theta, \psi]^T$$ (9.14)

is a set of Euler angles describing the end-effector orientation.

Following the formulation of the operational space admittance given in [43], two distinct admittance filter are considered: a positional admittance filter and a rotational one.

Positional Admittance Filtering

In case the orientation flag is equal to 0, the projection of the estimated applied forces/torques $\mu_{\text{prj}}$ corresponds to the principal Cartesian component of the applied external force and the positional admittance filter converts $\mu_{\text{prj}}$ in an operational space linear velocity reference $\dot{\mathbf{\sigma}}^{\text{ref}}$, determining an operational space translational displacement reference $\mathbf{\sigma}^{\text{ref}}$ along the Cartesian direction selected by the voting system and the FSM.

The admittance dynamic relation is defined as follows:

$$\mu_{\text{prj}} = M_p\dot{\mathbf{\sigma}}^{\text{ref}} + D_p\ddot{\mathbf{\sigma}}^{\text{ref}}$$ (9.15)

Obviously, the relation is stable, provided that constants $M_p$ and $D_p$ are positive.
9.3. Lead-through Programming Algorithm

Considering a second-order Taylor discretization, it is possible to rewrite the admittance relation as follows:

\[
\sigma_{k+1}^{ref} = \sigma_{k}^{ref} + \left( \Delta t - \frac{\Delta t^2}{2} \frac{D_p}{M_p} \right) \dot{\sigma}_{k}^{ref} + \frac{\Delta t^2}{2} \frac{1}{M_p} \mu_{k}^{prj} \quad (9.16)
\]

\[
\dot{\sigma}_{k+1}^{ref} = \left( 1 - \Delta t \frac{D_p}{M_p} \right) \dot{\sigma}_{k}^{ref} + \Delta t \frac{1}{M_p} \mu_{k}^{prj} \quad (9.17)
\]

where \( \mu_{k}^{prj} \) represents the projection of the estimated applied forces/torques computed at time \( t_k \) and \( \Delta t \) is the discrete time step.

Finally, the switching between different translational motions can be easily managed by the positional admittance filter, thanks to the FMS. As a matter of fact, in order to change the direction of the enforced motion, first the FSM internal state must be reset to State #1 and then it must be set to State #2, State #3, or State #4 (see again Figure 9.3) on the basis of the applied forces. Since the transition to State #1 requires the robot to stop, the new translational motion necessarily starts with the manipulator being still, thus allowing the admittance filter to handle the switching by simply setting both \( \dot{\sigma}_{k}^{ref} \) and \( \ddot{\sigma}_{k}^{ref} \) equal to zero, before starting to compute the new velocity and position references along the Cartesian direction selected by the new value of the projection vector \( n \).

Rotational Admittance Filtering

When the orientation flag is set to 1, the projection of the estimated applied forces/torques \( \mu_{k}^{prj} \) consists in the largest Cartesian component of the applied external momentum. Different approaches to the problem of defining a rotational impedance relation can be found in the literature, ranging from quaternion-based approaches [26, 28, 29] to techniques based on an angle-axis representation [27]. In this case, the rotational admittance filter converts \( \mu_{k}^{prj} \) in an operational space angular velocity reference \( \dot{\sigma}_{k}^{ref} \), determining an operational space angular displacement reference \( \sigma_{k}^{ref} \) for a specific Euler angle.

The rotational admittance dynamic relation can be defined in the following way:

\[
\mu_{k}^{prj} = M_\Phi \ddot{\sigma}_{k}^{ref} + D_\Phi \dot{\sigma}_{k}^{ref} \quad (9.18)
\]

Once again, the relation is stable when constants \( M_\Phi \) and \( D_\Phi \) are positive, but, differently from the positional case, there is no bijective relation between the Cartesian component of the applied torque and the Euler angles first-order derivatives \( \dot{\Phi} \). Consequently, it is not possible to use a unique representation of the end-effector frame orientation in order to convert an external torque applied on a specific Cartesian direction to an angular velocity of the frame around the same axis. Nevertheless, by using a different Euler angles representation for each possible rotation (along the \( X \), \( Y \) or \( Z \) axis) it is possible to establish a one-to-one correspondence between the end-effector angular velocity around the desired Cartesian axis and the first-order derivative of a specific Euler angle.

For instance, when the output of the FSM is equal to \( n = [0, 0, 0, 1, 0, 0]^T \), \( \mu_{k}^{prj} \) corresponds to the \( X \) axis component of the applied momentum. By defining the end-effector frame angular velocity \( \omega \) in the following way:

\[
\omega = [\omega_X, \omega_Y, \omega_Z]^T \quad (9.19)
\]
the end-effector frame rotation around the reference frame $X$-axis that must be enforced is described by:

$$\omega = [\omega_X, 0, 0]^T \quad (9.20)$$

Considering $XZX$ Euler angles $\Phi_{XZX}$, it is possible to express the angular velocity $\omega$ as a function of the Euler angles first-order derivatives $\dot{\Phi}_{XZX}$:

$$\Phi_{XZX} = [\phi_X, \theta_Z, \psi_X]^T$$
$$\dot{\Phi}_{XZX} = \begin{bmatrix} \dot{\phi}_X \\ \dot{\theta}_Z \\ \dot{\psi}_X \end{bmatrix}^T$$
$$\omega = W(\Phi_{XZX}) \dot{\Phi}_{XZX}$$

$$W(\Phi_{XZX}) = \begin{bmatrix} 1 & 0 & \cos \theta_Z \\ 0 & -\sin \phi_X & \cos \phi_X \sin \theta_Z \\ 0 & \cos \phi_X & \sin \phi_X \sin \theta_Z \end{bmatrix}$$

Consequently, by imposing $\dot{\theta}_Z = \dot{\psi}_X = 0$, $\omega_X$ is equal to:

$$\omega_X = [1, 0, \cos \theta_Z] \begin{bmatrix} \dot{\phi}_X \\ \dot{\theta}_Z \\ \dot{\psi}_X \end{bmatrix} = \dot{\phi}_X \implies \dot{\omega}_X = \ddot{\phi}_X \quad (9.21)$$

As a consequence a one-to-one correspondence is established between the end-effector angular velocity around the $X$ axis and the first-order derivative of a specific Euler angle, allowing us to set $\dot{\sigma}_{\text{ref}} = \dot{\phi}_X$ and $\ddot{\sigma}_{\text{ref}} = \ddot{\phi}_X$ inside the rotational admittance relation $\sigma_{\text{ref}}$ in order to generate the desired rotational motion.

Similarly, in case the motion to be enforced is a rotation of the end-effector frame around the $Y$ axis ($n = [0, 0, 0, 0, 0, 1]^T$), $YXY$ Euler angles are chosen, while when the desired motion consists in a rotation of the end-effector frame around the $Z$ axis ($n = [0, 0, 0, 0, 0, 1]^T$) the $ZYZ$ set of Euler angles is employed.

Once again, it is possible to discretize the rotational admittance relation $\sigma_{\text{ref}}$ by simply substituting $M_\phi$ and $D_\phi$ for $M_p$ and $D_p$, respectively, into equations $(9.16)$ and $(9.17)$. Finally, the rotational admittance filter manages the switching between different rotational motion by stopping the robot before a new rotation axis is selected, depending on the applied moments.

### 9.3.4 Lead-through programming algorithm and redundancy resolution

In the following, we present the whole algorithm developed for accurate lead-through programming, in presence of possible known obstacles. The algorithm consists of a quadratic programming (QP) problem which is solved at every time step in order to compute joint reference accelerations as control variables:

$$u_k = \ddot{q}_{\text{ref}}^k \quad (9.22)$$

Then, position and velocity references ($q_{k+1}^{\text{ref}}$ and $\dot{q}_{k+1}^{\text{ref}}$, respectively) are updated on the basis of $u_k$ and sent to the lower level axis controller. The goal of the QP problem is to allow best reference tracking of the outputs of the admittance filter $\sigma_{k+1}$ and
9.3. Lead-through Programming Algorithm

\[ \dot{\sigma}_{k+1} \text{ in (9.15) or (9.18), whilst guaranteeing the satisfaction of joint position, velocity and accelerations limits, task space velocity limits, e.g. according to [5], and obstacle avoidance.} \]

The complete QP problem is reported in (9.23).

\[
\begin{align*}
\min_{u_k, \dot{\sigma}_k} & \left( Q_v \left( \dot{\sigma}_{k+1} - \dot{\sigma}_{k+1}^{\text{ref}} \right)^2 + Q_p \left( \sigma_{k+1} - \sigma_{k+1}^{\text{ref}} \right)^2 \right) \\
J_k u_k + J_k \dot{q}_k^{\text{ref}} &= \Gamma \dot{\sigma}_k + K_d \dot{e}_k + K_p e_k \\
\dot{\sigma}_{k+1} &= \dot{\sigma}_k + \Delta t \ddot{\sigma}_k \\
J_k &= J \left( q_k^{\text{ref}} \right) \\
e_k &= \sigma_k^{\text{ref}} - \sigma \left( q_k^{\text{ref}} \right) \\
\dot{e}_k &= \dot{\sigma}_k^{\text{ref}} - J_k \ddot{q}_k^{\text{ref}} \\
\Gamma &= \begin{cases} n & \text{State #0, #1, #2, #3, #4} \\ [0 0 0 1 0 0]^T & \text{otherwise} \end{cases}
\end{align*}
\]

where:

\[
\sigma_{k+1} = \sigma_k + \Delta t \dot{\sigma}_k + \frac{\Delta t^2}{2} \ddot{\sigma}_k \\
\dot{\sigma}_{k+1} = \dot{\sigma}_k + \Delta t \ddot{\sigma}_k \\
J_k = J \left( q_k^{\text{ref}} \right) \\
e_k &= \sigma_k^{\text{ref}} - \sigma \left( q_k^{\text{ref}} \right) \\
\dot{e}_k &= \dot{\sigma}_k^{\text{ref}} - J_k \ddot{q}_k^{\text{ref}} \\
\Gamma &= \begin{cases} n & \text{State #0, #1, #2, #3, #4} \\ [0 0 0 1 0 0]^T & \text{otherwise} \end{cases}
\]

The cost function (9.23a) is meant to guarantee best tracking performance of the output of the admittance filter (in terms of displacement and velocity). Constraints in (9.23b) represent a second order CLIK inverse kinematics algorithm, while those in (9.23c) implement joint position, velocity and acceleration limits, as well as (9.23d) and (9.23e) represent Cartesian acceleration and velocity limits. Finally, as discussed in Chapter [7], the inequalities in (9.23f) account for possible obstacles within the robot workspace to be avoided. Notice that the obstacle avoidance constraints have higher priority than the admittance rule in the cost function. Hence, in case the human operator tends to push the robot TCP against an obstacle, the QP algorithm will automatically decrease the velocity of the robot to avoid any possible collision between the robot and any workspace obstacle.

In addition, in case of kinematic redundancy of the robot, several solutions of the QP problem in (9.23) exist, differing in the null-space components of the optimal solution \( u_k \). For this reason, another optimization layer can be adopted to further optimize such components. In particular, following the approach in [64], the QP problem in (9.24) can be introduced.

\[
\begin{align*}
\min_{u_k} & \left( \frac{1}{2} \Delta t u_k^T u_k + \dot{q}_k^T u_k \right) \\
J_k u_k + J_k \dot{q}_k^{\text{ref}} &= \Gamma \dot{\sigma}_k + K_d \dot{e}_k + K_p e_k \\
\dot{\sigma}_{k+1} &= \dot{\sigma}_k + \Delta t \ddot{\sigma}_k \\
J_k &= J \left( q_k^{\text{ref}} \right) \\
e_k &= \sigma_k^{\text{ref}} - \sigma \left( q_k^{\text{ref}} \right) \\
\dot{e}_k &= \dot{\sigma}_k^{\text{ref}} - J_k \ddot{q}_k^{\text{ref}} \\
\Gamma &= \begin{cases} n & \text{State #0, #1, #2, #3, #4} \\ [0 0 0 1 0 0]^T & \text{otherwise} \end{cases}
\end{align*}
\]

99
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

<table>
<thead>
<tr>
<th>qpOases call Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean [ms]</td>
</tr>
<tr>
<td>std  [ms]</td>
</tr>
<tr>
<td>min [ms]</td>
</tr>
<tr>
<td>max [ms]</td>
</tr>
</tbody>
</table>

Table 9.1: Optimization stage execution time: average execution time, standard deviation, minimum and maximum time.

\[ J_k u_k = J_k^0 u_k^0 \]  
all the constraints in (9.23c) and (9.23f). (9.24c)

where \( u_k^0 \) is any of the solutions of the QP problem in (9.23). The goal of this second QP problem is to preserve the optimality of the first problem, through the enforcement of constraint (9.24b), whilst selecting the smallest null space velocity compatible with all the other constraints.

Finally, position and velocity references at the next time step are obtained as follows:

\[
\dot{q}_{k+1}^{ref} = \dot{q}_{k}^{ref} + \Delta t u_k \\
q_{k+1}^{ref} = q_{k}^{ref} + \Delta t \dot{q}_{k}^{ref} + \frac{\Delta t^2}{2} u_k
\]

and sent to the lower level axis controller.

Notice that, in case of saturation due to activation of constraints, the robot end-effector might not be able to follow the references computed by the admittance filter. For this reason, the output of the direct kinematics is sent back to the filter to avoid wind-up phenomena (see again Figure 9.2). The state of the admittance filter is then updated as

\[
\sigma_{k+1}^{ref} \leftarrow \Gamma^T \sigma_{k+1} = \Gamma^T \sigma \left( q_{k+1}^{ref} \right) \\
\dot{\sigma}_{k+1}^{ref} \leftarrow \Gamma^T \dot{\sigma}_{k+1} = \Gamma^T \dot{J} \left( q_{k+1}^{ref} \right) \dot{q}_{k+1}^{ref}
\]

9.4 Experiments

For the implementation of the proposed lead-through programming strategy, we here consider the 14-DOF dual-arm redundant robot prototype ABB FRIDA (see again Figure 9.1a), equipped with a control system based on ABB IRC5 industrial controller, located inside its torso.

In order to solve the QP problems (9.23) and (9.24), we use the “qpOASES” solver [41][42]. As far as the computational burden of the optimization stage is concerned, Figure 9.4 shows the measured execution times of more than 25,000 calls to qpOASES organized in a histogram. On the other hand, Table 9.1 provides some more details regarding average, minimum and maximum execution time. Given these data, we can state that the QP solver can easily solve the optimization problems within the 4 ms cycle time of the real-time external controller.

In the following, we first provide details regarding the identification of the manipulator unknown dynamic parameters (along with results of a validation experiment of
9.4. Experiments

![Histogram showing the measured execution times of more than 25,000 calls to qpOASES.](image)

**Figure 9.4:** Histogram showing the measured execution times of more than 25,000 calls to qpOASES.

<table>
<thead>
<tr>
<th>Joint</th>
<th>Avg. Error [%]</th>
<th>Max Error [%]</th>
<th>Std. Deviation [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5.2867</td>
<td>12.00</td>
<td>4.90</td>
</tr>
<tr>
<td>2</td>
<td>-7.7000</td>
<td>10.00</td>
<td>3.31</td>
</tr>
<tr>
<td>3</td>
<td>+6.2283</td>
<td>15.00</td>
<td>6.35</td>
</tr>
<tr>
<td>4</td>
<td>-3.1350</td>
<td>12.00</td>
<td>4.73</td>
</tr>
<tr>
<td>5</td>
<td>-2.5000</td>
<td>15.00</td>
<td>7.55</td>
</tr>
<tr>
<td>6</td>
<td>+1.1867</td>
<td>15.00</td>
<td>7.05</td>
</tr>
<tr>
<td>7</td>
<td>-3.7667</td>
<td>15.00</td>
<td>6.43</td>
</tr>
</tbody>
</table>

**Table 9.2:** Results of FRIDA’s Dynamic Model Validation. For each joint: average error, maximum absolute error and standard deviation of the error with respect to maximum joint torques.

The inertial, centrifugal and gravitational terms of FRIDA’s dynamic model have been assembled on the basis of the dynamic parameters provided by the manufacturer. In order to determine viscous and static friction coefficients, first an identification dataset has been acquired and then the identification procedure described in Section 9.2 has been performed.

Table 9.2 summarizes the results of a validation experiment performed on a validation dataset. Validation error statistics show the accuracy of the identified model and demonstrate that in this case it is absolutely reasonable to rely on manufacturer’s datasheet for building the inertial, centrifugal and gravitational terms of the dynamic model.

9.4.1 Unknown Dynamic Parameter Identification

The inertial, centrifugal and gravitational terms of FRIDA’s dynamic model have been assembled on the basis of the dynamic parameters provided by the manufacturer. In order to determine viscous and static friction coefficients, first an identification dataset has been acquired and then the identification procedure described in Section 9.2 has been performed.

Table 9.2 summarizes the results of a validation experiment performed on a validation dataset. Validation error statistics show the accuracy of the identified model and demonstrate that in this case it is absolutely reasonable to rely on manufacturer’s datasheet for building the inertial, centrifugal and gravitational terms of the dynamic model.

9.4.2 Experimental Validation

The lead-through robot programming approach described in the previous Sections has been experimentally tested considering three different scenarios: LTP without obstacles, LTP without obstacles and reduced velocity limits, and finally LTP in the presence of obstacles. A fourth and final experiment provides results obtained from standard
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

LTP techniques, demonstrating that the proposed LTP strategy guarantees superior accuracy. In the following, each considered scenario is detailed and the corresponding experimental results are presented.

Experiment #1: lead-through without obstacles

In the first experiment, the robot is driven by the human worker without considering any known obstacle. The initial experimental setup is represented in Figure 9.1a.

At first the “orientation flag” is set to 0 and the human worker can modify only the end-effector Cartesian position. The estimated external forces applied by the human worker to the manipulator and the linear end-effector velocity components resulting from the lead-through programming algorithm are shown in Figure 9.5.

From the collected data it is clear that the lead-through programming algorithm correctly identifies the main component of the force applied by the human operator to the end-effector, allowing it to move only along the selected Cartesian direction.

Then, the human operator sets the orientation flag to 1 and starts modifying the end-effector orientation by applying moments. The estimation of these moments is shown in Figure 9.6 along with the end-effector angular velocity components resulting from the lead-through programming algorithm.
9.4. Experiments

Figure 9.6: Experiment #1. Top plot - estimation of external moments applied to the robot: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). Bottom plot - end-effector angular velocity: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, Stop, State #5 - Rotation around X, State #6 - Rotation around Y, State #7 - Rotation around Z.

Once again, experimental data prove that the algorithm successfully identifies the main component of the applied momentum, allowing the operator to enforce a rotation only along the selected Cartesian axis. Moreover these data demonstrate that our approach to lead-through programming provides a level of accuracy of the end-effector motion comparable to teach pendant-based programming while being definitely more user-friendly.

Finally, Figure 9.7a shows that the end-effector linear velocity norm never overcomes the maximum limit of 0.250 m/s, while it can be seen in Figure 9.7b that the end-effector angular velocity norm is always lower than the maximum limit of 180.00 °/s.

Experiment #2: lead-through without obstacles and reduced velocity limits

A second experiment is performed with the human operator modifying the end-effector orientation at first and then varying the end-effector position. Differently from the previous experiment, the maximum end-effector linear and angular velocity limits are set to 0.05 m/s and 5.00 °/s, respectively.

Estimated external forces are shown in Figure 9.8, while the estimation of applied moments is depicted in Figure 9.9. Not only Figure 9.8 and Figure 9.9 demonstrate that, once again, the algorithm successfully selects the right direction for both linear and rotational motion on the basis of the largest Cartesian component of the applied external forces and/or moments, but Figure 9.10a and Figure 9.10b clearly show that
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

Figure 9.7: Experiment #1. Fig. 9.7a end-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line). Fig. 9.7b end-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line).

the end-effector linear and angular velocities saturate at the imposed limits, regardless of the magnitude of the applied force/momentum. This result demonstrates how safety aspects have been successfully incorporated in the proposed lead-through programming strategy by allowing the operator to set these velocity limits as configuration parameters for the algorithm.

Experiment #3: lead-through in presence of obstacles

In order to demonstrate the effectiveness of the proposed lead-through programming strategy in terms of obstacle avoidance, the experimental setup is modified by introducing an obstacle inside the manipulator workspace (the white box shown in Figure 9.1b). This obstacle is calibrated by measuring its position with respect to the robot base frame and by computing a simplified triangular mesh describing its geometry. In addition, a yellow and red stop button (see again Figure 9.1b) is introduced as a goal that the manipulator has to reach.

The effectiveness of the obstacle avoidance constraints is documented in video [102] and it is also demonstrated by several screenshots of the experiment shown in Figure 9.11. While the human operator successfully guides the robot end-effector to the goal, the kinematic redundancy of the manipulator is exploited in order to comply with the obstacle avoidance constraints and to obtain a collision-free motion of the entire manipulator.

These results not only demonstrate that the programming algorithm can be easily configured in order to consider zero or more calibrated obstacles, but also show that kinematic redundancy is successfully exploited by the algorithm to ensure collision free motion of the entire manipulator during lead-through.

Experiment #4: unconstrained lead-through

In order to prove the accuracy of our approach with respect to standard LTP techniques, in the last experiment we disabled the voting system, the FSM and the optimization stage. The results shown in Figure 9.12 demonstrate that, even though the force ap-

104
9.4. Experiments

Figure 9.8: Experiment #2. Top plot - estimation of external forces applied to the robot: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). Central plot - end-effector linear velocity: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). Bottom plot - end-effector Cartesian position: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, State #2 - Translation along X, State #3 - Translation along Y, State #4 - Translation along Z.

plied by the operator is always characterized by one Cartesian component significantly greater than the other ones, the physical interaction determines displacements of the end-effector position along all the Cartesian axes, thus producing a substantially less accurate motion.

As a matter of fact, accuracy issues necessarily arise when performing traditional LTP, since it is very difficult (if not impossible) for a human operator to apply a force/momentum exactly aligned with a specific Cartesian direction. For example, consider the first row in Figure 9.8, which exemplifies the approach phase of a picking task (to be grasped from the red tray). Using a traditional LTP method, it would be difficult to maintain the alignment of the TCP with the object to be grasped. In turn, with the approached proposed in this paper, the human operator is only responsible for moving the robot towards the object, whilst the alignment is automatically guaranteed by the controller.
Chapter 9. Safety in physical HRI: an approach to safe and accurate Lead-Through Programming

Figure 9.9: Experiment #2. Top plot - estimation of external moments applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector angular velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, Stop, State #5 - Rotation around X, State #6 - Rotation around Y, State #7 - Rotation around Z.

Figure 9.10: Experiment #2. Fig. 9.10a: end-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line). Fig. 9.10b: end-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line).
Figure 9.11: Experiment #3: the manipulator is safely guided by the human operator towards its goal, without colliding with the obstacle (white box).
Figure 9.12: Experiment #4. Top plot - estimation of external moments applied to the robot: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line). Bottom plot - end-effector angular velocity: X component (solid black line), Y component (solid dark grey line), Z component (solid light grey line).
CHAPTER 10

Conclusions

This thesis aims at giving a contribution to overcome current industrial robot controllers limitations in order to achieve safe and efficient Human-Robot Interaction during the execution of cooperative tasks. A twofold approach is followed. At first, the robot control system is endowed with more sophisticated perception capabilities by integrating various kinds of exteroceptive sensors, especially RGB cameras and depth cameras. Data acquired from these sensors are merged together and they are used in combination with different kinematic models to track and predict the motion of human workers inside the supervised robotic cell. Then, several safety-oriented control strategies and trajectory generation algorithms are proposed that rely on the information supplied by the perception system in order to guarantee human worker’s safety while trying to maximize the productivity of the manipulator during the execution of a cooperative task.

In Part I the problem of increasing the perception capabilities of the robot controller has been approached with a specific focus on two relevant aspects: human motion tracking and sensor fusion. After having introduced different kinematic models describing human motion (Chapter 2), a first motion tracking algorithm has been proposed that is able to track the walking motion of human workers inside a robotic cell by merging together multiple RGB images acquired by several fish-eye RGB surveillance cameras (Chapter 3). The algorithm has been proven capable of tracking more than one person simultaneously and of tracking human motion even in presence of a moving robot. Then a second algorithm has been introduced, that relies on the information acquired by several depth sensors (Chapter 4). Not only this motion tracking strategy is able to track both walking and upper limbs motion (according to the full human kinematic model), but it is also capable of meeting strict real-time execution requirements imposed by the robot controller, thus providing a fast and reliable solution to the problem
Chapter 10. Conclusions

of human motion tracking in a supervised robotic cell.

Moving on to Part II, another fundamental aspect of the perception problem has been taken into account: human motion prediction. At first the problem of predicting the final destination of a human worker’s walking trajectory inside a robotic cell has been addressed (Chapter 5), by proposing a solution able to perform correct predictions while simultaneously tracking one or more human workers moving inside the robotic cell. On the other hand, in order to realize a prediction that takes into account the full human kinematic model, a methodology to compute the entire swept volume that a human worker can occupy within a defined time span has been proposed and successfully validated (Chapter 6).

Finally, in Part III the focus has switched from the perception perspective to the control one. In particular, several safety-oriented control strategies and trajectory generation algorithms for safe HRI have been presented and discussed. Starting from an extension of the safety constraints (Chapter 7), two distinct approaches to the safety-aware trajectory generation problem have been proposed (Chapter 8). The first one consists in a constraint-based kinematic scaling strategy that preserves the geometry of the pre-programmed task while guaranteeing collision avoidance with respect to the human operator. The second approach consists in a point-to-point trajectory generation algorithm that allows the robot to deviate from the task to better enforce safety constraints. A more general control architecture is also proposed, that integrates the mentioned trajectory generation algorithms with the motion prediction strategy described in Chapter 5, in order to maximize the robot productivity while enforcing safety constraints only when it is necessary. Finally, a novel Lead-Through Programming is proposed (Chapter 9) that has been proven able to guarantee human safety and programming accuracy without relying on dedicated hardware.

Future Developments

In this thesis, the problem of safe HRI has been tackled starting from the perception domain and moving towards the control field. The motion tracking and prediction strategies developed in this research have been successfully validated and fruitfully integrated inside safety-oriented control and trajectory generation algorithms.

The most promising possible extension is represented by the possibility to develop new kind of safety constraints to be seamlessly integrated into the control algorithms here proposed. While the different formalizations of the safety constraints here described are mainly inspired by the minimum separation distance criterion, it could be possible to formulate new kinds of constraints based of different safety requirements and taking into account physical quantities other than distance. For instance it could be possible to formulate constraints that limit the robot kinetic energy during the execution of a cooperative task or that monitor the manipulator momentum during its motion in presence of human workers.

Another possible development is represented by the possibility to extend the perception system by integrating new kind of sensors, like for instance laser scanners and distance/proximity sensors, into the motion tacking algorithm here proposed.
# List of Figures

1.1 Traditional industrial robotic cell with safety barriers. ................................. 2
1.2 HRI scenario where human workers and robot work directly in touch. ................ 2

2.1 Representation of the non-holonomic kinematic model describing human walking motion. ....................................................... 11
2.2 Kinematic model of the human (right) arm and torso flexion/extension angle $\rho$. ................................................................. 12
2.3 Complete human kinematic model: DOFs, frames and bodies. .......................... 13
2.4 Approximated Human Skeleton. ...................................................................... 14

3.1 Example of a setup including two calibrated cameras and corresponding image planes. ................................................................... 16
3.2 Example of multiple camera image fusion. Left: image acquired from camera #01. Middle: image acquired from camera #02. Right: combined image resulting from image fusion. ................................................ 17
3.3 Example of Single Camera BG/FG Segmentation. Left: input image. Middle: foreground mask. Right: foreground image. ......................... 18
3.4 Example of k-d tree where non-leaf nodes represent splitting hyperplanes and leaf nodes consist in actual 2-dimensional points. .......... 19
3.5 Left: human worker detected silhouette (grey), pose and circumscribed box according to the explained motion model. Middle: graphic output of a particle motion model state considering a single camera. Right: pixel determining the particle’s value considering a single camera. .................. 20
3.6 The experimental setup comprising the two AXIS fish-eye surveillance cameras, the PC running the HDT application and the Ethernet connection between the different components. The entrance area is highlighted in yellow and the two walking paths and the corresponding destination areas are highlighted respectively in black and grey. ........................................ 22
3.7 Graph showing that the human position estimate computed by the HDT System (dashed red) is always included inside the path drawn on the ground (dashed black). ........................................................................... 23
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>Graph showing that the human worker position estimates computed by the HDT System (dashed red and dashed green) are always included inside the corresponding paths drawn on the ground (respectively dashed black and dashed grey).</td>
</tr>
<tr>
<td>3.9</td>
<td>Screenshots taken from Experiment #1 showing that a single human worker is correctly tracked by the HDT System.</td>
</tr>
<tr>
<td>3.10</td>
<td>Screenshots taken from Experiment #2. The first human worker is tracked by a red box, while the other is tracked by a green box.</td>
</tr>
<tr>
<td>3.11</td>
<td>Screenshots taken from Experiment #3 showing that a single human worker is correctly tracked by the HDT System while the robot motion is ignored.</td>
</tr>
<tr>
<td>4.1</td>
<td>A scheme of the experimental setup comprising the two depth sensors, the data acquisition processes and the main sensor fusion/motion tracking process.</td>
</tr>
<tr>
<td>4.2</td>
<td>Tracking experiment - human joint positions: data acquired from depth sensor #1 (red solid line), data acquired from depth sensor #2 (green solid line), estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).</td>
</tr>
<tr>
<td>4.3</td>
<td>Tracking experiment - human joint velocities: estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).</td>
</tr>
<tr>
<td>4.4</td>
<td>Tracking experiment - human joint accelerations: estimated values (blue solid line) and corresponding lower and upper bounds (dashed blue lines).</td>
</tr>
<tr>
<td>4.5</td>
<td>Tracking experiment - human kinematic model parameters: data acquired from depth sensor #1 (red solid line), data acquired from depth sensor #2 (green solid line) and values estimated by the tracking algorithm (blue solid line).</td>
</tr>
<tr>
<td>4.6</td>
<td>Tracking experiment - linear velocity along the X-axis: numerical differentiation of data acquired from depth sensor #1 (red solid line), numerical differentiation of data acquired from depth sensor #2 (green solid line) and values estimated by the tracking algorithm (blue solid line).</td>
</tr>
<tr>
<td>5.1</td>
<td>Robotic cell layout: entrance (black solid line, marked with 'X'), destination areas (red, green, blue and yellow solid lines, marked with 'X'), destination paths (red, green, blue and yellow dashed lines) and robots/obstacles (cyan solid lines, marked with 'X').</td>
</tr>
<tr>
<td>5.2</td>
<td>Computed trajectories inside cell layout (red, green, blue and yellow dotted lines).</td>
</tr>
<tr>
<td>5.3</td>
<td>Gaussian Mixture Regression of computed trajectories (red, green, blue and yellow drawings) plus undefined goal points Gaussian Mixture Model (magenta ellipses).</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiment 01: tracked trajectories (red, green, blue and yellow solid lines) with points corresponding to prediction times (red, green, blue and yellow large circles). Experiment 02: tracked trajectory (magenta solid line) with point corresponding to prediction time (magenta large circle).</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiment 01: probabilities of the different destinations (coloured dashed lines) and threshold probability (black dashed line). From top to bottom: RED destination, GREEN destination, BLUE destination and YELLOW destination.</td>
</tr>
<tr>
<td>5.6</td>
<td>Experiment 02: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the tracked trajectory.</td>
</tr>
<tr>
<td>5.7</td>
<td>Experiment 03: first switching trajectory (black solid lines) and second one (black dashed line) with points corresponding to prediction times (red, magenta, green and blue large circles).</td>
</tr>
<tr>
<td>5.8</td>
<td>Experiment 03: probabilities of the different destinations (red, green, blue and yellow dashed lines), probability of the undefined goal (magenta dashed lines) and threshold probability (black dashed lines) during the execution of the tracked trajectories.</td>
</tr>
<tr>
<td>5.9</td>
<td>Experiment 04: trajectory of the first human worker (black solid lines) and trajectory of the second human worker with points corresponding to prediction times (red and blue large circles).</td>
</tr>
<tr>
<td>5.10</td>
<td>Experiment 04: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the first tracked trajectory (destination = RED).</td>
</tr>
<tr>
<td>5.11</td>
<td>Experiment 04: probabilities of the different destinations (red, green, blue and yellow dashed line), probability of the undefined goal (magenta dashed line) and threshold probability (black dashed line) during the execution of the first tracked trajectory (destination = BLUE).</td>
</tr>
<tr>
<td>6.1</td>
<td>Unicycle reachable set (dashed) and its convex approximation (solid bold)</td>
</tr>
<tr>
<td>6.2</td>
<td>Unicycle reachable set (dashed) and its convex approximation (solid bold)</td>
</tr>
<tr>
<td>6.3</td>
<td>Figure 6.3a: translational swept volume of convex object. Figure 6.3b: rotational swept volume of convex object. The corresponding convex hulls are also highlighted.</td>
</tr>
<tr>
<td>6.4</td>
<td>Triangular approximation of a circular arc</td>
</tr>
<tr>
<td>6.5</td>
<td>Example of swept volumes computed on the basis of joint positions and fixed bounds on joint velocities</td>
</tr>
<tr>
<td>6.6</td>
<td>Example of swept volumes computed on the basis of joint positions and velocities and fixed bounds on joint accelerations.</td>
</tr>
<tr>
<td>6.7</td>
<td>Graphic reconstruction of the human kinematic configuration versus the human swept volumes computed at time ( T = 0 ). First row: human kinematic configuration at time ( T = 0 ). Second row: human kinematic configuration at time ( T = \frac{1}{2} T_s ). Third row: human kinematic configuration at time ( T = \frac{1}{4} T_s ). Fourth row: human kinematic configuration at time ( T = \frac{3}{4} T_s ). For each row, from left to right, 3D view, projection in the X-Z plane and projection in the X-Y plane.</td>
</tr>
<tr>
<td>7.1</td>
<td>Trade-off between productivity and safety.</td>
</tr>
<tr>
<td>7.2</td>
<td>A rigid beam representing one link and a point-shaped obstacle.</td>
</tr>
</tbody>
</table>
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>Regions to be avoided around an industrial robot in motion with nominal velocity $\dot{q}_1$ (green) and $\dot{q}_2$ (light green).</td>
</tr>
<tr>
<td>7.4</td>
<td>Rigid beam with obstacle and dual interpretation of the clearance parameter $\Delta$.</td>
</tr>
<tr>
<td>7.5</td>
<td>A generic polytopic (convex) obstacle.</td>
</tr>
<tr>
<td>8.1</td>
<td>Kinematic scaling algorithm block scheme</td>
</tr>
<tr>
<td>8.2</td>
<td>Diagram representing the experimental setup</td>
</tr>
<tr>
<td>8.3</td>
<td>Picture taking during the experiment showing the interaction between the human worker and the manipulator.</td>
</tr>
<tr>
<td>8.4</td>
<td>Validation Experiment: the first plot represents speed scaling (solid blue) and minimum distance (solid green), while the second plot shows robot Tool Center Point velocity (solid red).</td>
</tr>
<tr>
<td>8.5</td>
<td>From top left to bottom right: graphical reconstruction of the human swept volumes computed during the experiment along with the robot.</td>
</tr>
<tr>
<td>8.6</td>
<td>Experimental setup - deployment diagram</td>
</tr>
<tr>
<td>8.7</td>
<td>Experiment #1 - setup: the metal pieces placed on the rack and the deposit box are highlighted in green, the depth sensors in red and the known obstacle in yellow.</td>
</tr>
<tr>
<td>8.8</td>
<td>Experiment #1. Graphic reconstruction of the nominal trajectory (blue line) and of the second trajectory (red line) followed in presence of the obstacle (light blue box). From the top left corner: 3D visualization, projection in the $X$-$Y$ plane, projection in the $X$-$Z$ plane and projection in the $Y$-$Z$ plane.</td>
</tr>
<tr>
<td>8.9</td>
<td>Experiment #1. Top plot: nominal end-effector Cartesian velocity ($m/s$, blue solid line). Bottom plot: end-effector Cartesian velocity in presence of the obstacle ($m/s$, blue solid line) versus the minimum robot-obstacle distance ($m$ dashed red line).</td>
</tr>
<tr>
<td>8.10</td>
<td>Experiment #2 - setup: the static obstacle is removed and the human worker stands near the robot during the task execution.</td>
</tr>
<tr>
<td>8.11</td>
<td>Experiment #2. Graphic reconstruction of the nominal trajectory (blue line) and of the second trajectory (red line) followed in presence of the human worker. From the top left corner: 3D visualization, projection in the $X$-$Y$ plane, projection in the $X$-$Z$ plane and projection in the $Y$-$Z$ plane.</td>
</tr>
<tr>
<td>8.12</td>
<td>Experiment #2. Top plot: nominal end-effector Cartesian velocity ($m/s$, blue solid line). Bottom plot: end-effector Cartesian velocity in presence of the human worker ($m/s$, blue solid line) versus the minimum human-robot distance ($m$ dashed red line).</td>
</tr>
<tr>
<td>8.13</td>
<td>Block scheme of the Safety Controller endowed with the Intention Estimation functionality.</td>
</tr>
<tr>
<td>9.1</td>
<td>Fig. 9.1a: ABB prototype robot FRIDA and experimental setup for Experiment #1, Experiment #2, and Experiment #4. Fig. 9.1b: modified experimental setup for Experiment #3 with calibrated obstacle and goal object.</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>Block diagram of the proposed algorithm for sensorless, accurate and safe lead-through programming</td>
</tr>
<tr>
<td>9.3</td>
<td>FSM state transition diagram</td>
</tr>
<tr>
<td>9.4</td>
<td>Histogram showing the measured execution times of more than 25,000 calls to qpOASES</td>
</tr>
<tr>
<td>9.5</td>
<td>Experiment #1. Top plot - estimation of external forces applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Central plot - end-effector linear velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector Cartesian position: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, State #2 - Translation along X, State #3 - Translation along Y, State #4 - Translation along Z.</td>
</tr>
<tr>
<td>9.6</td>
<td>Experiment #1. Top plot - estimation of external moments applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector angular velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, Stop, State #5 - Rotation around X, State #6 - Rotation around Y, State #7 - Rotation around Z.</td>
</tr>
<tr>
<td>9.7</td>
<td>Experiment #1. Fig. 9.7a: end-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line). Fig. 9.7b: end-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line).</td>
</tr>
<tr>
<td>9.8</td>
<td>Experiment #2. Top plot - estimation of external forces applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Central plot - end-effector linear velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector Cartesian position: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, State #2 - Translation along X, State #3 - Translation along Y, State #4 - Translation along Z.</td>
</tr>
<tr>
<td>9.9</td>
<td>Experiment #2. Top plot - estimation of external moments applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector angular velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). The corresponding FSM state is reported on top of the plots: State #1 - Stop, Stop, State #5 - Rotation around X, State #6 - Rotation around Y, State #7 - Rotation around Z.</td>
</tr>
</tbody>
</table>
List of Figures

9.10 Experiment #2. Fig. 9.10a: end-effector linear velocity norm (solid grey line) with respect to linear velocity upper bound (dashed black line). Fig. 9.10b: end-effector angular velocity norm (solid grey line) with respect to angular velocity upper bound (dashed black line). . . . . . . 106

9.11 Experiment #3: the manipulator is safely guided by the human operator towards its goal, without colliding with the obstacle (white box). . . . . 107

9.12 Experiment #4. Top plot - estimation of external moments applied to the robot: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). Bottom plot - end-effector angular velocity: $X$ component (solid black line), $Y$ component (solid dark grey line), $Z$ component (solid light grey line). . . . . . . . 108
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Experiment 01: Arrival time and corresponding prediction time for each tracked trajectory</td>
<td>47</td>
</tr>
<tr>
<td>9.1</td>
<td>Optimization stage execution time: average execution time, standard deviation, minimum and maximum time.</td>
<td>100</td>
</tr>
<tr>
<td>9.2</td>
<td>Results of FRIDA's Dynamic Model Validation. For each joint: average error, maximum absolute error and standard deviation of the error with respect to maximum joint torques</td>
<td>101</td>
</tr>
</tbody>
</table>
Bibliography

Bibliography


Bibliography


[102] MERLIN Lab POLIMI. Accurate sensorless lead-through programming for lightweight robots in structured environments. [https://www.youtube.com/watch?v=FyWhNzgJ19c](https://www.youtube.com/watch?v=FyWhNzgJ19c).

[103] MERLIN Lab POLIMI. Detecting, tracking and predicting human motion inside an industrial robotic cell using a map-based particle filtering strategy. [https://www.youtube.com/watch?v=gRQpYLQpRQ](https://www.youtube.com/watch?v=gRQpYLQpRQ).
Bibliography

[104] MERLIN Lab POLIMI. Multiple camera human detection and tracking inside a robotic cell - an approach based on image warping, computer vision, k-d trees and particle filtering. [https://www.youtube.com/watch?v=LceApT0hXc](https://www.youtube.com/watch?v=LceApT0hXc)

[105] MERLIN Lab POLIMI. Safety-aware trajectory scaling for human-robot collaboration with prediction of human occupancy. [https://www.youtube.com/watch?v=Z7DgzHC9e9E](https://www.youtube.com/watch?v=Z7DgzHC9e9E)


Appendix A - Forward Human Kinematics

In this appendix, the calculations that allow to compute the forward kinematic of model (2.10) are detailed. Considering the complete kinematic model (2.10), forward kinematics can be defined as:

\[
[T \ N \ H \ LS \ RS \ LE \ RE \ LW \ RW]^T = f_{\text{kine}}(p, \pi)
\]

where points \( T, H, LS, RS, LE, RE, LW, RW \) correspond to the skeletal representation of the kinematic model given in Figure 2.4, \( p \) is the vector containing the complete kinematic configuration of the human (see (2.9)) and \( \pi \) is the vector containing the kinematic parameters specific to each human being:

- \( d_{G,T} \): height of the thorax point \( T \) with respect to the ground plane;
- \( d_{T,N} \): distance between thorax point \( T \) and neck point \( N \);
- \( d_{N,H} \): distance between neck point \( N \) and head point \( H \);
- \( d_{N,RS} \): distance between neck point \( N \) and right shoulder point \( RS \);
- \( d_{N,LS} \): distance between neck point \( N \) and left shoulder point \( LS \);
- \( d_{RS,RE} \): distance between right shoulder point \( RS \) and right elbow point \( RE \);
- \( d_{LS,LE} \): distance between left shoulder point \( LS \) and left elbow point \( LE \);
- \( d_{RE,RW} \): distance between right elbow point \( RE \) and right wrist point \( RW \);
- \( d_{LE,LW} \): distance between left elbow point \( LE \) and left wrist point \( LW \);

To properly determine the skeletal points, several intermediate matrices must be computed and then post-multiplied. In the following each intermediate matrix is detailed.

Matrix \( T_{11} \) - rotation due to \( \theta \):

\[
T_{11} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 & 0 \\
\sin(\theta) & \cos(\theta) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Appendix A. Appendix A - Forward Human Kinematics

Matrix $T_{t2}$ - translation due to $x$ and $y$:

$$T_{t2} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{t3}$ - from ground plane to thorax point $T$:

$$T_{t3} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & d_{G,T} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{t4}$ - rotation due to $\rho$ - from torso point $T$ to neck point $N$:

$$T_{t4} = \begin{bmatrix} \cos (\rho) & -\sin (\rho) & 0 & d_{T,N} \cos (\rho) \\ \sin (\rho) & -\cos (\rho) & 0 & d_{T,N} \sin (\rho) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{t5}$ - rigid transformation from neck point $N$ to head point $H$:

$$T_{t5} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{r0}$ - rigid transformation from neck point $N$ to right shoulder point $RS$:

$$T_{r0} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -d_{N,RS} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{r1}$ - rotation due to $\alpha_{1}^{right}$:

$$T_{r1} = \begin{bmatrix} \cos (\alpha_{1}^{right}) & -\sin (\alpha_{1}^{right}) & 0 & 0 \\ \sin (\alpha_{1}^{right}) & \cos (\alpha_{1}^{right}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{r2}$ - rotation due to $\alpha_{2}^{right}$:

$$T_{r2} = \begin{bmatrix} \cos (\alpha_{2}^{right}) & 0 & \sin (\alpha_{2}^{right}) & 0 \\ \sin (\alpha_{2}^{right}) & 0 & -\cos (\alpha_{2}^{right}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $T_{r3}$ - rotation due to $\alpha_{3}^{right}$ - right elbow point $RE$:

$$T_{r3} = \begin{bmatrix} \cos (\alpha_{3}^{right}) & 0 & -\sin (\alpha_{3}^{right}) & 0 \\ \sin (\alpha_{3}^{right}) & 0 & \cos (\alpha_{3}^{right}) & 0 \\ 0 & 1 & 0 & d_{RS,RE} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
Matrix $T_{r4}$ - rotation due to $\alpha_{4}^{\text{right}}$:

$$T_{r4} = \begin{bmatrix}
\cos(\alpha_{4}^{\text{right}}) & 0 & -\sin(\alpha_{4}^{\text{right}}) & 0 \\
\sin(\alpha_{4}^{\text{right}}) & 0 & \cos(\alpha_{4}^{\text{right}}) & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{r5}$ - rigid transformation from right elbow point $\text{RE}$ to right wrist point $\text{RW}$:

$$T_{r5} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 1 & 0 & d_{\text{RE,RW}} \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l0}$ - rigid transformation from neck point $\text{N}$ to left shoulder point $\text{LS}$:

$$T_{l0} = \begin{bmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & d_{\text{N,LS}} \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
\cos(\alpha_{1}^{\text{left}}) & 0 & \sin(\alpha_{1}^{\text{left}}) & 0 \\
\sin(\alpha_{1}^{\text{left}}) & 0 & -\cos(\alpha_{1}^{\text{left}}) & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l1}$ - rotation due to $\alpha_{1}^{\text{left}}$:

$$T_{l1} = \begin{bmatrix}
\cos(\alpha_{1}^{\text{left}}) & -\sin(\alpha_{1}^{\text{left}}) & 0 & 0 \\
\sin(\alpha_{1}^{\text{left}}) & -\cos(\alpha_{1}^{\text{left}}) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l2}$ - rotation due to $\alpha_{2}^{\text{left}}$:

$$T_{l2} = \begin{bmatrix}
\cos(\alpha_{2}^{\text{left}}) & 0 & \sin(\alpha_{2}^{\text{left}}) & 0 \\
\sin(\alpha_{2}^{\text{left}}) & 0 & -\cos(\alpha_{2}^{\text{left}}) & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l3}$ - rotation due to $\alpha_{3}^{\text{left}}$ - left elbow point $\text{LE}$:

$$T_{l3} = \begin{bmatrix}
\cos(\alpha_{3}^{\text{left}}) & 0 & -\sin(\alpha_{3}^{\text{left}}) & 0 \\
\sin(\alpha_{3}^{\text{left}}) & 0 & \cos(\alpha_{3}^{\text{left}}) & d_{\text{LS,LE}} \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l4}$ - rotation due to $\alpha_{4}^{\text{left}}$:

$$T_{l4} = \begin{bmatrix}
\cos(\alpha_{4}^{\text{left}}) & 0 & -\sin(\alpha_{4}^{\text{left}}) & 0 \\
\sin(\alpha_{4}^{\text{left}}) & 0 & \cos(\alpha_{4}^{\text{left}}) & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

Matrix $T_{l5}$ - rigid transformation from left elbow point $\text{LE}$ to left wrist point $\text{LW}$:

$$T_{l5} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 1 & 0 & d_{\text{LE,LW}} \\
0 & 0 & 0 & 1
\end{bmatrix}$$
Finally, by introducing the origin point $O$:

$$O = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$$

all the skeletal points can be obtained by computing:

$$T = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot O$$
$$N = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot O$$
$$H = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t5} \cdot O$$
$$RS = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t6} \cdot O$$
$$LS = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t0} \cdot O$$
$$RE = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t6} \cdot T_{t7} \cdot T_{t8} \cdot T_{t9} \cdot O$$
$$LE = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t0} \cdot T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot O$$
$$RW = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t6} \cdot T_{t7} \cdot T_{t8} \cdot T_{t9} \cdot T_{t0} \cdot O$$
$$LW = T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t0} \cdot T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t5} \cdot O$$
Appendix B - Inverse Human Kinematics

In this appendix, the calculations that allow to compute the inverse kinematic of model (2.10) are detailed. Considering the complete kinematic model (2.10), inverse kinematics can be defined as:

\[ [p, \pi] = f_{\text{kin}}^{-1} \left( \begin{bmatrix} T & H & LS & RS & LE & RE & LW & RW \end{bmatrix}^T \right) \]

where:

- \( p \) is the vector containing the complete kinematic configuration of the human (see (2.9));
- \( \pi \) is the vector containing the kinematic parameters specific to each human being (see Appendix A for further details);
- points \( T, H, LS, RS, LE, RE, LW, RW \) correspond to the skeletal representation of the kinematic model given in Figure 2.4.

The first step of the proposed closed-form inverse kinematics algorithm consists in the determination of the kinematic parameters contained in \( \pi \). To this purpose, let all the skeletal points be given in terms of homogeneous coordinates, like for instance:

\[ T = \begin{bmatrix} T_x & T_y & T_z & 1 \end{bmatrix}^T \]

Consequently, the kinematic parameters can be computed as:

\[
\begin{align*}
    d_{G,T} & = T_z \\
    d_{T,N} & = \| N - T \| \\
    d_{N,H} & = \| H - N \| \\
    d_{N,RS} & = \| RS - N \| \\
    d_{N,LS} & = \| LS - N \| \\
    d_{RS,RE} & = \| RE - RS \| \\
    d_{LS,LE} & = \| LE - LS \| \\
    d_{RE,RW} & = \| RW - RE \| \\
    d_{LE,LW} & = \| LW - LE \|
\end{align*}
\]
Appendix B. Appendix B - Inverse Human Kinematics

Having determined the components of vector \( \pi \), the kinematic configuration of the human being \( p \) can be computed in the following way:

\[
\begin{align*}
\theta &= \tan^{-1} \left( \frac{(LS_y - RS_y), (LS_x - RS_x)}{\pi/2} \right) \\
x &= \cos \theta \cdot T_x + \sin \theta \cdot T_y \\
y &= \cos \theta \cdot T_y - \sin \theta \cdot T_x \\
\rho &= \tan^{-1} \left( \sqrt{(N_x - T_x)^2 + (N_y - T_y)^2}, (N_z - T_z) \right)
\end{align*}
\]

As far as the arm kinematics are concerned, at first it is necessary to express the elbow points with respect to the corresponding shoulder frame:

\[
\begin{align*}
RE^{RS} &= (T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{r0})^{-1} \cdot RE \\
LE^{LS} &= (T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t0})^{-1} \cdot LE 
\end{align*}
\]

where \( T_{t1}, T_{t2}, T_{t3}, T_{t4}, T_{r0}, T_{t0} \) are the forward kinematics matrices (see Appendix A) that depend on the already computed DOFs \( x, y, \theta, \rho \). Consequently, the first two arm angles can be determined as:

\[
\begin{align*}
\alpha_1^{\text{right}} &= \tan^{-1} \left( -RE_y^{RS}, -RE_x^{RS} \right) \\
\alpha_2^{\text{right}} &= \tan^{-1} \left( -RE_x^{RS}, \sqrt{(RE_x^{RS})^2 + (RE_y^{RS})^2} \right) \\
\alpha_1^{\text{left}} &= \tan^{-1} \left( -LE_y^{LS}, -LE_x^{LS} \right) \\
\alpha_2^{\text{left}} &= \tan^{-1} \left( -LE_x^{LS}, \sqrt{(LE_x^{LS})^2 + (LE_y^{LS})^2} \right)
\end{align*}
\]

Then, wrist points must be express with respect to the corresponding elbow frame:

\[
\begin{align*}
RW^{RE} &= (T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{r0} \cdot T_{r1} \cdot T_{r2} \cdot T_{r3})^{-1} \cdot RE \\
LW^{LE} &= (T_{t1} \cdot T_{t2} \cdot T_{t3} \cdot T_{t4} \cdot T_{t0} \cdot T_{t1} \cdot T_{t2} \cdot T_{t3})^{-1} \cdot LE
\end{align*}
\]

where, once again, the intermediate matrices depend on the already determined DOFs. Finally, the last two arm angles can be computed as:

\[
\begin{align*}
\alpha_3^{\text{right}} &= \tan^{-1} \left( RW_y^{RE}, RW_x^{RE} \right) \\
\alpha_4^{\text{right}} &= \tan^{-1} \left( \sqrt{(RW_x^{RE})^2 + (RW_y^{RE})^2}, (d_{RS,RE} - RW_x^{RE}) \right) \\
\alpha_3^{\text{left}} &= \tan^{-1} \left( LW_y^{LE}, LW_x^{LE} \right) \\
\alpha_4^{\text{left}} &= \tan^{-1} \left( \sqrt{(LW_x^{LE})^2 + (LW_y^{LE})^2}, (d_{LS,LE} - LW_x^{LE}) \right)
\end{align*}
\]