Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Mechanical Engineering (33rd Cycle)

Real-Time Trajectory Planning for Human-Friendly Collaborative Robotics

By

Matteo Melchiorre

******

Supervisor(s):
Prof. Stefano Mauro, Supervisor
Prof. Stefano Pastorelli, Co-Supervisor

Doctoral Examination Committee:
Prof. Benedetto Allotta, Referee, Università degli Studi di Firenze
Prof. Francesco Braghin, Referee, Politecnico di Milano

Politecnico di Torino
2021

Declaration

I hereby declare that the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Matteo Melchiorre

2021

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

To my family

Preface

This dissertation synthesizes the major outcomes of a work lasting more than 3 years that has been conducted on the side of my colleague and friend Leonardo Sabatino Scimmi. We spent a lot of time together researching on collaborative robotics with the aim to make human-robot collaboration the closest possible. We started by following the state of the art and we moved on by proposing some novel solutions that we hope can inspire researchers in this field.

In some ways, this thesis is complementary to the previous work by Leonardo Sabatino Scimmi and presents the best advances of my studies, that have been mainly focused on robot control strategies for a human-friendly collaboration. The title of the dissertation addresses this point and anticipates that the solutions here proposed fit the applications where a fast response of the robot to the human intentions is required.

For better readability, because the dissertation treats different aspects and applications of human-robot collaboration, the contents are organized by topics, distinguishing two main problems: collision avoidance and hand-over.

The work proposes original solutions to the two problems at different levels: some novelties recall the algorithms presented by Leonardo Sabatino Scimmi in his PhD dissertation and introduce important advances, others have a significant theoretical contribution.

In particular, the central part of the thesis is dedicated to a novel method to control the robot trajectory during collision avoidance using artificial potential fields with multiple attractors.

Abstract

Collaborative robotics identifies human-robot collaboration in the industrial field. It is realized by removing physical fences between human workers and robotic systems in favor of a shared workspace. In this way, human and robot can work side by side to accomplish a task.

Robots designed for this type of application are called "collaborative robots" and are mostly in the form of anthropomorphic manipulators. For a safe and effective collaboration, collaborative robots must respond in real-time to the operator's movements. This work describes a solution based on 3D vision systems and novel algorithms for real-time trajectory planning, specifically designed for the highest level of collaboration. Two possible applications are described: collision avoidance and hand-over.

Collision avoidance algorithms allow the robot to avoid obstacles (objects or the human operator) so that the trajectories are safe and ergonomic. This work presents two collision avoidance algorithms. One has minor improvements with respect to the state of the art. It drives the entire robot body away from obstacles using repulsive actions, expressed in terms of repulsive velocities. The other is a major contribution to the current state of knowledge. It exploits artificial potential fields with multiple attractors to obtain a controlled collision-free motion of the end effector.

Hand-over algorithms allow the bidirectional exchange of objects between operator and robot. A reactive hand-over is obtained by driving the robot towards the human hand, which can give or receive objects in any point of the robot workspace. This work describes an improved algorithm that combine results of previous works with a predictive scheme to anticipate human hand motion. The method also considers ergonomics and is designed to orient the robot tool according to the upper limb of the worker.

Each algorithm uses robot and human position data to calculate robot commands in terms of joint velocities. The actual configuration of the robot is

obtained by direct kinematics from feedback signals. Human motion is tracked by a 3D vision system based on multiple Microsoft Kinect cameras. Sensor fusion techniques are implemented to optimize human position data by means of skeleton as well as point cloud.

Depending on the grade of novelty, the developed algorithms are verified either by simulation or experimental test. The simulation environment is built in Matlab, considering a kinematic model of the collaborative robot. The experimental tests are carried out with a robotic cell composed of a collaborative robot UR3 from Universal Robots and the 3D vision system. The results show collision avoidance and hand-over algorithms performances in different scenarios.

A real-world application of collision avoidance based on repulsive velocities is presented. The experimental test with the robotic cell proves that the method can safely drive an assembly task. The robot collects the parts to be mounted while the human prepares the assembly. The operator can access anytime the robot workspace. His position is monitored with skeleton tracking. If the distance from the robot is within a defined threshold, the robot reacts with collision-free trajectories. Compared to the lowest level of collaboration, where the robot stops to prevent hazards, the proposed system optimizes productivity by saving up to the 18% of the cycle time.

Collision avoidance based on repulsive velocities is also tested with human point cloud instead of skeleton. Simulation results show that the methodology is effective and produces safe robot trajectories. However, the experimental test indicates some limitations due to the large amount of data to be processed in real-time.

Collision avoidance based on artificial potential fields with multiple attractors is verified in the simulation environment and validated with the real robot. The obstacles are combined with local attractors opportunely placed to influence the robot collision-free trajectory along preferred direction. Results show that the end-effector avoid obstacles choosing the side of the local attractor. This can be used to control the robot behavior when dealing with obstacles, so that trajectories can be predicted by the operator.

A hand-over task is tested with the robotic cell. The improved hand-over algorithm can quickly adjust the robot pose according to the human forearm. The prediction scheme ensures a more natural hand-over by reducing the waiting time of the operator.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decays the role of the robot in the society has been redefined due to the scientific and technological progress. The trend is to drop the paradigm of the robot as separate identity and to propose a new scenario where it is paired with people in every-day life. Nowadays, robots are matched with people in many applications. There are robots meant for assisting people and others designed to collaborate with them.

Human-robot collaboration (HRC) is a field whose importance is increasing day-by-day. A recent study shows that the number of publications on the topic of HRC from 1996 to 2015 has grown exponentially [1]. The authors in [1] give also an interesting definition: “Human-robot collaboration is defined when human(s), robot(s) and the environment come to contact with each other and form a tightly coupled dynamical system to accomplish a task”. In this context, the contact is not necessarily physical but is intended as a time-spatial engagement that requires mental and physical coordination. The effect of this engagement realizes in the environment, where human flexibility and robot reliability meet to accomplish the task. The word “reliability” synthesizes repeatability, precision and accuracy of the robot. The robot skills, combined with human flexibility, can deal with most complex applications. A meaningful example is the use of robots in the field of medicine, that has brought significant improvement in minimally invasive robotics surgery (Figure 1.1a). Of course, robots intended for collaboration are not standard ones: they are designed to facilitate the interaction with humans and to increase safety.

Figure 1.1 a) Minimally invasive robotic surgery by NHS [2]. b) Collaborative robotics by DIMEAS (Politecnico di Torino) and Ferrero S.p.A.

1.1 Collaborative robotics

In the industrial field, the HRC is identified as *collaborative robotics*. The basic idea is to ensure safe robotic systems to allow the removal of the fences that divide the working area of the robots from that of the human operators. In this way, robots and humans can work together to accomplish a task (Figure 1.1b). One of the first studies that introduced this concept is [3]. The authors of [3] referred to a mechanically compliant device indicated as *cobot* (from *collaborative robot*) and designed to help the worker maneuvering payloads safely and accurately through virtual surfaces. Since then, collaborative robotics has been extended to various industrial applications. An insightful overview of the most significant works in this field until 2009 is given by [4], while the up-to-date review [5] covers the last applications and aspects.

A factor that has marked the evolution of collaborative robotics in the last two decays is the refining of the capabilities of the cobot. Today, collaborative robots are mostly anthropomorphic manipulators specifically designed to increase the chances for a tight collaboration. They are equipped with passive and active safety features. The passive features are design choices that intrinsically reduce the risk of injury in the case of undesired contact, such us lightweight materials and rounded edges. The most advanced collaborative robots are equipped also with a set of sensors with dedicated software packages to actively handle unexpected contacts by stopping the robot if the joint torques or the contact forces exceed defined thresholds. Some of the most iconic examples of collaborative robots are shown in Figure 1.2. However, features that come with the cobot out of the box may not guarantee safe collaborative robotics. In fact, the workspace sharing can introduce new hazards and risks for the workers that must be taken into account when designing the application.

Figure 1.2 From left to right: collaborative robots LBR iiwa 14 from KUKA [6], YuMi - IRB 14000 from ABB [7] and UR5 from Universal Robots [8]

The design of the collaborative robotics systems is principally related to the task. In fact, according to the International Federation of Robotics (IFR), one may require a different level of interaction depending on human and robot activities [9]. The most common examples of collaborative robotics provide a shared workspace where the robot and employee work sequentially. This type of application is also identified as sequential collaboration. On the other hand, the most technically challenging application requires a responsive collaboration, which means that human and robot work at the same time in the shared workspace and they have to coordinate to accomplish the task safely [4]. The different levels of collaboration are summarized in Figure 1.3.

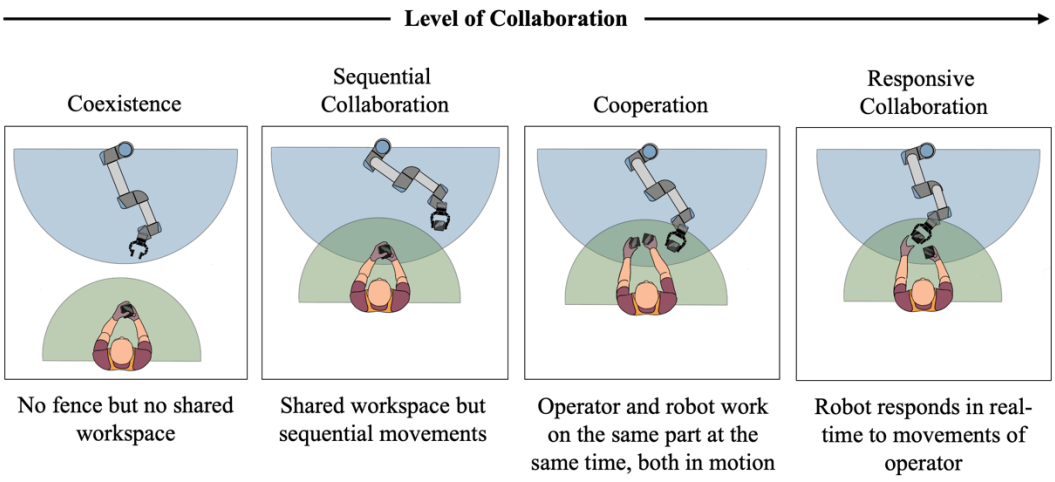


Figure 1.3 Level of collaboration identified by IFR [9]

Regardless of the level of collaboration, if the human and robot workspaces intersect, the industrial application must be designed according to the related

standards [10]. In particular, two different approaches can be implemented to guarantee safety: the *power and force limiting* (PFL) and the *speed and separation monitoring* (SSM). The PFL combines an accurate risk assessment with risk reduction measures so that the contact between the operator and the cobot does not result in harm to the operator. Therefore, in addition to passive safety design methods, one can actively control the energy of the moving parts of the robot by limiting forces, torques, velocities or momentum within a given range. In contrast, the SSM does not specify limitation to the robotic system as long as it maintains a protective separation distance from the operator. In practice, if the operator is within the robot workspace, the SSM translates into online robot controlling to generate alternative collision-free paths. In this case, the use of human tracking systems is needed for the minimum distance monitoring and collision avoidance algorithms shall be implemented to update in real-time the robot trajectory.

Concerning the human tracking, wearable systems and 3D vision systems are the best alternatives (see [11]). The first one would limit the human operator's movements with suits and sensors, while the use of cameras does not require wearing dedicated components. For this reason, this work considers a vision system based on RGB-D (where "D" stands for "depth") cameras as human tracking device. On the other hand, data obtained from cameras are sensitive to occlusions of the sensor. Thus, the problem of sensor fusion will be discussed to show how it is possible to use multiple depth sensors to optimize the output, the latter being a representation of the human in the three-dimensional space in terms of skeleton or point cloud.

The various aspects that have been discussed indicate that the highest level of collaboration, which has been identified as "responsive", is a synthesis of the whole set composed of cobot, operator, tracking system and control strategy for the safe accomplishment of a task. The thesis explores the responsive collaboration and presents hardware and software solutions to deal with the most demanding scenarios in the direction of the SSM approach. In particular, the discussion will focus on two main applications: the collision avoidance and the hand-over.

The importance of collision avoidance algorithms has been already pointed, while the hand-over represents the exchange of objects between human and robot (Figure 1.4). The dissertation treats collision avoidance and hand-over separately to simplify the discussion. They are apparently two opposite problems, but they can be combined opportunely to give more flexibility to the task. For example, in some cases it may be required to alternate a phase in which the robot and the

operator carry out two sub-tasks in parallel in the same workspace with a phase where they exchange objects. Moreover, even during the hand-over, the robot should approach the operator smoothly at the same time being careful to not collide with the environment.

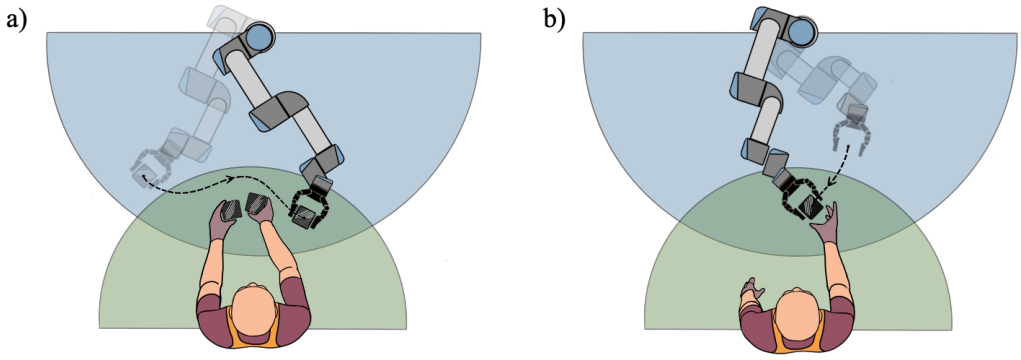


Figure 1.4 a) Collision avoidance. b) Hand-over

1.2 Contribution of this work

Concerning the novelties of this work, they will be contextualized to the state of the art and discussed in detail in the related chapters, but at this point it is worth to summarize the major advances with respect to the previous work by Leonardo Sabatino Scimmi [12].

Table 1.1 collects the key points of this document, divided by applications. The word “algorithm” refers to a series of instructions and calculations that properly combine some inputs, such as robot and human positions, to obtain an output, such as a robot command, that can quickly generate a controlled trajectory. This is specifically addressed in the title of this work as real-time trajectory planning.

At this point, a clarification is needed. A careful reader may observe that trajectory planning consists in defining the robot path and finding a time parametrization of the path under certain constraints. In offline applications, this can be dealt with classical techniques that allows to globally define the trajectory (see [13]). In contrast, most common real-time techniques define the trajectory locally, it means that only the motion within a certain range nearby the actual configuration is computed. To distinguish reactive local planners from global ones, some authors refer to “trajectory generation” instead of “trajectory planning” (e.g. [14]). The algorithms of this work use local methods based on *artificial potential fields* (APF) and the expression “real-time motion planning” is intended as a synonymous of “trajectory generation”. This is only a formal choice

which is consistent with the capability of APF to combine path planning and local trajectory planning in a feedback control law, all at once [15].

In addition to the theoretical novelties, the thesis distinguishes for the centrality of some human factors that can improve the experience of the operator for a more fluent collaboration. For example, the hand-over algorithm facilitates the object exchange by anticipating the human intentions and by orienting the end-effector according to the human arm. Moreover, a novel collision avoidance algorithm, which is able to control the alternative collision-free path of the robot, is presented in the central part of the thesis; the algorithm is named *multiple attractors potential* (MAP) and allows the robot to avoid the operator passing through selectable regions with the aim to make the robot path predictable.

Table 1.1 Contribution of this work

Application	**Algorithm**	**Description**	**Previous Work** [12]	**This work**
Collision Avoidance	*Robot links - human (Repulsive velocities)*	Evasive motion of the robot from the human	Simulation and experimental validation with human skeleton	Application to an assembly task and test with human point cloud
	Trajectory conditioning (MAP)	Controlled collision-free robot trajectory	Simulation and experimental validation using attractive and repulsive geometries	Simulation and experimental validation of a novel method based on multiple attractors potential (*MAP*)
Hand-over	*Hand position and orientation tracking*	Adaptive position and orientation of the TCP towards the human hand	Simulation and experimental validation	Experimental validation of an improved method with predicted hand position

It is important to specify that the algorithms presented in this work guarantee a safe interaction in the sense that they prevent unexpected contact of the robot with the operator. In particular, they can potentially satisfy the SSM approach, but they have not yet been tuned according to the standards. The reason is that the protective distance defined by the SSM is a function of the velocity of the operator, that is not directly measured by vision systems. When the operator's velocity is not available, [10] indicates the constant value of $1.6\,m/s$ to be

considered. However, this would have limited the testing phase because of a higher value of the protective distance. For this reason, a method to estimate the human velocity combining vision systems and/or wearable sensors is under study. This aspect will be addressed in future works.

The rest of the dissertation is mainly divided in 4 chapters. Chapter 2 is dedicated to human tracking by 3D vision system. Even if this chapter does not present theoretical novelties, the reason of spending a separate section relates to the importance of human tracking in responsive collaboration. An in-depth review of sensing technologies has been carried out to select the best tracking device. Chapter 2 also describes the multi-sensor layout and the fusion techniques that have been implemented to obtain improved human data, the latter being one of the inputs for collision avoidance and hand-over algorithms. In other words, this section introduces the scene in which the rest of the dissertation is set.

Chapter 3 is dedicated to the advances in collision avoidance techniques, by distinguishing two types of algorithms: the *Robot links - human* and the *Trajectory conditioning* (see Table 1.1). The *Robot links - human* algorithm will be also addressed as collision avoidance based on repulsive velocities. It is the same presented in [12], but it is reported to support some new applications in the results chapter. The second part of Chapter 2 is entirely dedicated to the collision-free trajectory conditioning problem and the novel method, which is identified as MAP.

Chapter 4 refers to the hand-over problem introduced in [12] and proposes a predictive scheme based on Kalman filter to anticipate the human movement.

Chapter 5 is dedicated to results. Some of them are presented by means of simulation tools, others are experimental tests. The results are applications of the algorithms presented in Chapter 3 and Chapter 4. In particular, the collision avoidance algorithm based on repulsive velocities and skeleton tracking presented in [12] is applied to a collaborative assembly task to discuss the advantages of collaborative robotics in a practical case. Moreover, the possibility to consider the human point cloud rather than the skeleton is investigated and tested. The MAP algorithm is verified through simulations and validated with experiments. The last section of Chapter 5 is dedicated to the outcomes of the hand-over algorithm.

The dissertation closes with conclusions and future works to summarize the strengths and the limitations of the work with a view on possible developments.

Chapter 2

Human tracking by 3D vision systems

The quality of the human tracking plays a central role in responsive collaboration since it may be decisive to ensure safety. In fact, wrong measurements of the human position may result in unexpected robot behaviors and serious hazard for the operator.

To limit the risk of injuries some works consider wearable devices, which consist of inertial sensors or active markers. However, contact methods create the disadvantage of intrusiveness or inconvenience for subjects, that may represent a limit for robotics or industrial applications.

In this work, a significant effort has been spent looking for the best alternative to wearables and the current technologies based on markerless 3D vision systems has revealed the most attractive solution. In particular, depth cameras, also known as RGB-D cameras, have been considered.

This chapter summarizes the steps that has led to the choice of the human tracking device and the sensor fusion algorithms that have been implemented to overcome the drawbacks typical of this kind of systems.

2.1 Depth cameras: a survey

The success of depth cameras, especially among researchers and developers, is due to relative low cost, easy programming and capability to give quite robust measurements without any wearable sensor. In particular, the last reason made this technology suitable for collaborative robotics, where depth cameras are used as human-robot interfaces so that the operator is not bound by any additional sensor.

The Microsoft Kinect v1, that was born as a gaming device in 2011, was the first cheap and reliable depth camera with built-in human tracking. This encouraged developers to create many different applications making their codes available for the community. After that, 3D sensors technology has significantly expanded and today the market is plenty of RGB-D cameras with different brands and specifications.

For the purpose of this work, the choice of the hardware and software technologies to drive the vision system of the collaborative robotic cell is a trade-off between accuracy, sensitivity to the ambient conditions, human tracking capability, hardware-software compatibility and price.

The raw measurement of a depth camera is a point cloud (Figure 2.1b), which is a 3D reconstruction of the observed environment (Figure 2.1a). The quality of the point cloud depends on the sensing technology of the camera, that can be distinguished in *active stereoscopy* (AS), *structured light* (SL) and *time-of-flight* (TOF) [16].

The next section briefly describes these technologies for a better evaluation of their applicability to collaborative robotics.

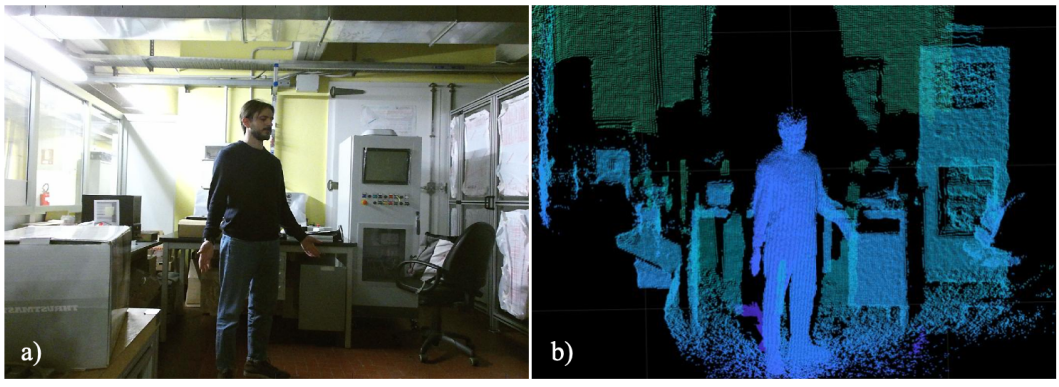


Figure 2.1 Color image (a) and point cloud (b) acquired by Kinect v2

2.1.1 3D sensing technologies

2.1.1.1 Active stereoscopy

Stereoscopy techniques use the same principle of the human eyes, reconstructing 3D information through images of the same scene from two different points of view. Active stereoscopy differs from the passive one due to the presence of a light projector, which covers the target with a random texture. This class of devices usually consists of two IR (Infra-Red) cameras and a single IR transmitter. IR is mounted because the active source is invisible to human eye and works in dark environment.

To merge the information from the two different points of view, feature-based or correlation-based matching are used; they identify similar features in the two images of the scene to overlap data and to generate a 3D map of the environment. The projected texture is an additional information which improves the matching in bad light condition or when few features are available. Figure 2.2 shows two pictures taken from the IR cameras of the Structure Core depth sensor [17]; the IR dotted pattern is clearly visible. In the yellow square it is also shown as the same feature does not locate in the same range of pixels for the two different images. This allows to extract the depth map on the right, where the colours refer to different depths, i.e. different distances of the objects from the sensor.

Since the depth is computed by triangulation, the algorithms are computationally expensive; in contrast, methods to solve the correspondence problem are very mature and a high framerate can be achieved [18]. Moreover, the depth resolution is quadratically dependent on the distance of the measured object, thus short-range applications are preferred [16]; long range and high resolution are also possible if longer baseline is chosen for the cameras, but it would lead to a bigger device [19]. Many studies revealed that the temperature of the IR projector generates texture variations, but in the active stereoscopy this is a minor issue since the pattern is not encoded [20]. The use of a random texture is also a major factor in multi-sensor setup and allows multiple units to easily coexist in close vicinity of each other [21]. Furthermore, the presence of two cameras results in a field of view limited by the optical components but makes this technology more suitable for outdoor applications [20].

Figure 2.2 Reconstruction of depth images from IR in active stereoscopy-based cameras

2.1.1.2 Structured light

Sensors based on structured light use a single camera with a structured-codified pattern projected on the target. The known pattern ensures unique correspondences to triangulate with the camera and no more sensors are needed. Figure 2.3 shows the depth processing of Kinect v1, which uses an IR texture projector and a single IR camera.

The depth information is computed from triangulation and epipolar geometry, thus the devices based on structured light technology suffer from the related problems mentioned for active stereoscopy. On the other hand, this type of system is more compact and the parts used for manufacturing are standard electrical and optical components, such as CCD imaging sensors and diffractive optical elements [20]. Other error sources for structured light are reflective materials and outdoor ambient light, which lead to non-valid pixels [16], [22]. The multi-sensor interferences have also been investigated in [23], showing some blind pixels in the overlapping area of the pattern projected by different laser; this happens because the IR projectors disturb each other, compromising the structure of the pattern used for triangulation. Furthermore, some SL sensors project sequentially different structured patterns to reconstruct the scene; this affects the computation time and may lead to wrong measurements of moving objects [20]. Finally, a good resolution and a limited field of view are characteristics of structured light-based devices.

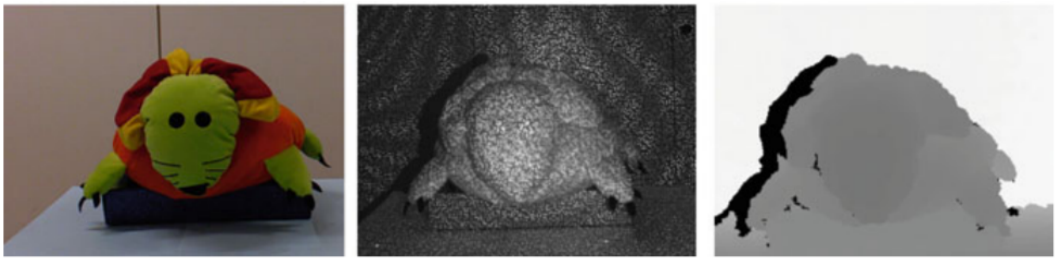


Figure 2.3 Images acquired from Kinect v1 sensor, using structured-light technology [20]

2.1.1.3 Time-of-flight

A time-of-flight system consists of a light transmitter and a receiver. The transmitter radiates a modulated signal which bounces the closest objects; the wave is catched from the receiver (Figure 2.4). Thus, the phase difference between the emitted signal and the received one is translated in a distance information. Depth cameras which exploit the TOF principle measure the depth for each pixel of the sensor matrix. They consist of an illuminating optic and a lens, which are mounted on the camera with a certain displacement. By means of the pin-hole camera model [24], if intrinsic and extrinsic parameters of the camera are known, is then possible to convert each pixel measurement into depth data. The most famous examples are the Microsoft Kinect v2 and its successor Kinect Azure [25], which were launched in 2013 and 2020 respectively. These devices, as most of the 3D cameras which works using TOF, have an IR projector.

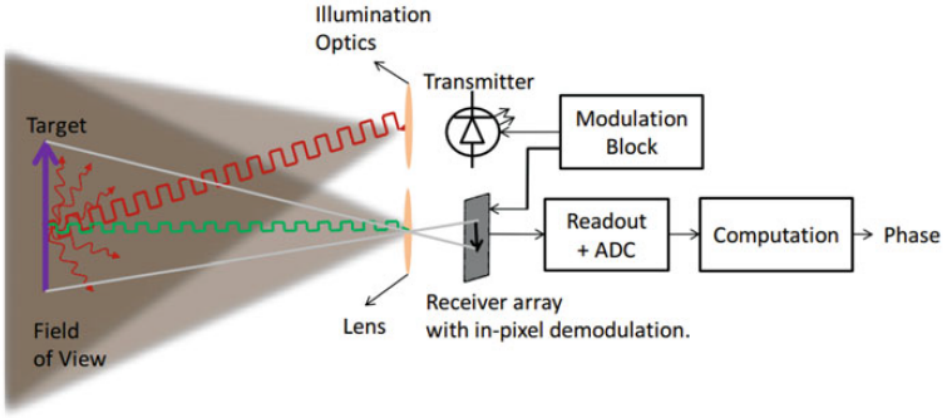


Figure 2.4 TOF working principle [16]

In general, TOF sensors are compact because the illumination is placed just next to the lens, whereas the other systems need a certain minimum baseline. Moreover, the extraction algorithm from signal phase to 3D map is straightforward. On the other hand, the IR signal is sensitive to outdoor light exposure, to temperature and to reflectivity of the target [16], [20], [26]. For instance, a comparison between Kinect v1 and Kinect v2 technologies is reported in [22]. The results show that the structured light sensor, when valid pixels are delivered, performs better in many critical conditions, e.g. outdoors or with reflective surfaces, but in all tests a significant quantity of blind pixels were registered. For the TOF sensor of Kinect v2, depth measurements were always available (few non-valid pixels) but less accurate. The same behavior was observed for multi-sensor interference. Moreover, Kinect v2 was more reliable

when tracking objects in motion. This is probably due to hardware-software optimization, since usually TOF systems require a certain time to integrate the signal captured by the matrix sensor and to increase the signal-to-noise ratio (SNR): in this time interval, the scene is supposed to be fixed. In many devices, to improve the SNR, multiple sensor pixels are utilized to calculate a single depth pixel value; this decreases the effective image size, leading to a lower resolution. Common error sources are also the "multipath-effect", due to reflective, diffracted or scattered copies of the transmitted signal (e.g. when illuminating reflective, transparent objects or concave corners) and the "flying pixels", which exhibit in regions of discontinuity in depth or reflection [20]. An advantage of TOF systems is that the field of view depends on modulation frequency and it is not restricted by special optical components [27].

2.1.1.4 Comparison

Three main classes of devices have been identified: structured light, active stereoscopy and time-of-flight. SL cameras have good resolution and accuracy; they perform better indoors than other technologies. On the other hand, they fail with direct sun light exposure. AS-based devices combine the advantages of SL and stereoscopy to reach high resolution and the best frame rate on the market. Moreover, because of their capability to work in any ambient light condition, they are the most flexible existing camera for robotics application. TOF cameras mount the most advanced technology, since their measurements come from the analysis of a physical quantity which directly carries the depth information, in contrast with correlation techniques. The results of the works cited in the previous sections show that TOF represents a compromise between the accuracy of SL and the flexibility of AS; thus, they still represent a reliable solution for indoor usage. Moreover, research is moving towards new pixel sensors able to increase the signal-to-noise ratio outdoors, so a significant improvement of TOF technology in the next years is predictable [28].

Table 2.1 contains a summary of the main aspects of the 3D sensing technologies. It is based on the works cited in the previous sections. Notice that marks are referred to the technology itself and not to particular devices since the hardware-software integration is crucial for these systems and may significantly improve performances of the depth camera.

To select the most suitable 3D sensor, the first step consisted of comparing the existing devices representing the different technologies on the market. Table 2.2, Table 2.3 and

Table 2.4 summarize the most recent depth sensors available for the mass market. The devices are grouped by sensing technology (AS, SL and TOF).

Even if Microsoft has announced the discontinuity of both Kinect v1 and v2 sensors, they are still considered because they represent great values in their category. In the last 4 years there has been a large growth of 3D commercial cameras, especially for AS devices. This is due to the high resolution and high frame rate which can be delivered with relative low cost, as well as to the possibility to work outdoors, that is a great feature for mobile robotics, e.g. drones and UGV. Today, Intel RealSense depth cameras represent the most mature AS sensor, offering a reliable hardware and a wide range of cross-platform software applications.

Some examples of 3D Vision systems for industrial applications are also reported in Table 2.5 and Table 2.6. However, in this phase they have been excluded due to the low value for the money and above all the incompatibility with the best affordable third-party skeleton tracking software [29]. On the other hand, some of the mass market sensors have built-in skeleton tracking, others are compatible with the third-party middleware [29].

At the end, the best cameras are the Microsoft Kinect v2 (TOF), the Intel Realsense (AS) and the Orbbec Astra Pro (SL). The Kinect Azure, although the improved specifications with respect to the Kinect v2, was still not available on the market at the beginning of this research. However, its integration is being considered for future works.

Table 2.1 3D sensing technologies comparison

Feature	**Active Stereo**	**Structured Light**	**Time-of-Flight**
Material Cost	medium	low	high
Compactness	low	medium	high
Software Complexity	high	medium	low
Response Time	medium	high	low
Pixels Resolution	high	medium	low
Depth Resolution	low	medium	high
Low-light Performance	medium	medium	high
Outdoors Performance	high	low	medium
Object Material Influence	medium	high	high
Temperature influence	medium	medium	high
Power Consumption	medium	medium	high

The second step for the selection of the 3D sensor is described in the next section and consisted in evaluating human tracking capabilities of the best in class of each category for a final verdict.

Table 2.2 Time-of-Flight (TOF) Cameras

	 **Microsoft Kinect v2**	 **Microsoft Kinect Azure**	 **PMD Monstar**	 **Lips Edge DL**
Technology	TOF	TOF	TOF	TOF
Range (m)	0.5 - 4.5	0.5 - 3.86	0.5 - 6	1 - 4
Resolution	512 x 424	640 x 576	352 x 287	320 x 240
Frame Rate (fps)	30	30	60	30
FOV (H x V)	70° x 60°	75° x 65°	100° x 85°	74° x 58°
Dimensions	299 x 66 x 67	103 x 39 x 126	62 x 66 x 29	113 x 37 x 5627
Software	Kinect SDK 2.0 (C,C++,C#), libfreenect2	Azure Kinect Sensor SDK, Azure Kinect Body Tracking SDK (C, C++)	Royale SDK (C/C++ based, Matlab, DotNet, CAPI, OpenCV, OpenNI2, ROS)	Lips SDK, OpenNI
OS	Windows8,10 (Debian, Ubuntu 14.04 or newer, Mac OS X with libreenect2)	Windows 10 (x64), Linux Ubuntu 18.04 (x64) with OpenGLv4.4 GPU driver	Windows 7/8/10, Linux/ARM, Ubuntu Linux 16.04 + Qt5.5, macOS	Windows,7/8/10, Linux, Android, MacOS

Table 2.3 Active Stereoscopy (AS) and Passive Stereoscopy (PS) Cameras

	 **Intel RealSense D435**	 **Structure Core**	 **Mynt Eye D**	 **ZED Stereo Camera**
Technology	AS	AS	AS	PS
Range (m)	0.1 - 10	0.3 - 5	0.275 - 10	0.5 - 20
Resolution	Up to 1280 x 720	1280 x 800	1280 x 720	2560 x 720
Frame Rate (fps)	Up to 90	60	60	60
FOV (H x V)	85° x 58°	59° x 46°	120° (D)	90° x 60°
Dimensions	90 x 25 x 25	105 x 15 x 21	145 x 20 x 28.6	175 x 30 x 33
Software	Intel RealSense SDK (Supports python, node, unity, ROS, C++)	Structure SDK, OpenNI2 (C++, C#)	Mynt Eye SDK (C, C++)	ZED SDK (API in C, C++, ROS, Matlab, OpenCV, Python, Unity)
OS	Windows 10, Linux, Mac OS	Windows, Linux, Mac OS, Android	Windows 10, Linux (Ubuntu 14.04/16.04), Mac OS, Tegra	Windows 7/8/10, Linux

Table 2.4 Structured Light (SL) Cameras

	 **Microsoft Kinect v1**	 **Orbbec Astra Pro**	 **Asus Xtion 2**
Technology	SL	SL	SL
Range (m)	0.4 - 4.5	0.6 - 8	0.8 - 3.5
Resolution	320 x 240	640 x 480	640 x 480
Frame Rate (fps)	30	30	30
FOV (H x V)	57° x 43°	60° x 49.5°	74° x 52°
Dimensions	280 x 63 x 38	165 x 30 x 40	110 x 35 x 35
Software	Kinect SDK 1.8, OpenNI 2.2 (C++, C#)	Orbbec Astra SDK, OpenNI	OpenNI SDK bundled (C++, C#)
OS	Windows, Linux, MacOS X	Android, Linux, Windows 7/8/10	Windows 8/10, Linux Ubuntu 14.04/16.04/17.04

Table 2.5 Time-of-Flight (TOF) Cameras for Industry

	 **Sick Visionary-T**	 **Ifm 3D Camera**	 **Basler 3D Camera**
Technology	TOF	TOF	TOF
Range (m)	0.5 - 60	0.3 - 8	0 - 13
Resolution	176 x 144	352 x 254	640 x 480
Frame Rate (fps)	30	25	20
FOV (H x V)	69° x 56°	45° x 60°	57° x 43°
Dimensions	162 x 116 x 104	120 x 95 x 76	142 x 62 x 69
Software	SOPAS, API (Java, Matlab), Webserver, Telegram listing (universal use, e.g. Python, C++, C#), Webserver, ROS	Ifm Vision Assistant, XML-RPC, ROS	ToF Software (C++, C#)
OS	Windows, Linux	Windows 7/8/10, Ubuntu 14.04/16.04	Windows, Linux

Table 2.6 Passive Stereoscopy (PS) and Multi-Structured Light (MSL) Cameras for Industry

	 **Carniege Multisense**	 **e-con Tara Stereo**	 **Nerian Scene Scan**	 **Ensenso N35**
Technology	PS	PS	PS	MSL
Range (m)	0.4 - 20	0.5 - 3	scalable	0 - 3
Resolution	960 x 480	752 x 480	800 x 592	1280 x 1024
Frame Rate (fps)	30	60	65	10
FOV (H x V)	80° x 45°	n/a	scalable	52° x 58°
Dimensions	130 x 130 x 65	100 x 30 x 35	144 x 41 x 35	175 x 50 x 52
Software	C++ Library, ROS NOde	e-CAMView, TARA SDK (OpenCV), ROS	API Library (C++, supports HALCON, Matlab,, ROS)	MVTec HALCON, API Library (C++, C# / .NET)
OS	Windows, Linux	Windows 7/8/10, Linux, MacOS, Android	Windows, Linux	Windows, Linux

2.1.2 Human tracking software

To estimate the human pose from raw data, depth sensors are paired with tracking algorithms that take the point cloud as input and perform body tracking in two stages. The first stage is the segmentation of the human shape from the whole point cloud; the second stage use registration methods to obtain human joint positions related to the different body part (Figure 2.5), commonly known as skeleton [30]. In practice, the motion tracking algorithm is implemented in software packages, that can be third-party or included in the SDK (from software development kit) of the depth sensor.

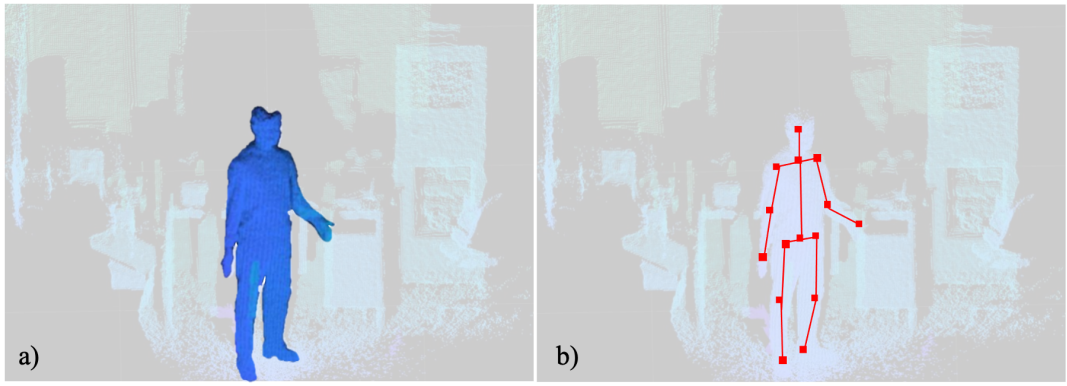


Figure 2.5 Point cloud segmentation (a) and skeleton tracking (b) by Kinect v2

What made Kinect cameras so attractive for research and robotics purposes, in addition to the depth sensing feature, is the robust proprietary motion tracking software, which delivers robust human joint position at 30 fps [31]. The discontinuity announced by Microsoft between the v2 and the Azure versions boosted the 3D Camera market for competitors and created opportunity for sensorless skeleton tracking software development. In particular, PMD, Creative, Mynt, Structure, ZED and Asus do not explicitly offer skeleton tracking but keep a cross-platform SDK which can be integrated with many of dedicated libraries. For instance, free software like OpenNi NiTE [32], Skeltrack [33], OpenPose [34] and OpenPTrack [35] are available for download on GitHub and enable to use point clouds to extract the human pose information. On the other hand, Intel RealSense SDK and Orbbec Body Tracking SDK already offer body tracking, even if a valuable option is the Nuitrack SDK [29], which is a 3D skeleton tracking middleware very similar to Kinect SDK 2.0.

Among the selected cameras, the Intel Realsense and the Orbbec Astra Pro can run the Nuitrack SDK, thus the skeleton tracking capabilities are similar to Kinect v2 sensor and Kinect SDK 2.0. However, the Kinect (hereafter the "v2" is not specified anymore) had some important advantages that made it the first choice for the development phase of the collaborative robotic cell: hardware-software integration, easy programming and vast literature. In fact, the sensor already comes with the skeleton tracking software bundle and it is officially supported by Matlab. In addition, the scientific literature is plenty of studies on the actual abilities of the sensor for different applications, as well as on possible solutions to improve human tracking.

A well-known problem is the wrong human pose estimation due to single view. In fact, the position and the orientation of the human with respect to the camera affects the skeleton tracking [30]. This aspect is even more complex in collaborative robotics application, where the human moves not only relatively to the camera but also around the robot, the latter representing a potential occlusion for the sensor.

The next section is dedicated to the human tracking algorithms that have been implemented to overcome the limitations of using a single Kinect.

2.2 Sensor fusion

An intuitive way to mitigate human tracking errors related to single view is to use multiple sensors. The idea is to place more Kinect observing the scene from

different viewing angles so that merging the information of the cameras results in a better human pose estimation. However, the use of multiple sensors is a necessary but not sufficient condition, since it translates into data fusion problem.

A major aspect concerns the type of data to match. Kinect SDK human tracking can deliver the human shape in terms of point cloud or a more precise representation as skeleton (Figure 2.5). In some applications, where distinguishing between the body parts is not relevant, the point cloud can be sufficient to describe the human. This has the advantage of using the real measurement of the sensor. On the other hand, if the application requires to track the position of the human joints, the skeleton is the best option. However, in this case human data are an estimation made from the measurement, which means that the risk of tracking errors is higher.

Depending on the type of data for input and output, three fusion schemes are possible. The first consist in merging the point clouds into one to obtain a better reconstruction of the human shape [36]. This is the most direct approach but may require large processing times depending on the resolution of the point clouds, which can be made of thousands of points. Because point clouds usually require to be denoised, this process can be time consuming and not suitable for real-time. Calculations can be parallelized using more computers but optimizing the algorithm for fast data transfer can be challenging. The second option is to merge the point clouds and to use registration methods on the resulting point cloud to extract the skeleton [37]. This is similar to the previous case but requires also advanced programming to implement customized skeleton tracking algorithms. The third is a more practical scheme that combines the skeleton to better estimate human joint positions. The techniques that are based on this approach are fast since they work on small amount of data as the Kinect skeleton is made of 25 joints. The output in terms of skeleton usually is not good as the second class of fusion schemes, but the algorithm is easier to implement.

For the purpose of this work, two different algorithms based on the first and the third fusion scheme are described. In particular, the following discussion is limited to a layout with two Kinect, but the algorithms here proposed can be extended to any number of devices, with some limitations due to the hardware and the processing time. More details will be given in the related sections.

Before describing the fusion algorithms, the method for the spatial matching of the depth sensors is reported.

2.2.1 Multiple Kinect setup

In Figure 2.6, the laboratory setup of the vision system based on multiple Kinect is shown. For the preliminary study the robot has not been considered to simplify the hardware architecture and the software implementation. Two Kinect are displaced observing the workbench with their z_K-axes forming an angle of about 90° and intersecting in the area where the human moves. This choice is related to the working principles of the fusion algorithms that have been tested [36], [38].

A schematic representation of the layout is depicted in Figure 2.7. Two computers C1 and C2 acquire the signal from each Kinect separately, as required by Kinect SDK [31]. The computers are connected through an ethernet cable and can exchange data in a master-slave configuration. The calculations are made in Matlab. In Figure 2.7, the world frame $\mathcal{F}_W$ and the ROI (from *region of interest*) are also shown. The world frame represents the coordinate system where human tracking data is processed once the spatial matching of the cameras has been done. For instance, the objects on the table in Figure 2.6, indicated with the letters O, a and b, are the tools for spatial matching. Further details are given in the next section.

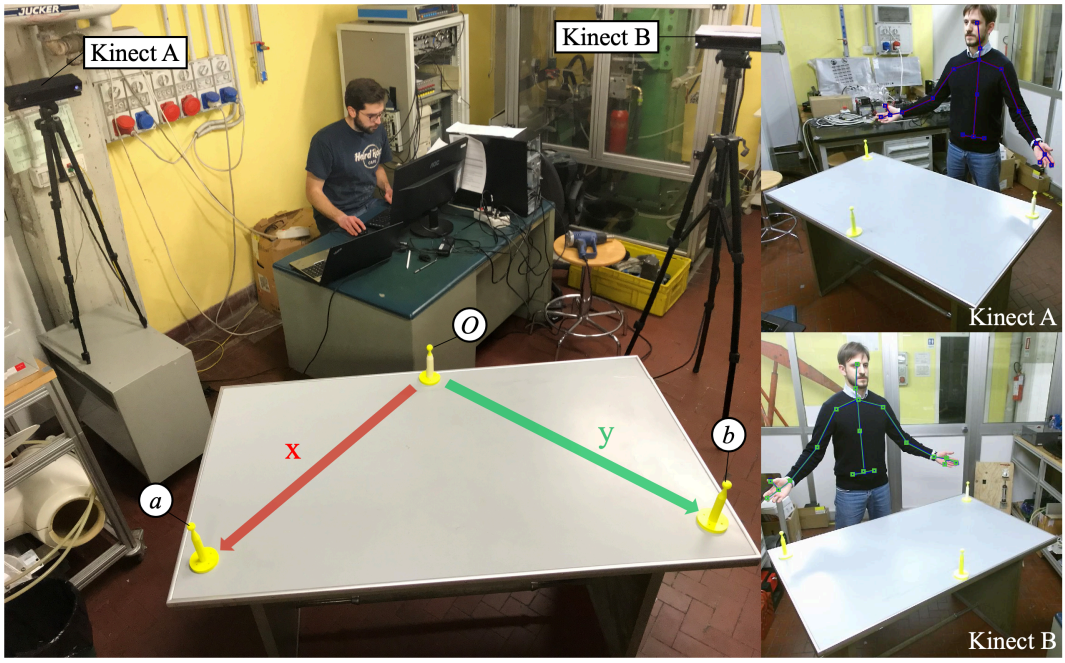


Figure 2.6 Multiple Kinect laboratory set-up

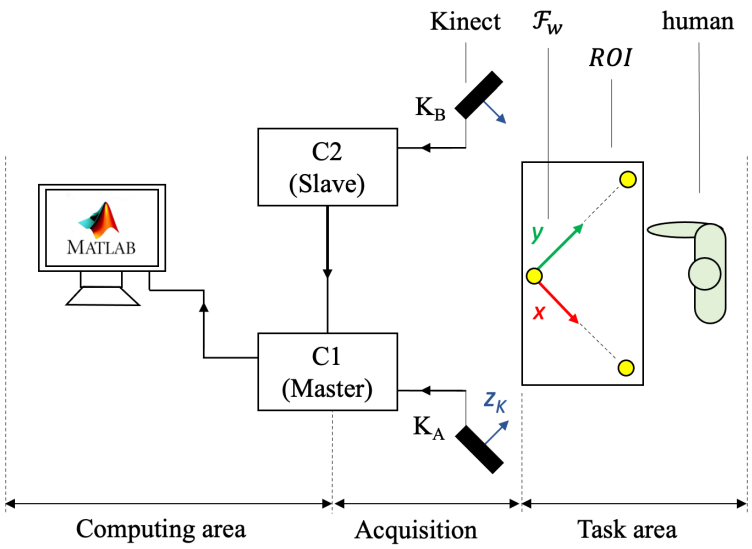


Figure 2.7 Vision system hardware architecture

2.2.2 Spatial matching

The Kinect is equipped by two sensors: the color sensor and the depth sensor. The color sensor gives RGB images, while the depth sensor based on TOF technology provides the depth information of the closest objects in the scene. The two sensors are mounted next to each other and there is a small gap between them, so that the color and the depth information are measured in two different pixel planes. The point cloud and the skeleton joint coordinates refers to the depth sensor. In particular, they are given by Microsoft Kinect SDK in the depth frame $O_K - x_K y_K z_K$ whose origin is located at the center of the IR sensor and the axis are oriented as in Figure 2.8.

In a multiple camera layout, the spatial matching is the transformation of the 3D information of each Kinect from the depth frame to a common reference frame $O - xyz$, also indicated as the world frame, in order to combine data consistently. This can be done by calculating the transformation matrix which allows to transform the points from the Kinect depth frame to the world frame. In order to identify the world frame with the depth sensor, a method based on solid markers has been developed [39].

Three spherical markers are placed on the table, at the top of a conical base (Figure 2.6). The markers are designed so that their geometry can be distinguished in the point cloud of the Kinect (Figure 2.9), up to $2.5\ m$ away from the IR sensor. The coordinates of the points $\boldsymbol{o}^K$, $\boldsymbol{a}^K$ and $\boldsymbol{b}^K$ (the superscript K indicates

the point observed in the Kinect depth frame) can be identified from the point cloud with an average depth accuracy error $< 2\ mm$ [40].

The world frame is built considering the origin in O, the x-axis aligned with a, the z-axis orthogonal to the plane containing the three markers and the y-axis completing the right-handed coordinate system (Figure 2.8). Once the origin $\boldsymbol{o}^K$ and the axis of the depth frame are known, it is straightforward to calculate $\boldsymbol{\mathcal{A}}_K$ using homogeneous transformation [13].

This method based on solid markers will be practical also for the spatial matching of the robot. In fact, if the markers are placed within the robot workspace, one can just place the TCP (from *tool center point*) in correspondence of the markers to acquire the coordinates of the points $\boldsymbol{o}^r$,$\boldsymbol{a}^r$,$\boldsymbol{b}^r$ (the subscript r identifies the points observed in the robot frame) and calculate $\boldsymbol{\mathcal{A}}_r$. Further details on the experimental set-up with the collaborative robot will be given in the results section.

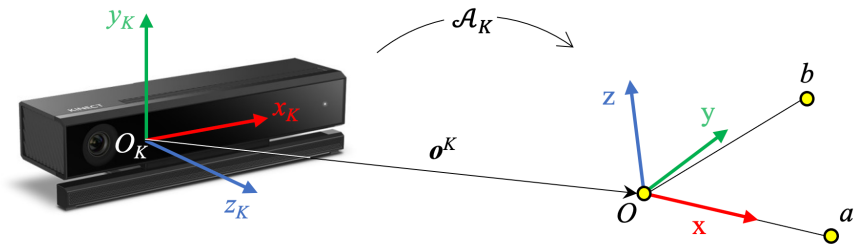


Figure 2.8 Transformation from Kinect frame to world frame

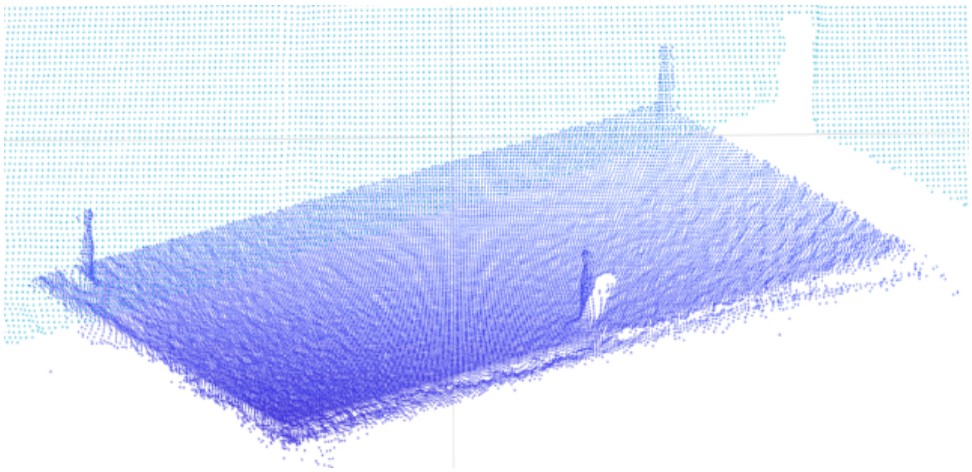


Figure 2.9 Point cloud of the workbench with the markers for spatial matching

2.2.3 Skeleton Fusion

The skeleton tracking with a single Kinect is not reliable for collaborative robotics due to several issues that may led to wrong human joint positions estimation [38]. For example, if objects are placed between the sensor and the operator, Kinect SDK tries to infer the position of the body parts that are not visible. This problem is known as sensor occlusion. Furthermore, even in the absence of objects, depending on the human body orientation with respect to the sensor viewing plane, some body parts may hide from each other, resulting in self-occlusion. Another problem is represented by bone-lengths variation, where for bone it is intended the distance between two consecutive joints. In fact, it has been observed that they vary during the acquisition of human motion, which is not desirable. Other minor errors are identified as vibrations and refer to flickering joints even if the human is not moving.

These problems can convincingly be addressed with the multiple Kinect algorithm proposed in [38]. Consider the layout in Figure 2.6. In the same figure on the right, the skeletons S_A and S_B of the two Kinect labeled with A and B are also shown. Assume that the cameras are spatially matched as described in the previous section. Let $\boldsymbol{p}_i^A \in S_A$ and $\boldsymbol{p}_i^B \in S_B$ denote the joints of each skeleton in the world frame, with $i = 1,2,\dots,n$ where $n = 25$ is the number of joints composing the single skeleton in full body modality. The algorithm is based on constrained minimization and consists of two steps: initialization and optimization.

During initialization, the operator stands in front of the cameras so that his body can be easily recognized by each Kinect (Figure 2.6). Thus, the mean $\overline{\boldsymbol{p}}_i^A$ and $\overline{\boldsymbol{p}}_i^B$ over 30 frames are considered to calculate the bone lengths $l_{i,j}$ as:

$$l_{i,j} = \frac{1}{2}\left\|(\overline{\boldsymbol{p}}_i^A + \overline{\boldsymbol{p}}_i^B) - (\overline{\boldsymbol{p}}_j^A + \overline{\boldsymbol{p}}_j^B)\right\| \tag{2.1}$$

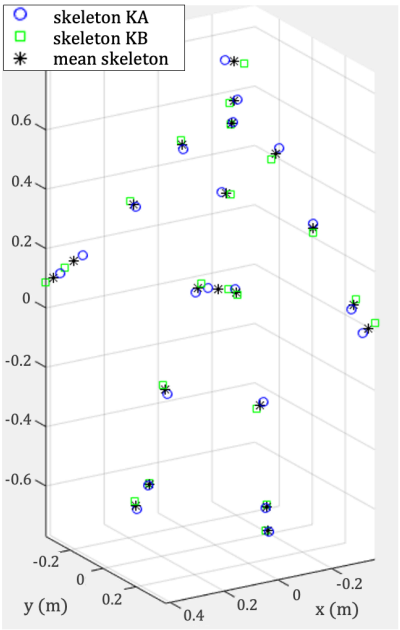
Figure 2.10 shows an example of initialization. As the bone lengths vary depending on the subject, this phase is performed once and the subject data is stored in a database.

The optimization phase takes into account the joint tracking confidence of the Kinect SDK (joint *tracked* or *inferred,* see [41]) and the conservation of the bone lengths. It is performed for each frame by solving the following problem for the optimal joint values $\boldsymbol{p}_i^*$:

$$\begin{aligned} \min \quad & \sum_{i \in S_A} w_i^A \left\| \boldsymbol{p}_i^* - \boldsymbol{p}_i^A \right\|^2 + w_i^B \left\| \boldsymbol{p}_i^* - \boldsymbol{p}_i^B \right\|^2 \\ \text{s. t.} \quad & \sum_{\{i,j\} \in S_A} \left(\left\| \boldsymbol{p}_i^* - \boldsymbol{p}_j^* \right\| - l_{i,j} \right)^2 = 0 \end{aligned} \tag{2.2}$$

where w_i^A and w_i^B are weights calculated according to [38] that depend on the tracking confidence of Kinect SDK. For instance, if w_i^A is greater than w_i^B, the optimal $\boldsymbol{p}_i^*$ is closer to the joint $\boldsymbol{p}_i^A$ observed by Kinect A, and vice versa. The problem (2.2) can be solved with the efficient numerical scheme reported in [38]. The code has been implemented in Matlab assigning the optimization phase to the Master C1, equipped with an intel i7-6700 processor, that can execute the algorithm with a processing time $< 4.5\ ms$.

Figure 2.11 shows the effect of the optimization. In this example, from the perspective of Kinect A the right arm is occluded and the Kinect SDK tracking algorithm fails the pose estimation of the hidden part. However, the optimization algorithm recognizes that the confidence of the joints of the right arm is higher in Kinect B and corrects the estimation.



bone	l (m)
spine_low	0.33378
spine_up	0.24442
neck	0.079718
head	0.14266
shoulder_left	0.16967
arm_left	0.28003
forearm_left	0.27407
hand_left	0.088435
shoulder_right	0.17318
arm_right	0.2887
forearm_right	0.26812
hand_right	0.084394
hip_left	0.066879
leg_upleft	0.41834
leg_lowleft	0.39019
foot_left	0.086926
hip_right	0.066396
leg_upright	0.40546
leg_lowright	0.38337
foot_right	0.089071

>>

Figure 2.10 Initialization of the skeleton fusion algorithm: bone lengths calculation

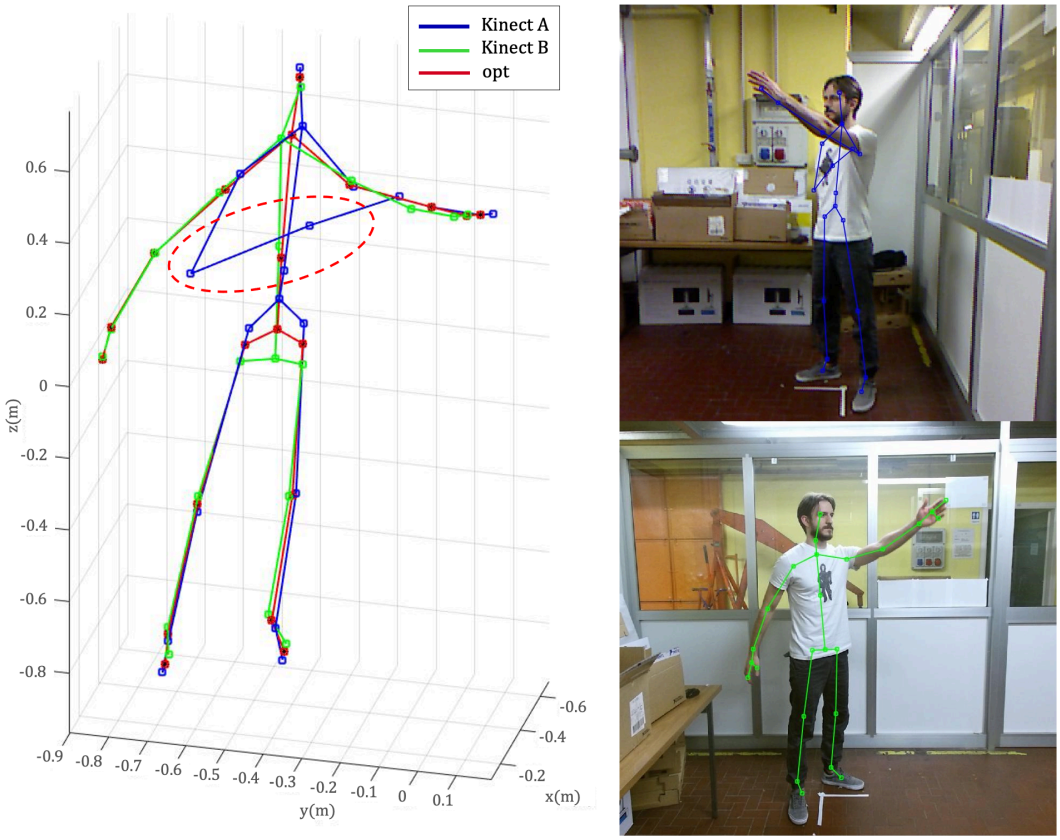


Figure 2.11 Skeleton fusion algorithm

2.2.4 Point cloud fusion

The skeleton fusion algorithm shows concrete improvements compared to the single camera. However, it stands on algorithms to interpret the depth information and thus suffer the limitations of registration methods [42]. A solution consists in directly using the point cloud to identify the human body volume: the errors can only lead to a loss of information, e.g. due to occlusions or reflective materials [40], not to a wrong human pose estimation. This is crucial because it is simpler to handle the lack of data than to fix the misleading one. For this reason, a method based on human recognition from point cloud that allows to merge the point clouds acquired by different Kinect has been developed [36].

The point cloud is the 3D representation of the environment that is directly computed from measurements. More precisely, the raw data of a depth sensor is a depth image, that gives distance information in a 2.5D format. This means that for each pixel of the image plane identified by the coordinates u_x and u_y in the pixel

frame, the component p_z^K of the observed point in Kinect frame $\boldsymbol{p}^K = [p_x^K \ p_y^K \ p_z^K]^{\mathrm{T}}$ is measured (Figure 2.12).

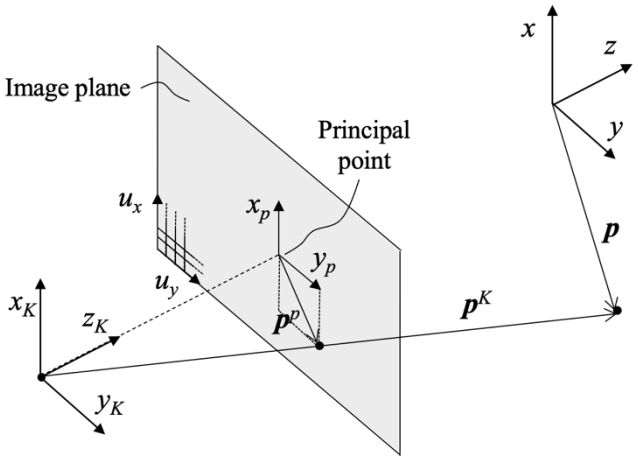


Figure 2.12 Pinhole camera model

To transform from pixel frame to world frame, a pinhole camera model with radial distortion is used [24], giving the following relationships:

$$\boldsymbol{\mathcal{A}}_K = [\boldsymbol{\mathcal{R}}_K | \boldsymbol{o}_K] \qquad \boldsymbol{\mathcal{K}} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$\boldsymbol{p}^K = \boldsymbol{\mathcal{A}}_K^{-1} \boldsymbol{p} \qquad \hat{\boldsymbol{p}}^K = \boldsymbol{p}^K / p_z^K \tag{2.4}$$

$$\begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \boldsymbol{\mathcal{K}} \hat{\boldsymbol{p}}^K \tag{2.5}$$

$$x_d = x_p(1 + k_1 r_p^2 + k_2 r_p^4 + k_3 r_p^6) \tag{2.6}$$

$$y_d = y_p(1 + k_1 r_p^2 + k_2 r_p^4 + k_3 r_p^6) \tag{2.7}$$

$$r_p = \sqrt{x_p^2 + y_p^2} \tag{2.8}$$

where $\boldsymbol{\mathcal{R}}_K$ is the rotation matrix from Kinect frame to world frame, $\boldsymbol{o}_K$ is the translation vector, $f_x f_y$ are the focal lengths, $c_x c_y$ are the coordinates of the principal point in the image plane and $k_1 k_2 k_3$ are the coefficients of radial distortion. $\boldsymbol{\mathcal{K}}$ defines the intrinsic parameters of the camera. $x_p \ y_p$ are the distortion-free coordinates on image plane, $x_d \ y_d$ are the corresponding real

coordinates. If $\mathcal{K}$ is unknown, a calibration procedure is required for each camera. In this case $\mathcal{K}$ is obtained directly from the camera using the C++ wrapper function in [43], passing through Microsoft SDK. Table 2.7 summarizes the intrinsic parameters of the two Kinect used in the laboratory setup of Figure 2.6. Using equations (2.3)-(2.8) and values in Table 2.7 it is possible to convert points from depth space to cartesian space, obtaining the 3D point cloud. In practice, this step is implemented in a custom Matlab function.

Table 2.7 Kinect intrinsic parameters

	Kinect A	**Kinect B**
f_x	366.0512	367.4633
f_y	366.0512	367.4633
c_x	260.5019	256.212
c_y	205.2539	208.8752
k_1	205.2539	208.8752
k_2	-0.27082	-0.2719
k_3	0.098178	0.10081

Once the point clouds in the world frame are available on both C1 and C2, the human volume is extracted by means of segmentation. Thus, the signal is downsampled and denoised using the Matlab Computer Vision System Toolbox. This step is required to speed up calculation and to eliminate outliers. Successively, the tuned point clouds are collected and merged in C1 to obtain the human shape. Figure 2.13 illustrates the steps for point cloud processing. In this example, a grid step value of 0.01 is chosen for the *downsample* Matlab function and the merged human point cloud consists of 11372 points in the cartesian space. Notice how the use of two sensors fills the self-occlusion of the human.

Human point clouds corresponding to different downsampling grid steps are shown in Figure 2.14 (decreasing the grid step results in higher resolution of the point cloud). Moreover, the possibility to use a convex hull instead of the point cloud is evaluated. The convex hull is built from the grid 0.02 point cloud. It was observed that using different grid steps does not produces significant differences in triangulation, since the mesh is convex. The effect of the convex hull will be analyzed in the results chapter. At this point, it is worth mentioning that the

convex hull can be intended as a more conservative approach, since the human size is augmented.

In Table 2.8, the influence of the point cloud density on the processing time is reported. Time data are obtained considering 300 test samples for different downsampling grid steps. Because some phases are parallelized (see Figure 2.13) it is specified that C2 is equipped with an intel i7-7500U processor. The standard deviation for each measurement was close to zero, so it is omitted. In all cases the entire algorithm performs at higher frequency than Kinect sensor refresh rate. This is a significant result since Matlab is not optimized for real time application with large amount of data.

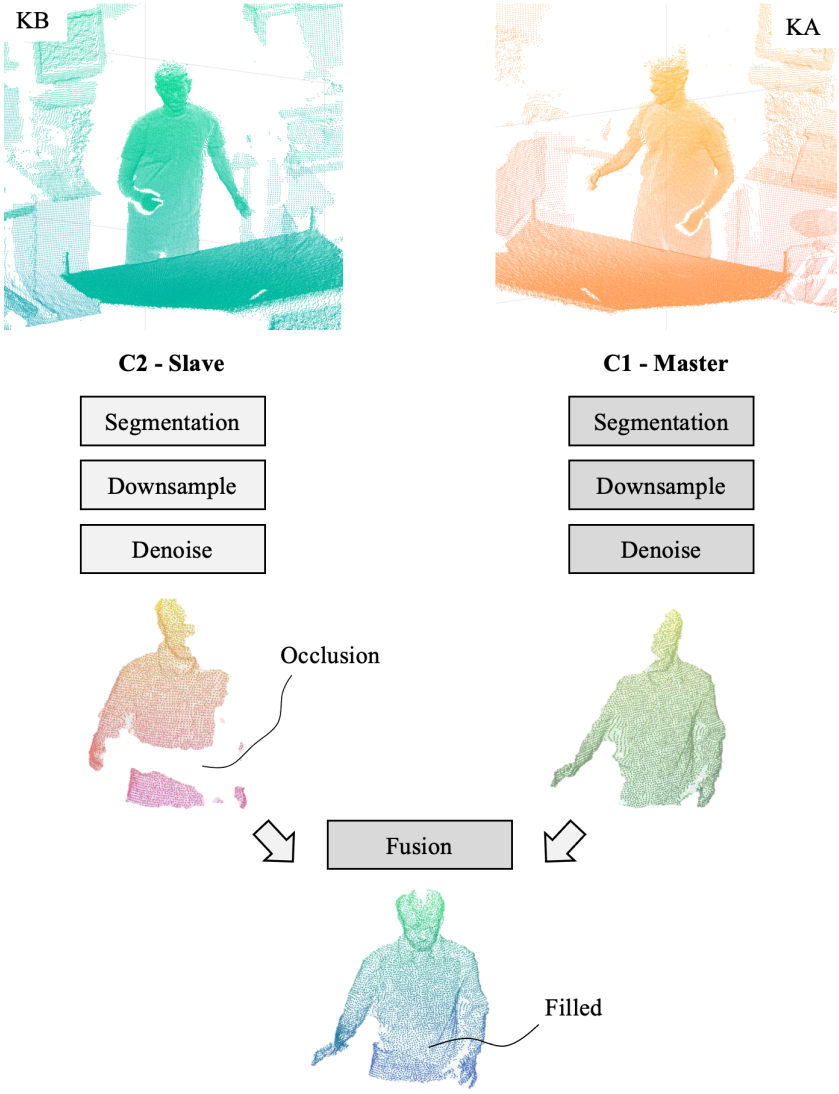


Figure 2.13 Point cloud fusion algorithm

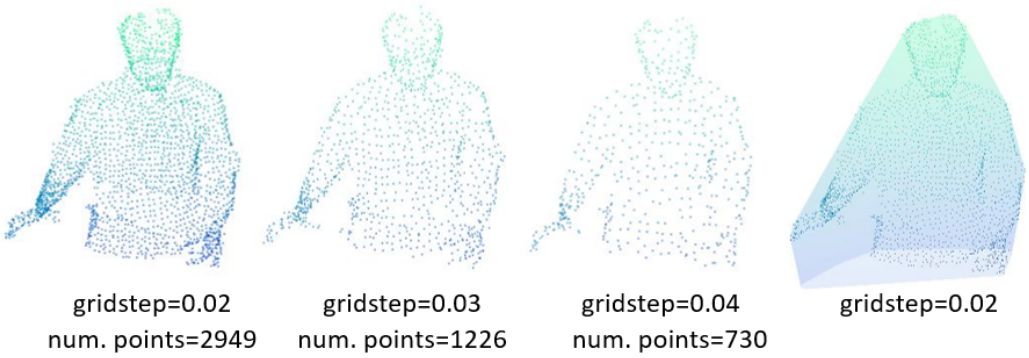


Figure 2.14 Point clouds and convex mesh with different downsampling grid steps

Table 2.8 Calculation times (in ms)

	Point Cloud		**Convex Hull**	
	grid 0.02	grid 0.03	grid 0.03	grid 0.04
Point cloud extract.	2.3	2.3	2.3	2.3
Downsample	4.1	4.1	4.1	3.8
Denoise	8.3	6.1	6.1	4.3
Send data	5	5	5	5
Merging	2.1	2	1.9	1.7
Total time	21.8	19.5	19.4	17.1

2.2.5 Discussion

The fusion algorithms presented in this chapter use the same hardware architecture to obtain an optimized output. They can be intended as a piece of software for human tracking, with dedicated inputs and outputs (Figure 2.15). However, depending on the selected algorithm type, human data is processed in different manners.

The skeleton fusion has the advantage of a better efficiency and the capability to recognize human body parts. These features make it the most attractive solution for collaborative robotics where a fast and human-friendly response of the robot is desired. For example, it will be seen that distinguishing the human pose is fundamental for the ergonomics of hand-over tasks and for trajectory conditioning in collision avoidance applications.

The point cloud fusion algorithm is better when a more reliable and precise human size representation is needed. However, it has a larger processing time and does not provide human pose estimation. Despite the limitations, point cloud

fusion is still considered in this work to show the current capabilities of the proposed method in collision avoidance applications and to lay the foundations for future developments.

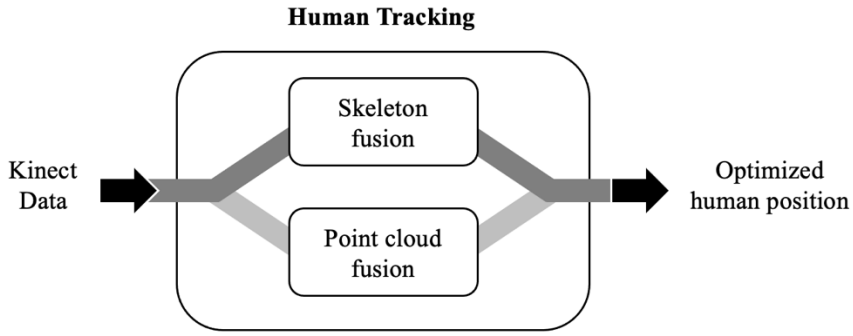


Figure 2.15 Human tracking block. The two ways indicate the possibility to select either skeleton or point cloud fusion. Depending on the selection, the optimized output is in terms of skeleton or point cloud

Chapter 3

Collision Avoidance

Collision avoidance is a main branch of robotics research. It collects strategies and algorithms to let the robot move according to the task without colliding with obstacles. Its most powerful application is in unknown or unstructured environments, where real-time control techniques are required to quickly react to dynamic obstacles by choosing alternative trajectories. Collaborative robotics is one of those application that can take advantage of collision avoidance. However, because the obstacle is a human and not an object, different aspects should be considered.

The aim of collaborative robotics is the safe and efficient accomplishment of a task. The safety is related to preventing potential hazards due to unexpected contacts, while the efficiency refers to task time. Depending on the level of collaboration, handling safety and efficiency can be more challenging.

For example, in sequential collaboration the risk of collision occurs only if the operator accesses accidentally to the shared workspace, when his presence is not expected. Since this may be a rare event, choosing to stop the robot can be an acceptable safety measure that may not significantly affect efficiency, the latter being more related to sequential task planning.

In responsive collaboration, since human and robot work at the same time in a shared workspace, stopping the robot every time a potential collision is detected can have high impact on task time. This can be addressed by collision avoidance, so that the robot carries on the task maintaining a safety distance from the

operator rather than stopping. In this case, collision avoidance is synonymous of both safety and efficiency.

However, a successful collision avoidance implementation, that ensure safety and prevent the robot to stop, may be not sufficient to improve efficiency. In fact, recent studies reveals that the mental engagement of the operator affects the fluency of collaboration and the task time [44], [45]. For example, the operator could feel uncomfortable due to sudden robot behaviors.

Human factors as the emotional state are not addressed by the standards but should be considered to exploit HRC. This is even more important as the level of collaboration increase. Consider responsive collaboration equipped by collision avoidance. The standards do not specify any requirements for the collision-free robot trajectory. However, the velocity and the predictability of the alternative trajectory are perceived by the human.

In addition to the aspects already discussed, the success of collision avoidance strategies lays on some technical challenges. Without loss of generality, consider an anthropomorphic manipulator. If human motion is tracked by a 3D vision system, his position with respect to the robot is given.

In this case, human-robot distances can be monitored to activate collision avoidance when the operator is in the proximity of the robot. It means that the algorithm should continuously check for the minimum distances between two data sets representing the robot and the human positions. Distance calculation may be not trivial, depending on the complexity of the approximation made to represent human and robot bodies. For this reason, it’s common practice to use elementary geometries, e.g. points, spheres and cylinders, to identify the body parts.

Another aspect is that the collision avoidance algorithm should take into account whole robot and human bodies, so that the evasive motion of the robot involves all links, not only the TCP. Assume the robot task defined as a TCP planned trajectory towards a point in the workspace. If a potential collision is detected, whatever robot or human parts are involved, the algorithm should drive TCP away from the obstacle and towards the target, through a collision free motion extended to the entire robotic arm.

Moreover, it is fundamental that the collision avoidance algorithm has low computation time. In fact, high delays between the instant when human is detected and the one when robot reacts may led to collision.

This chapter describes two different collision avoidance algorithms that have been developed trying to address the points above. The first algorithm is designed so that the robot feels a repulsion effect on each link depending on the distance from the operator. The third algorithm is more focused on human factors and wants to address predictability of the robot trajectory.

Before discussing the novelties and the theory behind each method, an overview of the existing robot collision avoidance techniques is presented.

3.1 State of the art

3.1.1 Robot collision avoidance

For a better classification of the existing methods, it is useful to consider collision avoidance as a combination of two problems: trajectory planning and motion planning.

Let the robot task consist of moving from a point to another, according to spatial and task constraints. Spatial constraints are the obstacles or, more in general, the environment; examples of task constraints are the time to reach the target or the energy consumption.

Trajectory planning consists of finding a geometric path (path planning) that satisfies the spatial constraints and endowing it with the time information [46]. Motion planning identifies the process of selecting the robot input such that all constraints are satisfied [47].

For this reason, the differences among collision avoidance methods may be found at different levels. In literature, there are works that focus only the path planning, others that address the trajectory planning and even more that propose different control schemes for motion planning. Due to such a wide topic, this dissertation focuses on a classification based on the application.

Consider the simplest case where spatial constraints are completely defined and do not vary in time. It means that the robot has a priori information on the static environment where it navigates. In this kind of application, the collision-free trajectory can be programmed offline, i.e. before the task has started.

However, even in static environment, the robot does not necessarily have complete information on the geometry of the obstacles. In other words, the spatial constraints may not be completely defined, even if they do not vary in time. If the robot is equipped with sensors, this problem can be dealt with online collision

avoidance methods, which means that the robot is able to detect the obstacle and to plan a collision-free trajectory on the go.

Now consider the most challenging application, with no a priori information on spatial constraints of a dynamic environment. In this case, as the previous one, the robot must be controlled to adjust the path once the obstacle is detected by sensors. However, the obstacle can move and decision on the evasive motion must be taken in a hurry.

A first classification can be made considering *offline* and *online* methods. However, a more common distinction, which is indirectly related to the previous one, refers to *global* and *local* methods [48]. Global methods allow to completely define the robot trajectory from the start point to the goal point and require prior information of the environment. For this reason, global methods are usually offline techniques. On the other hand, local methods consist of observing the proximity of the robot to change the path locally in the presence of an obstacle. For that, local methods are usually suitable for online collision avoidance.

3.1.1.1 Global methods

The most common approach of global methods is to use *optimization problem* (OP) to obtain the best path that satisfies spatial and task constraints [49]. Since infinitely many collision-free paths are usually possible, the criterions used to select the optimal path are most often the minimum time [50], the minimum jerk [51] and the lowest energy consumption [52]. Even a combination of the mentioned criterions can be used. This translates into a cost function to minimize. Depending on the number of variables, finding the global minimum of the cost function can be non-trivial and, most important, time consuming.

A significant example is [53], where the collision-free trajectory of an industrial manipulator is obtained with an approach based on OP. In [53], the objective function considers restrictions associated with the maximum power and torque of the robot, the jerk and the total energy. The author of [53] claims that the proposed approach is efficient since it can compute the best trajectory in $0.49\ s$. This result is relevant compared to the previous work based on similar methods, but clearly shows the limitation of OP.

OP can be formulated in the cartesian space or, more in general, in the C-space (from *configuration space*). For instance, the C-space of a robotic manipulator can be obtained considering the joint angles as the generalized coordinates. In addition to OP, there are global techniques that are based on a

geometric representation of the C-free, i.e. the portion of the C-space corresponding to collision-free robot configurations.

Roadmap techniques map the C-free into a system of one-dimensional curves (the roadmap) that describe adequately the connectivity of C-free. The collision-free path is then obtained by connecting the initial and final configurations through a feasible path [46]. Methods that use roadmaps mainly differs depending on how the roadmap is generated. *Voronoi diagrams*, for example, are a well-known criterion to select the feasible paths considering the Euclidean distance between the robot configuration and the C-obstacle, i.e. the portion of the C-space that maps collision configuration [13].

Another possible use of the C-space is via *cell decomposition*. It consists in dividing the C-space in simply shaped regions to obtain a connectivity graph. The nodes of this graph are the cells, that can be connected through a path which is a subset of C-free [13].

As the OP, roadmaps and cell decomposition are effective methods for global planning, but they have the disadvantages of being slow, especially in high-dimensional problems [54]. For this reason, they are not suitable for online collision avoidance.

3.1.1.2 Local methods

Existing local methods distinguish between adaptations of the global techniques and approaches based on artificial potential fields. In any case, if an obstacle is detected, they do not redefine the entire trajectory, but they act on the input of the robot at each control step so that the system evolves according to the measurements of the sensors.

The local application of OP consists of solving objective functions for the best control input that satisfy some criterion. For example, in [55] the separation distance between the robot a spherical obstacle is monitored with a vision system. At each sample, the solution of the OP is the manipulator velocity that most closely matches the desired velocity while not violating spatial and kinematics constraints. Once the optimal velocity has been found, it is used as control input for the robot. A similar application of OP is [14], where the algorithm is formulated in terms of a QP (from *quadratic programming*) problem whose solution, at each time step, is a set of joint reference accelerations. Recent developments in local use of OP consists of hybrid control schemes. The works [56], [57] combine global exploration and OP to compute the robot motion. In

particular, in [57] the control algorithm is separated in two stages: the path planning stage, which is solved with global techniques, and the joint configuration planning stage, handled by OP.

Probabilistic roadmaps are fast methods that address the high computational cost of the standard roadmaps. In this kind of algorithms, a commonly choice is to follow a rectilinear path between the initial and final configurations in the C-space [13]. By sampling the path with a sufficient resolution, if at a certain sample a possible collision is detected, the algorithm generates a random sample in the C-space. If the sample does not fall into the C-obstacle, it becomes a feasible configuration resulting in a local path variation. The drawback of this methods is that it may fail in narrow passages, i.e. zones surrounded by C-obstacle [13]. In this case, in fact, the probability to place a feasible configuration through random sampling is low and the algorithm may require some time until to obtain a solution.

A method which performs better in narrow passages is the *bidirectional RRT* (from *rapidly-exploring random tree*) [13]. Compared to roadmaps, the RRT explores only a subset of C-free, this resulting in a more efficient scheme.

In general, even if probabilistic roadmaps and RRT are significantly better than global methods in terms of efficiency, they have the disadvantage of planning in the C-space, where the obstacles need to be mapped. This can be computationally complex in dynamic environments, where moving obstacles must continuously be re-mapped [55].

When a fast response to moving obstacle is required, one of the best solutions is represented by the *artificial potential field* (APF). In APF-based methods, the target and the obstacle are modelled with potential functions so that the total potential shows a global minimum at the target and high potential in correspondence of the obstacle [13], [58]. The gradient of the potential function is used as the control vector to obtain a collision-free motion towards the target. Even if APF can also be used as global planner, their most common application is for online local planning. This explain why they have been listed among local methods.

Because of its centrality in the dissertation, the state of the art of APF is summarized in a dedicated chapter, with an extended review of the literature.

3.1.1.3 Artificial potential field

The artificial potential field is a well-known technique which can be used both as a global or a local planner for robots. The bases of obstacle avoidance with artificial potential fields were introduced in [58]. The robot moves under the effect of virtual forces generated by a potential field, built considering the target as the attractor, i.e. the global minimum of the potential, and the obstacles as repulsors, i.e. confined regions with high potential. If the negative gradient is chosen as the input force for the robot dynamics, the system would evolve towards the global minimum.

Thereafter, many studies have been conducted on this method. Major efforts have been focused on solving the local minima problem, which is the main drawback of this approach. A superquadratic potential function to eliminate local minima around the obstacle is proposed in [59]. One of the most interesting theoretical result is described in [15], where the navigation function is introduced to solve local minima. However, the proposed control scheme required prior information, stationary obstacles and fixed destination. An extension of the navigation function has provided an algorithm to calculate the parameters of the navigation function for obstacle avoidance in a sphere world [60]. Other studies that have been inspired by the drawback of the artificial potential fields lead to alternative methods, like the harmonic potential functions [61], the dynamic window [62], the collision cone [63] and the attractor dynamics [64].

Another branch of the potential field research moved towards more practical approaches that propose a different interpretation of the artificial potential. For example, the gradient tracking control based on sliding mode, described in [65], considers the gradient as the desired velocity vector field for the robot; this allows the exact tracking of gradient lines and the convergence to the goal, in contrast with the standard method. In fact, if the force is considered as the robot input, a dissipative term must be added to the control scheme to ensure asymptotic stabilization [58].

In general, the gradient vector field can be coupled with additional vector terms to prevent local minima or to generate convenient paths. For instance, tangential fields can be activated locally to escape from spurious stationary points and guide the robot around the obstacle [66]. Selective attraction and tangential fields can be merged to let the robot approach the goal from a desired direction [67]. Another example is the virtual target approach, that consists of substituting the global target with a local one around the object to avoid local minima [68],

[69]. Alternatively, the robot goal position can be temporarily moved [70] or projected [71] to overcome local minima.

Plenty of literature works studied how to solve or improve potential fields built from a single goal and multiple obstacles but there is a lack of contribute on the use of multiple attractors to make the robot reach the final goal by avoiding obstacles and passing through attractive regions.

An introduction to potential fields with multiple attractors and repulsors is given in [72], where the possibility to model and combine the potential fields of each element by using quadratic or exponential function is discussed. However, [72] is limited to introducing the concept of multiple attractors and does not give an optimal strategy to sculpt the potential field; moreover, it does not distinguish the roles of the global and local attractors.

3.1.2 Collision avoidance methods for collaborative robotics

Human-robot collaboration at the highest level requires fast robot reaction so that local methods are commonly preferred over the global ones. In this section, the most significant developments on collision avoidance algorithms for collaborative robotics are summarized, with a focus on approaches inspired by local methods.

One of the first examples is the application of APF-based techniques to human robot interaction. In [73] a collaborative robot carries out a task as a planned trajectory and the workspace is monitored by a Kinect. When the operator moves nearby the robot, a collision avoidance control scheme based on APF is used to generate an evasive motion of the manipulator. The originality of [73], in addition to the distance calculation algorithm, is the use of repulsive velocities as repulsive vector, instead of the classical APF. Even if this possibility had already been discussed by [65], [73] has shown that it can be effective for HRC. Another example of APF inspired collision avoidance algorithm for collaborative robotics is [74]. The similarity with APF is that [74] computes the robot control vector as the gradient of a scalar function built considering the position and the velocity of the source of danger. The control strategy of [74] is validated with a collaborative robot equipped with Kinect skeleton tracking. A more recent work which exploits APF and repulsive velocities is [75]. In [75], the repulsion effect is adjusted by varying the size of the operator bounding volumes used to approximate his body. In particular, the size of the area of influence around the obstacle is reshaped with a logic similar to [74], which takes into account the velocity of the obstacle. The method [75] has been applied to an industrial collaborative task with promising results. However, the operator is tracked with wearable sensors.

In addition to APF-based techniques, recent studies on collaborative robotics applications investigate the possibility to use local OP or RRT for fast human-robot collision avoidance. The work [76] formulates the robot collision-free trajectory through OP, by calculating at each control step the optimal joint positions vector which satisfies the safety constraints. The method in [76] is applied to a 6dof collaborative robot and the movements of the operator are monitored by Kinect skeleton tracking. Even if the results shows that the local OP is fast compared to global methods, the computing time is still higher than the APF-based approach. In fact, the author of [76] claim that the OP solver runs at $0.01\ s$, which is suitable for real-time. However, APF-based techniques, e.g. [73], [74], are able to calculate the repulsive control vector from 7 to 10 times faster. The authors of [77] describe an improved RRT algorithm that can potentially deal with moving obstacles. The method is tested with a collaborative robot and the human-robot distances are monitored by a Kinect sensor. Even in this case, the efficiency with respect to older RRT techniques is consistent, but the average planning time of $0.079\ s$ makes the improved RRT algorithm still unusable with fast obstacles.

Due to the limitations of other local methods, APF-based approaches remain the most popular solution for responsive collaboration. This trend emerges also in the detailed overview of safety systems for HRC [78].

3.2 Contribution to the current state of knowledge

This thesis discusses two different collision avoidance algorithms based on APF. The algorithms are summarized in Table 1.1.

The *Robot links - human* collision avoidance algorithm guarantees collision-free trajectories though a repulsive action of the obstacles, which is applied to the robot in terms of repulsive velocities. This algorithm is not novel, since it has been already introduced in [12], [79], [80], but within this work some practical ways for its implementation have been analysed and the results are reported in the dedicated chapter. A first application regards the study of an assembly task, which has inspired the latest publication [81]. In fact, the previous works [12], [79], [80] formulates and validates the algorithm from theoretical and technical point of views, but does not demonstrate how it can be used in a collaborative task to exploit the advantages of responsive collaboration. A second relevant result is about the algorithm testing with human point cloud as described in [36], that was not discussed in [12].

The most relevant advance obtained in this work is a new algorithm, identified as MAP algorithm, which is a novel technique that addresses the robot trajectory conditioning problem. In collaborative robotics, this problem involves human factors and it is based on recent studies [44], [45], which proves that it is not only important to drive the robot away from the obstacles, but also controlling the alternative trajectory can be fundamental. In particular, a more fluent collaboration is observed when the robot motion is predictable [45]. In this context, a predictable motion is defined as a functional motion that matches what the human would expect, given the robot goal [44]. When the robot moves undisturbed, anticipating the robot intention can be easier for humans because of repeatability of the robot task. Major issues arise when the robot chooses alternative path to avoid collision with unexpected obstacles, e.g. a moving object or the human's body itself. If the robot reaction is not controlled in some way, the trajectory becomes unpredictable for the human, even if he knows the robot final goal. To make it predictable, the robot could be controlled to behave in a repeatable way when avoiding obstacles, for example passing always through defined regions. In this way, the human can anticipate how the robot choose the collision-free trajectory. For this reason, the author of [12] has investigated the possibility to conditionate the robot alternative path. In [12], attractive and repulsive geometries like plane and cylinders are opportunely placed nearby the human skeleton to force the robot through preferred regions. However, the method in [12] has the limitation of being not precise, so that the effect on the trajectory may not be sufficient to generate predictable behaviors. The importance of finding a simple and effective solution to deal with obstacles in a more controlled manner motivated the author of this dissertation to develop a novel method.

In contrast with [12] and with the *Robot links - human* collision avoidance algorithm, the MAP does not use repulsive and attractive velocities, but require an optimal potential field to be sculpted so that the robot is driven towards the goal by following the negative gradient direction. The robot goal is represented as an attractor which is modelled to be the global minimum of the potential field. The obstacles are described as confined regions with high potential. The main aspect that distinguishes the MAP from the previous works on artificial potential fields is the addition of strategical attractive points, that are modelled as optimal deflections of the potential field, i.e. without being local minima. To the author knowledge, the problem of finding an optimal solution for the coexistence of attractors and repulsors has not been investigated in previous works. The MAP represents a solution to this problem in an original form.

3.3 Robot links - human collision avoidance algorithm

Consider a 6dof robotic manipulator whose generic pose is identified by $\boldsymbol{\chi}_e = [\boldsymbol{p}_e \ \boldsymbol{\phi}_e]^T$, where $\boldsymbol{p}_e$ is the (3×1) vector that defines the position of the end-effector and $\boldsymbol{\phi}_e$ is the (3×1) vector that defines orientation of the end-effector with respect to the world frame $O - xyz$. Assume that the robot task is given to the end-effector in terms of final position $\boldsymbol{p}_f$ and orientation $\boldsymbol{\phi}_f$. In particular, the robot starting from the pose $\boldsymbol{\chi}_s = [\boldsymbol{p}_s \ \boldsymbol{\phi}_s]^T$ must navigate towards the final pose $\boldsymbol{\chi}_f = [\boldsymbol{p}_f \ \boldsymbol{\phi}_f]^T$. In the absence of obstacles, the simplest path is the straight line connecting $\boldsymbol{p}_s$ to $\boldsymbol{p}_f$. In general, every path can be approximated to a sequence of $N+1$ points $\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots \boldsymbol{p}_N$ connected by N segments. If t_0 and t_N are respectively the initial and final time instants of the trajectory, for $t_0 < t < t_N$ offline trajectory planning techniques can be used to associate the $N+1$ points along the path to the values $\dot{\boldsymbol{p}}_d(t)$ and $\dot{\boldsymbol{\phi}}_d(t)$ corresponding to a desired velocity profile [13].

Once the trajectory related to the task has been planned as $\dot{\boldsymbol{\chi}}_d = [\dot{\boldsymbol{p}}_d(t) \ \dot{\boldsymbol{\phi}}_d(t)]^T$, a possible control scheme for the robot is the inverse kinematics algorithm with Jacobian inverse [13]. The control vector is chosen as the vector of joint velocities $\dot{\boldsymbol{q}}$ defined by:

$$\dot{\boldsymbol{q}} = J^{-1}(\boldsymbol{q})(\dot{\boldsymbol{\chi}}_d + \boldsymbol{K}_J \boldsymbol{e}) \tag{3.1}$$

where $\boldsymbol{q}$ is the (6×1) vector of joint positions, $J^{-1}(\boldsymbol{q})$ is the Jacobian inverse, $\boldsymbol{K}_J$ is a positive definite (6×6) matrix and $\boldsymbol{e}$ is the (6×1) vector of the operational space error between the desired and the actual end-effector position and orientation:

$$\boldsymbol{e} = \boldsymbol{\chi}_d - \boldsymbol{\chi}_e \tag{3.2}$$

In the absence of obstacles, the control law (3.1) ensures the convergence to the goal. For instance, if a rectilinear trajectory is chosen, the end-effector traces the path in Figure 3.1a.

Consider an obstacle entering in the robot workspace, so that it intercepts the planned trajectory during the task (Figure 3.1b). The idea behind the collision avoidance algorithm is to add a repulsive velocity term in the control law which depend on the distance between the robot and the obstacle. For example, if the

distance vector between the end-effector and the obstacle $\boldsymbol{d}_{e-o} = \boldsymbol{p}_e - \boldsymbol{p}_o$ is considered, the linear repulsive velocity $\boldsymbol{v}_{rep}$ can be modelled as:

$$\boldsymbol{v}_{rep} = v_{rep} \frac{\boldsymbol{d}_{e-o}}{\|\boldsymbol{d}_{e-o}\|} \tag{3.3}$$

$$v_{rep} = \frac{v_{rep,max}}{1 + e^{(\|\boldsymbol{d}_{e-o}\|(2/\rho)-1)\alpha}} \tag{3.4}$$

where v_{rep} is the magnitude of the linear repulsive velocity, $v_{rep,max}$ is the maximum magnitude of the linear repulsive velocity, α and ρ are the parameters that regulates the variability of v_{rep} with $\|\boldsymbol{d}_{e-o}\| = d_{e-o}$. An example is shown in Figure 3.2.

The repulsive velocity term for the control law is then obtained as:

$$\dot{\boldsymbol{\chi}}_{rep} = \begin{bmatrix} \boldsymbol{v}_{rep} \\ \mathbf{0} \end{bmatrix} \tag{3.5}$$

where $\mathbf{0}$ is the (3×1) null vector. Thus, the control law becomes:

$$\dot{\boldsymbol{q}} = J^{-1}(\boldsymbol{q})(\dot{\boldsymbol{\chi}}_d + \boldsymbol{K}\boldsymbol{e} + \dot{\boldsymbol{\chi}}_{rep}) \tag{3.6}$$

However, (3.3) and (3.4) considers only the distance between the end-effector and the obstacle. Moreover, due to the full Jacobian in (3.6), the repulsive action affects the end-effector only (see Figure 3.1b). The strategy proposed in [12] permits to extend the repulsive effect to the entire robot. In particular, the manipulator is identified by a finite number of control points displaced along the kinematic chain (Figure 3.1c). Each link of the robot is intended as a set L_i of control points, where the subscript i indicates the i-th set. For each set L_i, one can measure the closest control point to the obstacle to find the minimum distance $\boldsymbol{d}_{Li-o}$ between the corresponding link and the obstacle. Thus, a repulsive velocity $\boldsymbol{v}_{Li}$ is applied at the closest control point of each link (Figure 3.1d):

$$\boldsymbol{v}_{Li} = v_{Li} \frac{\boldsymbol{d}_{Li-o}}{\|\boldsymbol{d}_{Li-o}\|} \tag{3.7}$$

$$v_{Li} = \frac{v_{rep,max}}{1 + e^{(\|\boldsymbol{d}_{Li-o}\|(2/\rho)-1)\alpha}} \tag{3.8}$$

The repulsive effect related to each link, is converted to the joint space:

$$\dot{\boldsymbol{q}}_{Li} = J_{Li}^{-1}(\boldsymbol{q}_{Li})\dot{\boldsymbol{\chi}}_{Li} \tag{3.9}$$

where J_{Li}^{-1} is the partial Jacobian related to the closest control point of the link L_i and $\dot{\boldsymbol{\chi}}_{Li}$ is the corresponding repulsive velocity term, obtained using $\boldsymbol{v}_{Li}$ in (3.5). The control law (3.6) is modified as follows:

$$\dot{\boldsymbol{q}} = J^{-1}(\boldsymbol{q})(\dot{\boldsymbol{\chi}}_d + \boldsymbol{K}\boldsymbol{e}) + \sum_{Li} \dot{\boldsymbol{q}}_{Li} \tag{3.10}$$

In (3.10), the first term on the right side is related to the task while the summation over each set L_i provide the repulsive action.

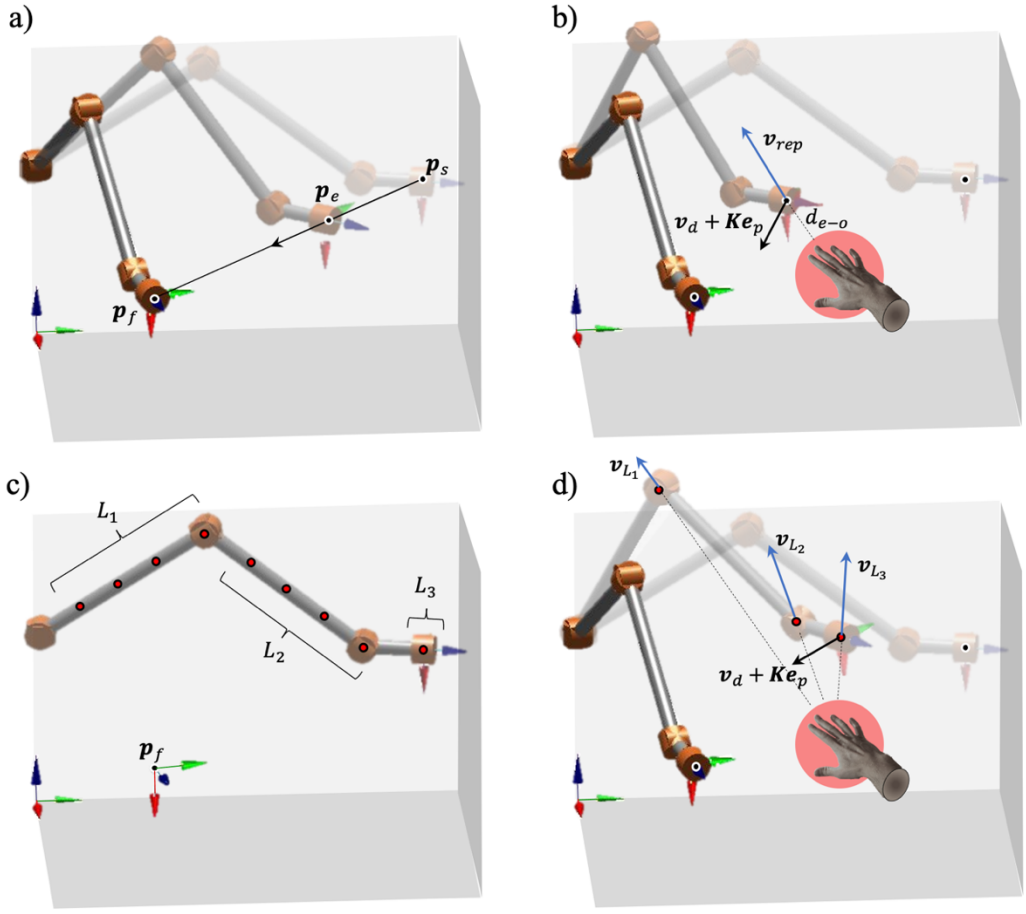


Figure 3.1 A 6dof manipulator represented as a kinematic chain in Matlab through the Corke Robotic Toolbox [82]. The robot is plotted in various configurations with different opacity to give the idea of motion

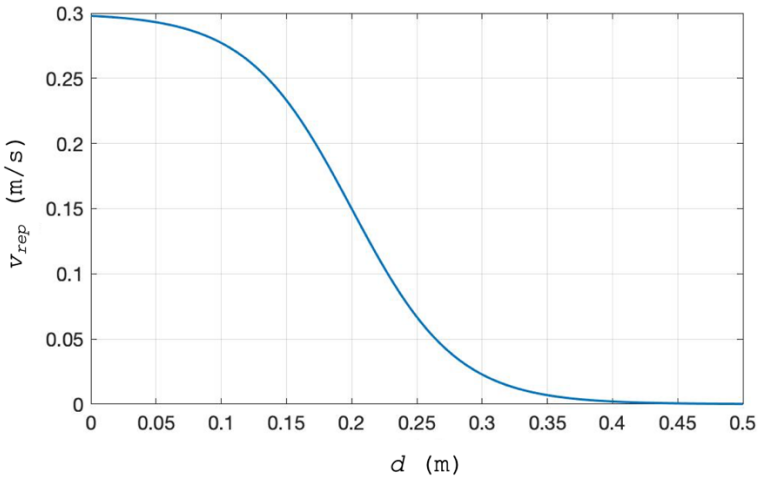


Figure 3.2 Magnitude of the repulsive velocity v_{rep} versus the generic distance d. The curve is obtained with $v_{rep,max} = 0.3\ m/s$, $\rho = 0.4\ m$ and $\alpha = 0.4$.

3.4 Multiple attractors potential (MAP)

In this section, the Multiple Attractors Potential (MAP) is presented. First, the approach is described in $\mathbb{R}^2$. The potential field is modelled by means of classic quadratic and exponential functions ([58], [72]). Thus, the parameters for the optimal tuning are discussed. Finally, the method is extended to $\mathbb{R}^3$.

3.4.1 Introduction to the MAP

Consider the robot end-effector as a point in the two-dimensional cartesian space, whose task is to reach the final position $\boldsymbol{p}_f$ from a starting position $\boldsymbol{p}_s$. The final position can be seen as an attractive potential field U_f which is modelled with the quadratic function ([58]):

$$U_f(\boldsymbol{p}) = \frac{1}{2}\sigma\|\boldsymbol{p} - \boldsymbol{p}_f\|^2 \tag{3.11}$$

where σ is a positive parameter which regulates the intensity of the quadratic function and $\boldsymbol{p} = [x\ y]^T$ represents the generic point in the cartesian space. In (3.11), it is written $U_f(\boldsymbol{p})$ to exploit the dependency on $\boldsymbol{p}$. In general, it is $U_f(\boldsymbol{p}, \boldsymbol{p}_f, \sigma)$ but $\boldsymbol{p}_f$ is given by the task and the parameter σ is supposed to be chosen. To simplify the reading, hereafter, only the spatial variable $\boldsymbol{p}$ is exploited, except where otherwise specified.

Suppose the presence of an obstacle. In general, it may have any geometry. Sometimes it is convenient to approximate objects by composing simple shapes,

i.e. spheres, cylinders and planes. Other applications may need more accurate potential functions to describe the obstacles. The method presented in this chapter can be applied to any geometry. For the discussion, only the case of a spherical object, i.e. the disc in two dimensions, is considered. The reader is referred to the literature for general obstacle modelling ([15], [58], [59], [83]).

The obstacle is centred in $\boldsymbol{p}_o$, with radius R_o (Figure 3.3a). It produces a repulsive potential field U_o, which is modelled with the exponential function ([72]):

$$U_o(\boldsymbol{p}) = \beta_o e^{-\frac{\gamma_o}{2}\|\boldsymbol{p}-\boldsymbol{p}_o\|^2} \tag{3.12}$$

where β_o and γ_o are the positive parameters that determine the shape of the gaussian around the obstacle; in particular, β_o is the peak value, while γ_o is the exponential decay parameter. In Figure 3.3a, the outer circle centred in $\boldsymbol{p}_o$ identifies the active region U_o^*, defined as the circle with radius R_o^*, so that the gradient of U_o goes to zero outside U_o^* and grows rapidly at the contour of the obstacle, at a radius R_o.

If only the final position and the obstacle are considered, the resulting total potential field U_{fo} can be written as:

$$U_{fo}(\boldsymbol{p}) = U_f + U_o = \frac{1}{2}\sigma\|\boldsymbol{p}-\boldsymbol{p}_f\|^2 + \beta_o e^{-\frac{\gamma_o}{2}\|\boldsymbol{p}-\boldsymbol{p}_o\|^2} \tag{3.13}$$

The generic potential field U_{fo} is shown in Figure 3.4a. The control law can be chosen so that the command vector has the direction of the negative gradient. In the specific case of Figure 3.3a, if $\boldsymbol{p}_s$, $\boldsymbol{p}_o$ and $\boldsymbol{p}_f$ are aligned, by following the negative gradient the robot can potentially stuck at the classical saddle point ([15]). This is a limit case. In fact, in presence of a small perturbation in the y direction, the robot can potentially trace either the continue or the dashed path.

In this thesis, the idea is to introduce an attractive source $\boldsymbol{p}_a$ nearby the obstacle, as depicted in Figure 3.3b. To distinguish the local attractive effect of $\boldsymbol{p}_a$ from the global one related to the final position, $\boldsymbol{p}_a$ is called “local attractor”, while the name “global attractor” is used to identify $\boldsymbol{p}_f$. The local attractor potential field U_a is modelled with the negative exponential function ([72]):

$$U_a(\boldsymbol{p}) = -\alpha_a e^{-\frac{\gamma_a}{2}\|\boldsymbol{p}-\boldsymbol{p}_a\|^2} \tag{3.14}$$

where α_a and γ_a are the positive parameters that regulate the intensity and the decay of the attractive effect. In particular, γ_a can be chosen so that the attractive gradient assumes an appreciable value at R_a, which identifies the radius of the local attractor. This aspect deserves a further discussion. In fact, for the obstacle it is straightforward to associate to R_o the physical meaning of the contour of the object, i.e. a region that the robot should not enter. This also explain the choice of a thicker line to represent the obstacle contour in Figure 3.3. On the other hand, R_a has not a tangible match in the real world but represents the region where the robot starts perceiving the attraction towards $\boldsymbol{p}_a$. In other words, R_a indicates the distance from $\boldsymbol{p}_a$ where the magnitude of the gradient of U_a is not negligible when combined with the one of U_{fo}.

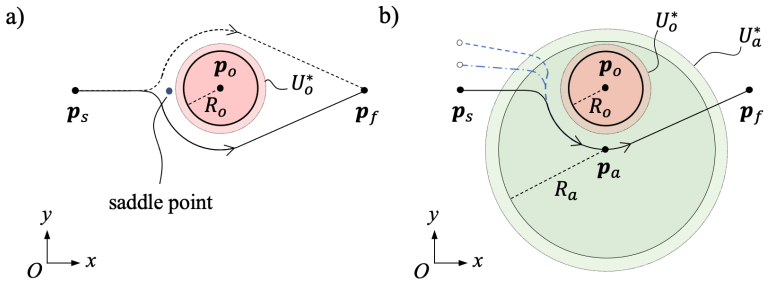


Figure 3.3 Two-dimensional analysis of the possible robot paths obtained by following the gradient lines of a quadratic potential function, with a single obstacle and a local attractor modelled with the exponential functions. a) In presence of the obstacle, the robot can potentially follow either the dashed or the solid lines; in the same figure, the classical saddle point is shown. b) The robot path is influenced by the local attractor placed on the obstacle side; the styles of the lines identify alternative paths related to different starting positions, i.e. different approaching directions of the robot towards the obstacle

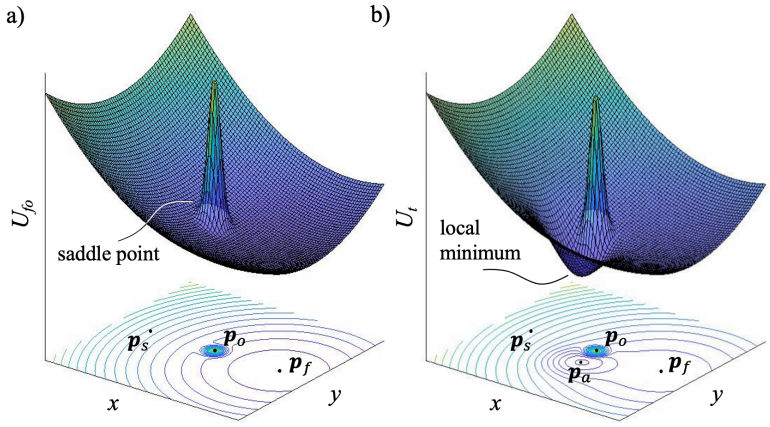


Figure 3.4 Influence of the exponential repulsive and attractive potential fields on a quadratic potential field; the xy-plane, in addition to the significant points, contains the isolines of the potential field. a) Case with a single obstacle; in the same figure, the classical saddle point is shown. b) Case with a single obstacle and a local attractor; the high intensity of the local attractor generates a local minimum, as indicated by the isolines

In Figure 3.3b, the outer circle centred in $\boldsymbol{p}_a$ identifies the active region U_a^*, defined as the circle with radius R_a^* so that the gradient of U_a goes to zero outside U_a^*. The parameters α_a and γ_a can be opportunely tuned to design the active region. For example, U_a^* can be extended all around the obstacle to guide the robot obstacle avoidance on the local attractor side for a good range of approaching directions, i.e. for different $\boldsymbol{p}_a$ (see the path lines in Figure 3.3b).

The total potential field with the obstacle and the two attractors becomes:

$$\begin{aligned} U_t(\boldsymbol{p}) &= U_f + U_o + U_a \\ &= \frac{1}{2}\sigma\left\|\boldsymbol{p}-\boldsymbol{p}_f\right\|^2 + \beta_o e^{-\frac{\gamma_o}{2}\|\boldsymbol{p}-\boldsymbol{p}_o\|^2} - \alpha_a e^{-\frac{\gamma_a}{2}\|\boldsymbol{p}-\boldsymbol{p}_a\|^2} \end{aligned} \tag{3.15}$$

A generic potential field U_t is shown in Figure 3.4b. The attractive source $\boldsymbol{p}_a$ acts bending the potential on its side.

A local minimum may result if α_a and γ_a are not adequately tuned and the robot would stop at the equilibrium point close to $\boldsymbol{p}_a$. Anyway, there exist some values of α_a and γ_a for which the total potential field deflates near $\boldsymbol{p}_a$ without suffering local minima (Figure 3.5). If this happens, the potential field U_t is a *MAP*, i.e. $U_t = U_{t,MAP}$.

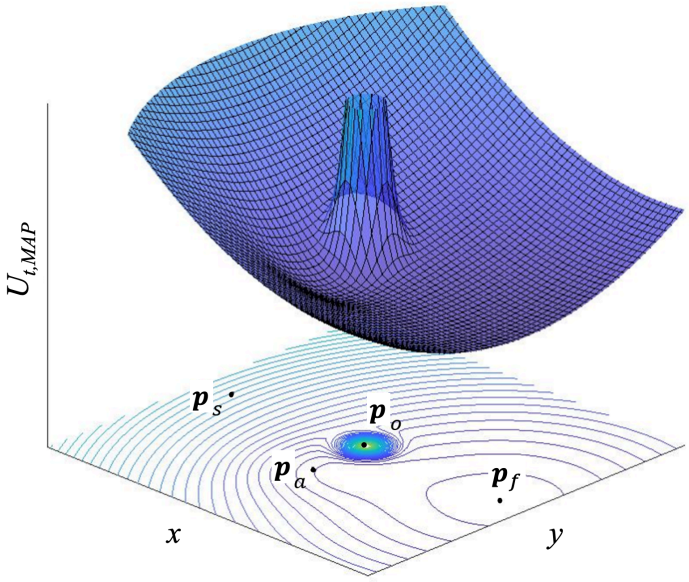


Figure 3.5 MAP resulting from a single obstacle and a local attractor. The intensity of the local attractor is limited so that the local minimum does not show, as indicated by the isolines

For simplicity, the case with single obstacle $\boldsymbol{p}_o$ and single $\boldsymbol{p}_a$ is considered in the next section, where the formula to obtain $U_{t,MAP}$ is presented. The name "multiple attractors", in this example, refers to the coexistence of the global attractor and the local one. However, the approach can be extended even combining a number n of local attractors $\boldsymbol{p}_{a,i}$, with $i = 1,2,\dots,n$, around any number m of obstacles $\boldsymbol{p}_{o,p}$, with $p = 1,2,\dots,m$. A generalization of the *MAP* will be given at the end of the chapter.

3.4.2 MAP parameters tuning

In this section, the method to sculpt the potential field to obtain the MAP is presented. The final target $\boldsymbol{p}_f$ is modelled with the quadratic function U_f. The purpose is to find the values of α_a and γ_a which generate the maximum deflection nearby the obstacle without that the local minimum due to $\boldsymbol{p}_a$ occurs. First, the strategy to choose the obstacle parameters β_o and γ_o is discussed. Then, the formulas to calculate the optimal values of α_a and γ_a are deduced for the local attractor.

Given the position of the disc obstacle $\boldsymbol{p}_o$, its potential field U_o can be modelled with the exponential function (3.12). By considering the repulsive gradient modulus as a design variable, the obstacle design ratio λ_o is introduced as:

$$\lambda_o(R_o, \gamma_o) = \frac{|\nabla U_o|_{R_o}}{|\nabla U_o|_{max}} \tag{3.16}$$

where $|\nabla U_o|_{R_o}$ is the modulus of the gradient at the distance R_o from $\boldsymbol{p}_o$, and $|\nabla U_o|_{max}$ is the maximum magnitude of the gradient (see Appendix A for further details). Usually, it is required that the potential grows with a sufficient rate at the disc contour since it is not necessary that U_o assumes an infinite value at the edge of the obstacle ([59]). Thus, given R_o and by fixing a suitable λ_o, one can find the value of γ_o which satisfies (3.16):

$$\gamma_o = -\frac{1}{R_o^2} W_{-1}\left(-\frac{\lambda_o^2}{e}\right) \tag{3.17}$$

where W_{-1} is the solution corresponding to the lower branch of the Lambert W function (see Appendix B for further details). In other words, (3.17) gives the value of γ_o so that the gradient at R_o is λ_o times the $|\nabla U_o|_{max}$. Intuitively, one

can choose $\lambda_o = 1$ to set the maximum gradient at the contour; then, β_o can be arbitrarily chosen to regulate the gradient magnitude. In practice it will be seen that the robot reacts to the obstacle even with $\lambda_o < 1$ for certain values of β_o.

Furthermore, once γ_o and β_o have been selected, it is useful to estimate the active region U_o^*. The radius R_o^* can be calculated by solving $|\nabla U_o| = s_\epsilon$, where s_ϵ is a small positive value, hereafter named "zero threshold" (see Appendix C for further details):

$$R_o^* = \left[-\frac{1}{\gamma_o} W_{-1}\left(-\frac{s_\epsilon^2}{\beta_o^2 \gamma_o}\right)\right]^{1/2} \tag{3.18}$$

A practical example, which will be studied more in details in the results chapter, is presented. In Figure 3.6a the function U_o in the radial direction of a disc obstacle is shown, for two different values of λ_o. If the gradient vector field is utilised to generate the robot path, e.g. [65], it can be convenient to choose the value $\lambda_o = 0.2$ to concentrate U_o^* near the obstacle. In this case, in fact, a small λ_o can be sufficient to bend the gradient lines around the obstacle. On the other hand, if the negative gradient is used to produce repulsive forces on the robot, e.g. [58], the choice $\lambda_o = 1$ is justified but translates into a wider U_o^*, whose limit in Figure 3.6a is represented by the dotted circle.

Once the obstacle is identified, the attractor $\boldsymbol{p}_a$ can be placed near $\boldsymbol{p}_o$ to affect the collision avoidance path. It is assumed that $\boldsymbol{p}_a$ is sufficiently far from the active region U_o^*. Without loss of generality the following relation must hold:

$$\|\boldsymbol{p}_a - \boldsymbol{p}_o\| > R_o^* + \tilde{\varepsilon} \tag{3.19}$$

so that the problem of finding the local minimum related to U_a will simplify. The meaning and the lower bound of the positive quantity $\tilde{\varepsilon}$ will be pointed later. Moreover, the following assumption which regulates the size of U_a^* must be satisfied:

$$\|\boldsymbol{p}_a - \boldsymbol{p}_f\| \geq R_a^* \tag{3.20}$$

otherwise, the global minimum would be perturbed. The constraints (3.19) and (3.20) are acceptable and do not represent a limit in practical applications, as will be shown in the results section. In Figure 3.7 a generic case which satisfies conditions (3.19) and (3.20) is depicted.

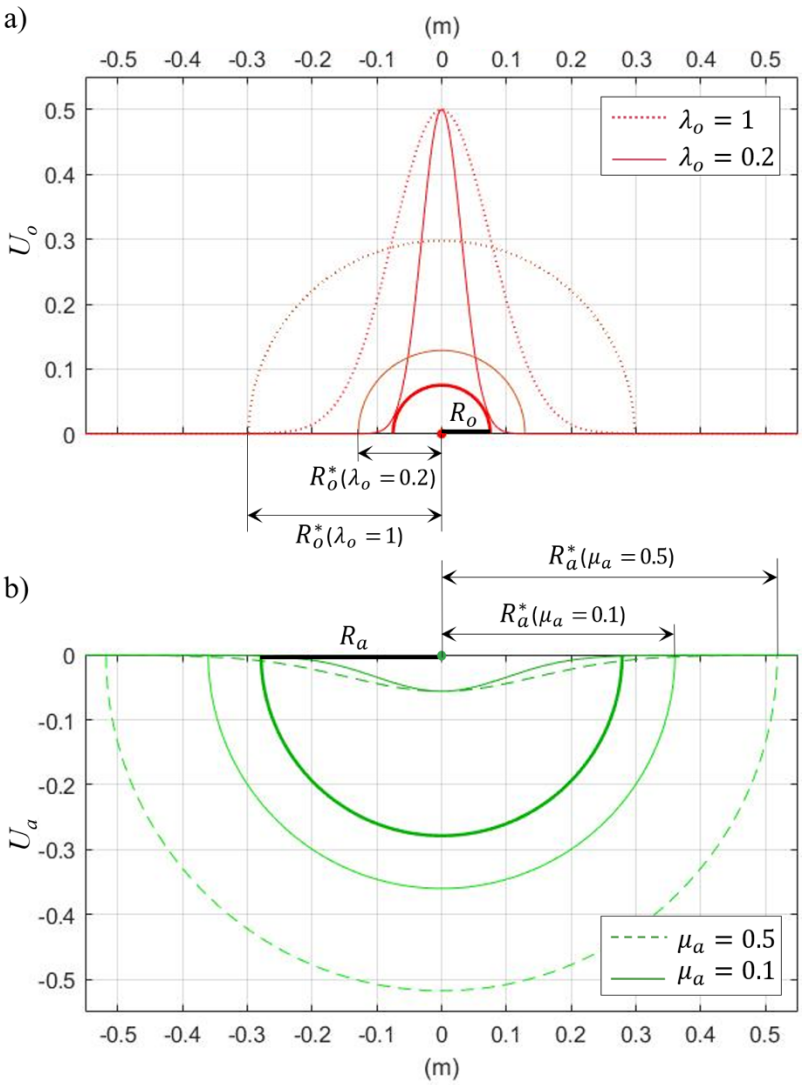


Figure 3.6 Numeric example which shows the influence of λ_o and μ_a on the shape of the repulsive and attractive exponential functions. a) Function U_o in the radial direction of a disc obstacle with $R_o = 0.075\ m$, $\beta_o = 0.5$ and the active region defined by $s_\epsilon = 10^{-2}$; to visualize the extension of the obstacle and of U_o^*, the circles at R_o and R_o^* are projected on the plane where the potential curve is plotted. b) Function U_a in the radial section, with $R_a = 0.2786\ m$, $\alpha_a = 0.0558$ and the active region defined by $\mu_\epsilon = 10^{-2}$; to visualize the extension of the attractor, the circles at R_a and R_a^* are projected on the plane where the potential curve is plotted

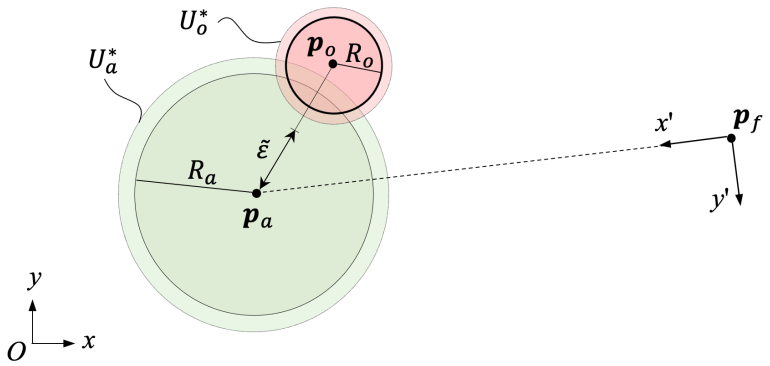


Figure 3.7 Generic case with the obstacle and the local attractor potential functions designed according to the MAP geometrical constraints

When placing $\boldsymbol{p}_a$ it is desirable to shape the function U_a so that the magnitude of the attractive gradient $|\nabla U_a|_{R_a}$ at a certain distance R_a is an appreciable fraction of the maximum value $|\nabla U_a|_{max}$. The local attractor design ratio μ_a is introduced as:

$$\mu_a(R_a, \gamma_a) = \frac{|\nabla U_a|_{R_a}}{|\nabla U_a|_{max}} \tag{3.21}$$

with the same meaning of (3.16) but related to the attractive potential (see Appendix D for further details). By choosing the R_a and a desired μ_a, one can calculate:

$$\gamma_a = -\frac{1}{R_a^2} W_{-1}\left(-\frac{\mu_a^2}{e}\right) \tag{3.22}$$

The active region U_a^* is determined by finding the distance from $\boldsymbol{p}_a$ in which the gradient magnitude goes to zero. For the attractor, it is convenient to write this condition by solving for the distance where the attractive gradient becomes a small fraction μ_ϵ of the maximum value, where μ_ϵ is named "zero-ratio threshold" (see Appendix E for further details):

$$R_a^* = \left[-\tfrac{1}{\gamma_a} W_{-1}\left(-\tfrac{\mu_\epsilon^2}{e}\right)\right]^{1/2} \tag{3.23}$$

In practical application, if μ_ϵ is sufficiently small, R_a^* obtained with (3.23) is also a distance where $|\nabla U_a|$ is negligible; thus, formulations (3.18) and (3.23) do not play different roles in practice. However, the distinction is made for the following reasons. From a theoretical point of view, definition (3.18) is more rigorous; this

will be useful to deal with the local minima problem. On the other hand, (3.23) does not depend on α_a and this will bring two advantages: first, in the design phase, the geometrical assumptions (3.20) can be quickly verified before that the optimal α_a is calculated; second, the analytic solution of the optimal parameters in the case of dynamic attractors will be simplified, as will be seen later.

In Figure 3.6b, a practical example is shown for the local attractor. Here the function U_a in the radial section is plotted for two different values of μ_a. Even in this case, decreasing μ_a allows to concentrate the effect within R_a. Notice that the curves are obtained with $\alpha_a = 0.0558$, that at this point of the discussion has an indicative value only.

To calculate the optimum value of α_a, the local minimum problem related to $\boldsymbol{p}_a$ is discussed. This can be solved by considering the function U_{fa}, defined as the sum of the two potential fields of the attractors:

$$U_{fa}(\boldsymbol{p}) = U_f + U_a = \frac{1}{2}\sigma\|\boldsymbol{p} - \boldsymbol{p}_f\|^2 - \alpha_a e^{-\frac{\gamma_a}{2}\|\boldsymbol{p}-\boldsymbol{p}_a\|^2} \tag{3.24}$$

In fact, because of assumption (3.19), the local minimum will appear in the region where $U_{fa} \cong 0$. By studying the points where the gradient vanishes, the condition is:

$$\frac{\partial}{\partial \boldsymbol{p}} U_{fa}(\boldsymbol{p}) = \sigma(\boldsymbol{p} - \boldsymbol{p}_f) + \alpha_a\gamma_a(\boldsymbol{p} - \boldsymbol{p}_a)e^{-\frac{\gamma_a}{2}\|\boldsymbol{p}-\boldsymbol{p}_a\|^2} = 0 \tag{3.25}$$

The problem can be further simplified by writing the system (3.25) in the auxiliary reference frame $O' - x'y'$ with origin in $O' \equiv \boldsymbol{p}_f$ and whose x'-axis is aligned with $\boldsymbol{p}_a$ (see Figure 3.7). Thus, by considering $\boldsymbol{p}'_f = [0\ 0]^T$ and $\boldsymbol{p}'_a = [x'_a\ 0]^T$ the system becomes:

$$\frac{\partial}{\partial x'} U_{fa}(\boldsymbol{p}') = \sigma x' + \alpha_a\gamma_a(x' - x'_a)e^{-\frac{\gamma_a}{2}[(x'-x'_a)^2+y'^2]} = 0 \tag{3.26}$$

$$\frac{\partial}{\partial y'} U_{fa}(\boldsymbol{p}') = \sigma y' + \alpha_a\gamma_a y' e^{-\frac{\gamma_a}{2}[(x'-x'_a)^2+y'^2]} = 0 \tag{3.27}$$

where $\boldsymbol{p}' = [x'\ y']^T$ is the spatial variable which identifies a point in the auxiliary reference frame. Since σ, α_a and γ_a are positive, (3.27) is true only for $y' = 0$. This suggests that the local minimum must lie on the x'-axis. Therefore, the

problem can be studied in one dimension. In fact, by considering $y' = 0$, (3.26) reduces to:

$$\frac{\partial}{\partial x'} U_{fa}(x', 0) = \sigma x' + \alpha_a \gamma_a (x' - x'_a) e^{-\frac{\gamma_a}{2}(x' - x'_a)^2} = 0 \tag{3.28}$$

Given σ, x'_a and γ_a, equation (3.28) is parametric in α_a and the number of solutions for x' depend on α_a. This can be visualized by plotting the solution in a graphical fashion. By considering:

$$A = \sigma x' \quad , \qquad B = \alpha_a \gamma_a (x' - x'_a) e^{-\frac{\gamma_a}{2}(x' - x'_a)^2} \tag{3.29}$$

the graphical solution is shown in Figure 3.8b, for fixed σ, x'_a and γ_a. In Figure 3.8a, the resulting potential $U_{fa}(x', 0)$ is plotted. For small values of α_a, the curves identifying $-A$ and B have only one intersection point at the global minimum, i.e. in $x' = 0$. By Increasing the value of α_a, the potential starts bending around $x' = x'_a$. The value of $\alpha_a$for which $U_{fa}(x', 0)$ shows a saddle point in $x' = \tilde{x}'$, i.e. when the curves $-A$ and B become tangent (dashed line), is then the upper bound $\tilde{\alpha}_a$. For $\alpha_a > \tilde{\alpha}_a$, the green and the blue curves intersect in 3 points, i.e. at the global minimum and at the local stationary points; in this case the local minimum lies in the interval $0 < x' < x'_a$.

The important result is that, given σ, x'_a and γ_a, if $\alpha_a < \tilde{\alpha}_a$ no local stationary points occur. Hereafter, the saddle point represents the theoretical limit in this sense: the maximum admissible depression near $\boldsymbol{p}'_a$, i.e. the maximum attraction, is obtained by choosing an α_a just below $\tilde{\alpha}_a$.

Notice that, except where otherwise specified, the saddle point here discussed is the one related to the local attractor; in fact, the one introduced in Figure 3.3a and Figure 3.4a has a different meaning and it is identified as the “classical” saddle point.

To find $\tilde{\alpha}_a$, the condition which generates the saddle is studied. In the saddle point, the first and the second derivative must vanish. Therefore, (3.28) must hold together with the following:

$$\frac{\partial^2}{\partial^2 x'} U_{fa}(x', 0) = \sigma + \left[\frac{1}{\gamma_a} - (x' - x'_a)^2\right] \alpha_a {\gamma_a}^2 e^{-\frac{\gamma_a}{2}(x' - x'_a)^2} = 0 \tag{3.30}$$

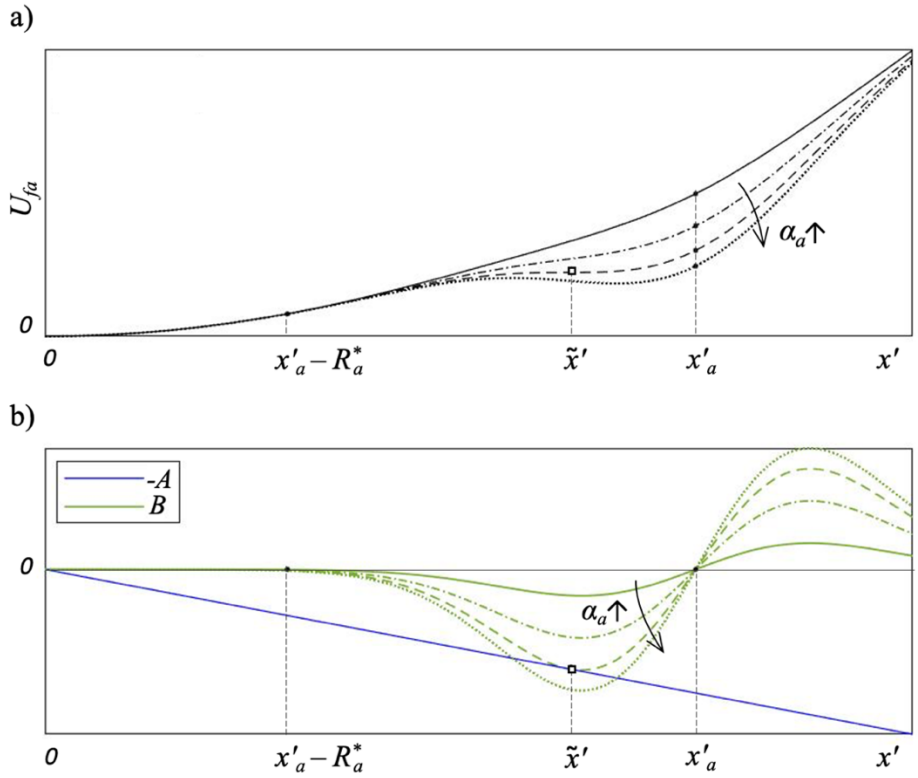


Figure 3.8 Analysis of the parametric solution of equation (3.28). The different style of the lines refers to different values of α_a; the square identifies the saddle point related to the local attractor. a) Potential function U_{fa} along the x' axis; the saddle point exists for the dashed line, at $\tilde{x}'$. b) Graphical solution by means of the intermediate variables (3.29); the saddle point is represented by the point of tangency

The system of equations (3.28) and (3.30) is verified if (see Appendix F):

$$(\gamma_a)x'^3 - (2x'_a\gamma_a)x'^2 + (x'^2_a\gamma_a)x' - x'_a = 0 \tag{3.31}$$

which gives the $x' = \tilde{x}'$ where the inflection arises (see Figure 3.8). The cubic (3.31) has 3 real solutions for x' if:

$$x'_a > \sqrt{\frac{27}{4\gamma_a}} \tag{3.32}$$

In this case, the meaningful solution is:

$$\begin{aligned} \tilde{x}'(x'_a, \gamma_a) &= \frac{2}{3}x'_a\left[\cos\left(\frac{\vartheta + 4\pi}{3}\right) + 1\right] \\ \vartheta(x'_a, \gamma_a) &= \cos^{-1}\left(\frac{27}{2\gamma_a x'^2_a} - 1\right) \end{aligned} \tag{3.33}$$

And by substituting (3.33) in (3.28) it results:

$$\tilde{\alpha}_a(\sigma, x'_a, \gamma_a) = \frac{-\sigma\tilde{x}'}{\gamma_a(\tilde{x}' - x'_a)e^{-\frac{\gamma_a}{2}(\tilde{x}'-x'_a)^2}} \tag{3.34}$$

where it is stressed the dependency of $\tilde{\alpha}_a$ from σ, x'_a and γ_a. Solution (3.34) is related to the case (3.32), which says that the distance between the desired point and the attractor x'_a must be greater than a certain value. However, if assumption (3.20) holds, in practice (3.32) is always satisfied, as can be seen by substituting (3.23) into (3.20) and comparing with (3.32):

$$-W_{-1}\left(-\frac{\mu_\epsilon^2}{e}\right) > \frac{27}{4} \tag{3.35}$$

which is true for $\mu_\epsilon < 0.1466$. In practice μ_ϵ must be smaller than 0.1466, because it determines the distance where the gradient of U_a goes to zero through (3.26). This result is shown in Figure 3.9, where the curves refer to different positions i of the local attractor $\boldsymbol{p}'_{ai} = [x'_{ai}\ 0]^T$ characterized by the same R_a, i.e. same γ_a, and by different $\tilde{\alpha}_{ai} = \tilde{\alpha}_a(\sigma, x'_{ai}, \gamma_a)$. At this point it is specified that with the letter and subscript $\boldsymbol{p}_{ai}$ it is indicated the same attractor, i.e. with a fixed R_a, placed in different positions i; on the other hand, the nomenclature $\boldsymbol{p}_{a,i}$ refers to different attractors i, i.e. with different positions and radii. In Figure 3.9, the dotted line is related to $\boldsymbol{p}'_{a3} = [x'_{a3}\ 0]^T$, which is placed at the limit (3.32); the dotted green line intersects the blue one at a point $x' \neq 0$, therefore the global minimum is perturbed. To avoid this, condition (3.20) has been introduced. In the same figure, the point $\boldsymbol{p}'_{a2} = [x'_{a2}\ 0]^T$ is placed at the lower boundary of (3.20); in this case, since the derivative of U_a goes to 0 in $x' = 0$ (curve B with the solid line), the global minimum is undisturbed. Finally, $\boldsymbol{p}'_{a1} = [x'_{a1}\ 0]^T$ is another point which satisfies (3.20).

In Figure 3.10a, a possible two-dimensional representation of the 3 cases discussed in Figure 3.9 is shown. In fact, because U_f and U_a are radial fields, for any $\boldsymbol{p}_{ai}$ one can analyze the potential along the relative x'-axis and compare the results in the same graph.

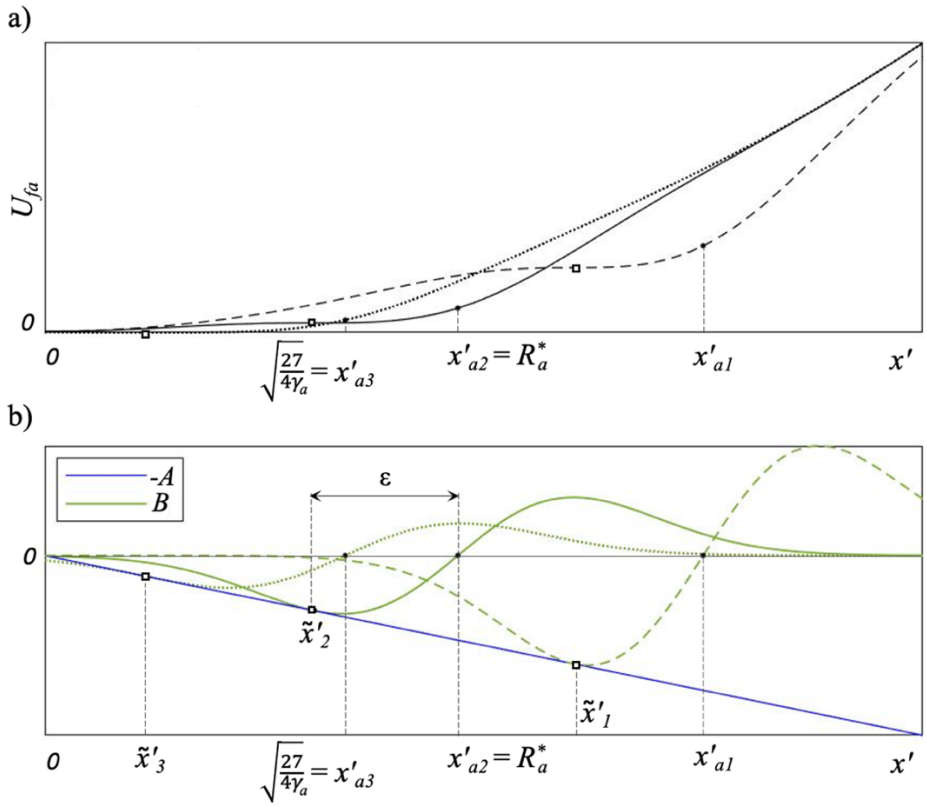


Figure 3.9 Analysis of the influence of x'_a on the solution of (3.28); the different style of the lines refers to 3 positions of the local attractor $\boldsymbol{p}'_{ai}$, with intensity $\tilde{\alpha}_{ai}$. a) Potential function U_{fa} along the x'-axis; the saddle point is indicated with a square. b) Graphical solution; the saddle point is represented by the point of tangency at $\tilde{x}'_i$

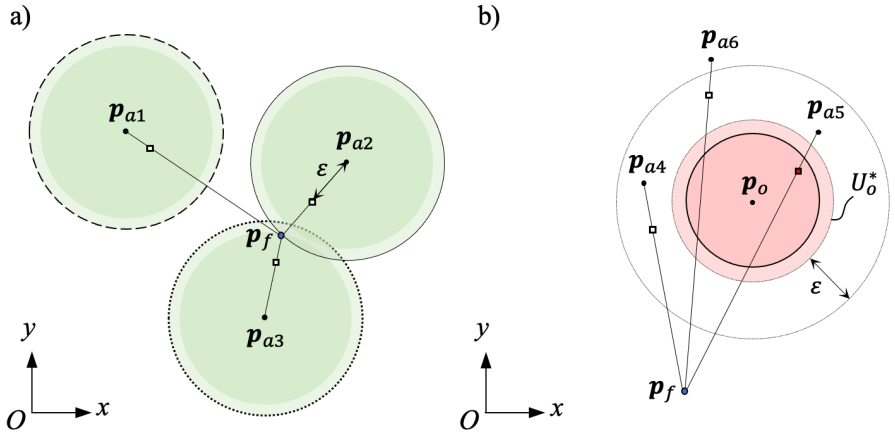


Figure 3.10 Analysis of the saddle point for different positions of the local attractor $\boldsymbol{p}_{ai}$, with intensity $\tilde{\alpha}_{ai}$. For each $\boldsymbol{p}_{ai}$, the relative frame $O' - x'y'$ is not reported, but the direction of the x'-axis is identified with a line segment connecting $\boldsymbol{p}_f$ to $\boldsymbol{p}_{ai}$; thus, the saddle point is identified with a square on the x'-axis. a) The circles identify the active regions; the different style for the contours of circles refer to the 3 cases of Figure 3.9. b) Study of the importance of the saddle point position in presence of an obstacle

The point on the x'-axis where the saddle appears is given by (3.33). Notice that the gradient lines of U_{fa} in the active region U_a^* would converge towards the saddle point. This can be guessed even by observing the one-dimensional plot of U_{fa} in Figure 3.8a. Here, the saddle point is indicated with a square on the dashed line and it is $\tilde{x}' < x'_a$. Therefore, in general, even if U_a is centred in $\boldsymbol{p}_a$, the sum with U_f causes a slight shifting of the resultant attractive effect in the direction of the x'-axis, towards $\tilde{\boldsymbol{p}}' = [\tilde{x}' \ 0]^T$. This aspect can be quantified by calculating the distance $\|\boldsymbol{p}'_a - \tilde{\boldsymbol{p}}'\| = x'_a - \tilde{x}'$, which depends only on x'_a and γ_a. In particular, given γ_a, the distance increases for smaller x'_a, i.e. when the local attractor $\boldsymbol{p}_a$ gets closer to the final position $\boldsymbol{p}_f$. In fact, in Figure 3.9b, where for each $\boldsymbol{p}_{ai}$ the saddle point coordinate is identified with a square in $\tilde{x}'_i$, it is $(x'_{a1} - \tilde{x}'_1) < (x'_{a2} - \tilde{x}'_2) < (x'_{a3} - \tilde{x}'_3)$. Because of (3.20), the maximum distance $\varepsilon = (x'_a - \tilde{x}')_{max}$ is calculated with $x'_a = x'_{a2} = R_a^*$ (see Appendix G for further details). The result is:

$$\begin{aligned} \varepsilon &= \frac{1}{3}\left[-\frac{1}{\gamma_a} W_{-1}\left(-\frac{\mu_\epsilon^2}{e}\right)\right]^{\frac{1}{2}} \left[1 - 2\cos\left(\frac{\vartheta_\varepsilon + 4\pi}{3}\right)\right] \\ \vartheta_\varepsilon &= \cos^{-1}\left(\frac{-27}{2W_{-1}\left(-\frac{\mu_\epsilon^2}{e}\right)} - 1\right) \end{aligned} \tag{3.36}$$

which is the expression of ε as a function of γ_a, given μ_ϵ.

In Figure 3.10b different positions of the local attractor nearby an obstacle are analysed. For each $\boldsymbol{p}_{ai}$ one can find the saddle point by studying U_{fa}, as discussed. This stationary point is aligned with the relative x'-axis and is identified with a square symbol.

Two scenarios can be distinguished. In the first one, the line segment $\overline{\boldsymbol{p}_a\boldsymbol{p}_f}$ does not intersect U_o^* (see point $\boldsymbol{p}_{a4}$); in this case, the stationary point would fall outside U_o^*. The second scenario arises if the line segment $\overline{\boldsymbol{p}_a\boldsymbol{p}_f}$ crosses U_o^* (see the points $\boldsymbol{p}_{a5}$ and $\boldsymbol{p}_{a6}$): here, if the attractor $\boldsymbol{p}_a$ is inside the circle of radius $R_o^* + \varepsilon$ the saddle point of U_{fa} may overlap with the obstacle (see the point $\boldsymbol{p}_{a5}$); thus, when considering the total potential U_t the saddle may not exist and the control on the attraction effect of $\boldsymbol{p}_a$ is lost. In this case, the main issue is that the depression which can still manifest within U_a^* would push the robot towards the obstacle, which is not advisable. On the other hand, if $\overline{\boldsymbol{p}_a\boldsymbol{p}_f}$ crosses U_o^* but the local

attractor is outside the circle of radius $R_o^* + \varepsilon$, this will not happen (e.g. point $\boldsymbol{p}_{a6}$).

From this analysis, the value $\tilde{\varepsilon}$ in condition (3.19) can be calculated for the two different scenarios as:

$$\begin{aligned} \tilde{\varepsilon} &= 0 \qquad \text{if } \overline{\boldsymbol{p}_a \boldsymbol{p}_f} \cap U_o^* = 0 \\ \tilde{\varepsilon} &= \varepsilon \qquad \text{if } \overline{\boldsymbol{p}_a \boldsymbol{p}_f} \cap U_o^* \neq 0 \end{aligned} \tag{3.37}$$

Since for a given μ_ϵ the distance ε depends only on the attractor parameter γ_a, equations (3.36) and (3.37) can be applied to any geometry of the obstacle and, more precisely, for any shape of the function U_o. For example, Figure 3.11 shows different positions of the local attractor near different shapes of U_o^* that satisfies the geometrical constraints (3.19) and (3.20).

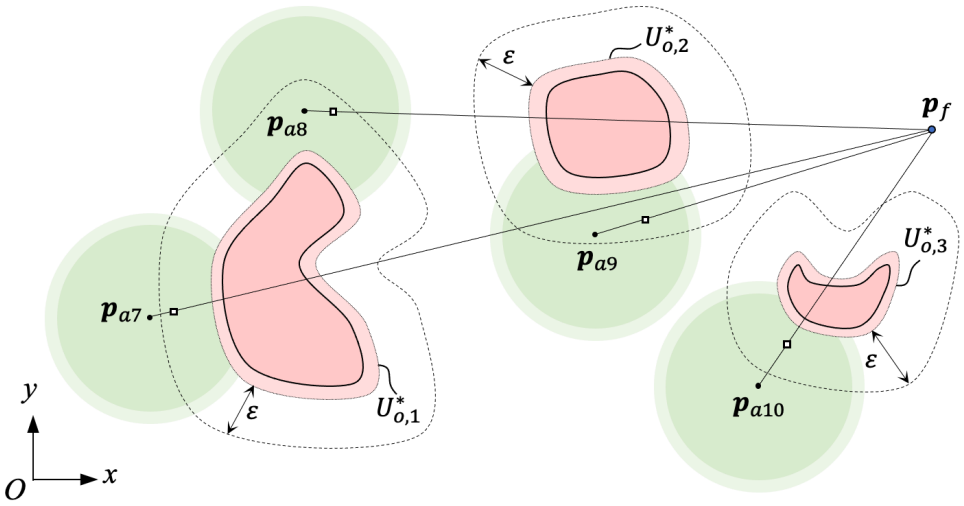


Figure 3.11 Example of a MAP built using different shapes rather than the circle to represent the potential field of the obstacle. For each $\boldsymbol{p}_{ai}$, the saddle point is identified with a square on the line segment connecting $\boldsymbol{p}_f$ to $\boldsymbol{p}_{ai}$ and the geometrical constraints (3.19),(3.20) are satisfied; in fact, the saddle points do not overlap with $U^*_{o,p}$

For the disc obstacle, the following method is proposed to facilitate the design phase. When building the *MAP*, if the attractor $\boldsymbol{p}_a$ is placed so that line segment $\overline{\boldsymbol{p}_a \boldsymbol{p}_f}$ does not intersect U_o^*, it is $\tilde{\varepsilon} = 0$ and condition (3.19) is always verified. On the other hand, if $\overline{\boldsymbol{p}_a \boldsymbol{p}_f}$ crosses U_o^*, equation (3.36) can be used to set the upper bound to R_a. In fact, once the attractor position $\boldsymbol{p}_a$ has been chosen, since $\boldsymbol{p}_o$ is supposed to be identified, one can obtain the admissible ε_{lim} from (3.19). Then, the $\gamma_{a,lim}$ can be exploited from (3.36):

$$\begin{aligned} \varepsilon_{lim} &= \|\boldsymbol{p}_a - \boldsymbol{p}_o\| - R_o^* \\ \gamma_{a,lim} &= \frac{1}{9}\left[-\frac{1}{\varepsilon_{lim}^2} W_{-1}\left(-\frac{\mu_\epsilon^2}{e}\right)\right]\left[1 - 2\cos\left(\frac{\vartheta_\varepsilon + 4\pi}{3}\right)\right]^2 \end{aligned} \tag{3.38}$$

and used in (3.22) to calculate:

$$R_{a,lim} = \left[-\frac{1}{\gamma_{a,lim}} W_{-1}\left(-\frac{\mu_a^2}{e}\right)\right]^{1/2} \tag{3.39}$$

Therefore, if in (3.22) is selected $R_a < R_{a,lim}$, condition (3.19) is verified.

Finally, formula (3.34) is discussed. If $\boldsymbol{p}_a$ is close to the global attractor, the $\tilde{\alpha}_a$ decreases and a smaller depression can be obtained. On the other hand, $\tilde{\alpha}_a$ is proportional to σ. In other words, by increasing σ one can augment the attraction towards the desired position and, at the same time, the effect of the local attractor. This suggest that one can select higher values for σ when $\boldsymbol{p}_a$ is in the proximity of $\boldsymbol{p}_f$ and the effect of the local attractor is poor.

3.4.3 Dynamic obstacle and attractor

In the previous section the obstacle is treated as a steady object in the xy-plane. As a consequence, the local attractor $\boldsymbol{p}_a$ can be fixed in a desired position to guide the collision avoidance along a preferred region nearby the obstacle. However, in many applications the object can move, thus it is desirable that $\boldsymbol{p}_a$ moves as well to keep the collision avoidance conditioning around the obstacle. In this scenario, the attractor can be fixed to the obstacle frame in order to reflect its motion. The result is that $\boldsymbol{p}_a$ can potentially move relatively to the goal $\boldsymbol{p}_f$.

Consider the following practical example. In Figure 3.12a, the object whose position and orientation are defined by the reference frame $O'' - x''y''$ with $O'' \equiv \boldsymbol{p}_o$, has been identified at time t_0 and its shape has been approximated to a sphere with centre in $\boldsymbol{p}_o(t_0)$ and radius R_o. The repulsive potential can be modelled with the exponential function U_o characterized by β_o, γ_o and λ_o according to (3.17) and (3.18). These parameters are constant because they depend only on the obstacle geometry. The final goal $\boldsymbol{p}_f$ is a fixed point in the plane, i.e. in the world frame, described by the attractive quadratic function U_f with the intensity σ.

For some reason, it is desired that the robot avoids the object passing through the region defined by the circle with centre in $\hat{\boldsymbol{p}}_a'' = [0\ y_a''\ 1]^T = \boldsymbol{\mathcal{A}}''(t)\hat{\boldsymbol{p}}_a(t)$ and

radius $R_a(t_0)$, where $\boldsymbol{\mathcal{A}}''(t)$ is the (3×3) matrix which transforms the vector $\overline{\boldsymbol{p}}_a(t)$ from the world frame to the obstacle frame. For consistency, the vectors $\hat{\boldsymbol{p}}_a''$ and $\hat{\boldsymbol{p}}_a$ have been introduced to indicate respectively $\boldsymbol{p}_a''$ and $\boldsymbol{p}_a$ in homogeneous coordinates.

To facilitate reading, hereafter the subscript t_h indicates the physical quantity at the times instant $h = 0,1,2$. For instance, it is $R_{a,t_0} = R_a(t_0)$. Otherwise, the subscript t indicates the value at the generic time. For example, it is $\boldsymbol{p}_{a,t} = \boldsymbol{p}_a(t)$.

The attractive potential $U_{a,t}$ is modelled with the exponential function. At time t_0, the parameters $\gamma_{a,t_0} = \gamma_a(R_{a,t_0})$ and $\alpha_{a,t_0} < \tilde{\alpha}_a(\sigma, x'_{a,t_0}, \gamma_{a,t_0})$, with $\|\boldsymbol{p}_{a,t_0} - \boldsymbol{p}_f\| = x'_{a,t_0}$, can be calculated according to (3.22) and (3.34). Moreover, suppose that assumptions (3.19) and (3.20) are verified at t_0 (Figure 3.12a).

Suppose that for $t_0 < t \leq t_1$ the object $\boldsymbol{p}_{o,t}$ moves in the world frame with a generic trajectory (Figure 3.12b). The attractor $\boldsymbol{p}_{a,t}$ also moves in the plane accordingly to $\boldsymbol{\mathcal{A}}_t''$. Thus, the distance $\|\boldsymbol{p}_{a,t} - \boldsymbol{p}_f\| = x'_{a,t}$ may vary. The value of $\gamma_{a,t}$ is considered constant and equal to γ_{a,t_0} since it is defined by the attractive range R_{a,t_0}. On the other hand, for each instant, one can adjust the intensity $\alpha_{a,t}$ in order to have the maximum attractive effect near the point $\boldsymbol{p}_{a,t}$. In fact, by measuring $x'_{a,t}$, it is straightforward to calculate the $\tilde{\alpha}_a(\sigma, x'_{a,t}, \gamma_{a,t_0})$ with (3.34). The limit case related to time t_1 is depicted in Figure 3.12c; here it is $x'_{a,t_1} = R^*_{a,t_0}$.

Then, at time $t_1 < t \leq t_2$ the $\boldsymbol{p}_{a,t}$ gets close to $\boldsymbol{p}_f$ so that $x'_{a,t} < R^*_{a,t_0}$ (Figure 3.12d). In this particular case, one can shrink the active range of $U_{a,t}$ by dynamically adjusting $\gamma_{a,t}$ with the value which satisfy (3.20). This is applicable because of the choice of defining U^*_a through (3.23): since γ_a does not depend on $\tilde{\alpha}_a$, one can calculate $\gamma_{a,t} = \gamma_a(x'_{a,t})$ by considering $R^*_a = x'_{a,t}$ in (3.23); because $\gamma_{a,t}$ is not constant in $t_1 < t < t_2$, by substituting the expression of $\gamma_a(x'_{a,t})$ in (3.34), one can also find $\tilde{\alpha}_a(\sigma, x'_{a,t}, \gamma_{a,t}) = \tilde{\alpha}_a(\sigma, x'_{a,t})$.

Another aspect to consider is that, for $t_1 < t \leq t_2$ by varying $\gamma_{a,t}$ also the distance $\varepsilon(\gamma_{a,t})$ changes. But, since here $\gamma_{a,t} > \gamma_{a,t_0}$ (to shrink $U^*_{a,t}$, the parameter $\gamma_{a,t}$ must grow), if condition (3.19) is verified for $t_0 < t \leq t_1$, it remains true in $t_1 < t \leq t_2$. In fact, by increasing $\gamma_{a,t}$, from equation (3.36) $\varepsilon(\gamma_{a,t}) < \varepsilon(\gamma_{a,t_0})$.

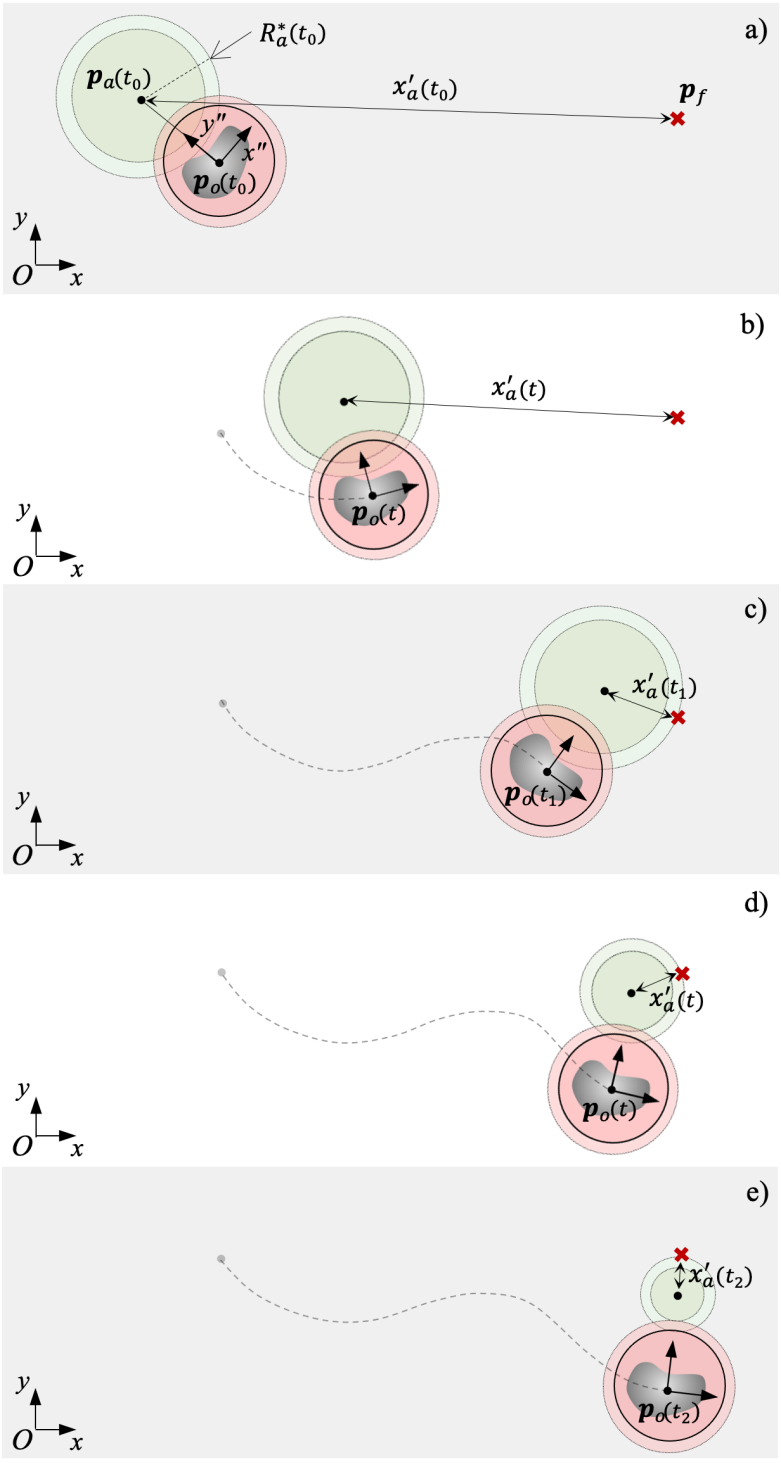


Figure 3.12 Case study with dynamic obstacle and local attractor. The dashed line represents the obstacle path. The object moves with a generic motion from $\boldsymbol{p}_o(t_0)$ to $\boldsymbol{p}_o(t_2)$; the local attractor is fixed to the obstacle frame and moves with respect to the world frame. The figures from a) to e) represents the scene at significant times

The important result is that, given σ, regardless of the obstacle trajectory in $t_0 < t < t_2$, the distance $x'_{a,t}$ is the only information required to regulate the parameters $\gamma_{a,t}$ and $\alpha_{a,t}$. To visualize how γ_a and $\tilde{\alpha}_a$ vary depending on x'_a, a numerical example is shown in Figure 3.13 (this will be discussed more in details in the result chapter). Notice that if one wants a higher intensity α_a, a solution is to increase σ.

Finally, one wonders if the strategy of squeezing U_a^* is just a compromise to satisfy condition (3.20). Actually, the proposed method is also justified by the fact that, in practice, when x'_a becomes small, the role of the local attractor starts confusing with the global one and vice-versa (Figure 3.12e). In fact, at the limit case $\boldsymbol{p}_a = \boldsymbol{p}_f$, it is reasonable to set α_a, i.e. U_a, to zero. In other words, when $\boldsymbol{p}_a$ gets close to $\boldsymbol{p}_f$, one can leave to $\boldsymbol{p}_f$ both the roles of global and local attractor. It means that the robot would navigate towards $\boldsymbol{p}_f$, but also avoiding the obstacle on the side where $\boldsymbol{p}_a$ has been placed.

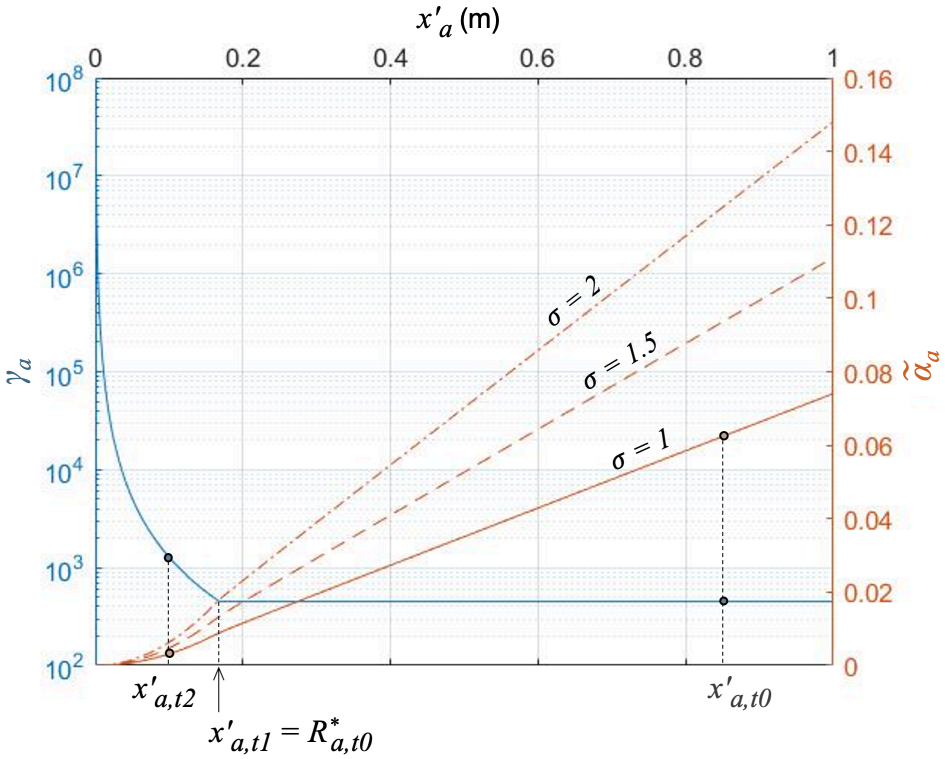


Figure 3.13 Numerical example of the case described in Figure 3.12. The example refers to a local attractor with the initial values $R_{a,t_0} = 0.1304\ m$, $\gamma_{a,t_0} = 449.5$, $R^*_{a,t_0} = 0.1685\ m$. The graph shows also the $\tilde{\alpha}_a$ curve for different values of σ; the γ_a curve remains the same, since it does not depend on σ

3.4.4 Three-dimensional trajectories

In the previous sections, the MAP has been discussed in two dimensions, considering a single obstacle and one local attractor. In this section, the method is extended to the three-dimensional case with multiple obstacles $\boldsymbol{p}_{o,p}$ and local attractors $\boldsymbol{p}_{a,i}$.

By considering the robot as a point $\boldsymbol{p}_r$ moving in the three-dimensional space, the generalization is straightforward if it is observed that it is always possible to identify a plane which contains $\boldsymbol{p}_r$, $\boldsymbol{p}_f$ and $\boldsymbol{p}_a$ Figure 3.14. In this plane, since U_f and U_a are radial fields, all the results from the previous sections are still valid. Moreover, after choosing σ and R_a, the $\tilde{\alpha}_a$ is a function of the only distance between the final point and the attractor $\|\boldsymbol{p}_a - \boldsymbol{p}_f\| = x'_a$ and can be obtained from the same equation (3.34). Notice that the conditions (3.19) and (3.20) remain since they are geometrical constraints that can be easily transposed to three dimensions. In particular, the presence of the obstacle is handled by (3.19), which says that the attractor $\boldsymbol{p}_a$ must be at a distance greater than $\tilde{\varepsilon}$ from U_o^*. Thus, if one chooses a more complex U_o and can compute the distance from the obstacle, the shape and the dimension of the object becomes secondary.

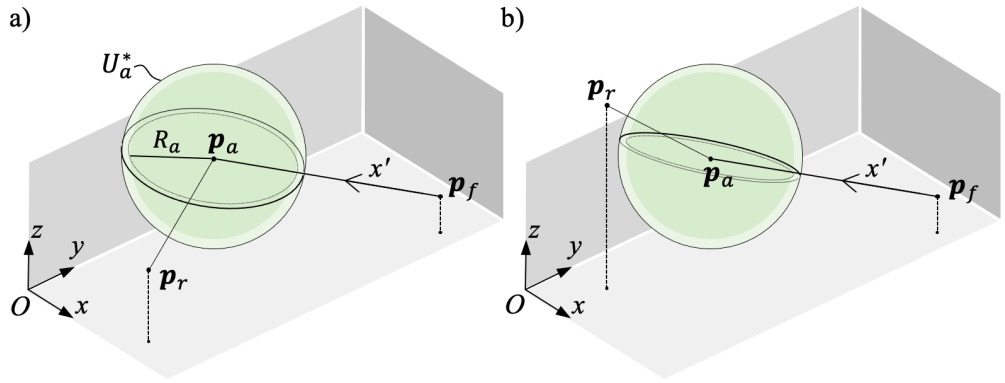


Figure 3.14 Three-dimensional analysis of the spherical active region of a generic local attractor. The circles with radius R_a and R_a^* identify the section of the sphere in the plane containing $\boldsymbol{p}_r$, $\boldsymbol{p}_f$ and $\boldsymbol{p}_a$. The figures a) and b) considers two different positions of the robot

Consider now the case in which the robot navigates towards the goal $\boldsymbol{p}_f$ dealing with m obstacles $\boldsymbol{p}_{o,p}$ with $p = 1,2,\dots m$. Suppose that n attractors $\boldsymbol{p}_{a,i}$, with $i = 1,2,\dots n,$ are displaced around the obstacles in order to guide the robot along a preferred path Figure 3.15. The method can still be applied with the following assumptions:

$$\left\|\boldsymbol{p}_{a,i}-\boldsymbol{p}_{o,p}\right\|>R_{o,p}^{*}+\tilde{\varepsilon}_{i} \tag{3.40}$$

$$\left\|\boldsymbol{p}_{a,i}-\boldsymbol{p}_{f}\right\|\geq R_{a,i}^{*} \tag{3.41}$$

$$\left\|\boldsymbol{p}_{a,i}-\boldsymbol{p}_{a,j}\right\|\geq R_{a,i}^{*}+R_{a,j}^{*} \tag{3.42}$$

The constraints (3.40),(3.41) are the equivalent of (3.19),(3.20) extended to each attractor $\boldsymbol{p}_{a,i}$ and each obstacle $\boldsymbol{p}_{o,p}$. Condition (3.42) prevents the active regions $U_{a,i}^{*}$ from overlapping, so that the local minima problems are decoupled and each single attractor can be handled as discussed in the previous sections.

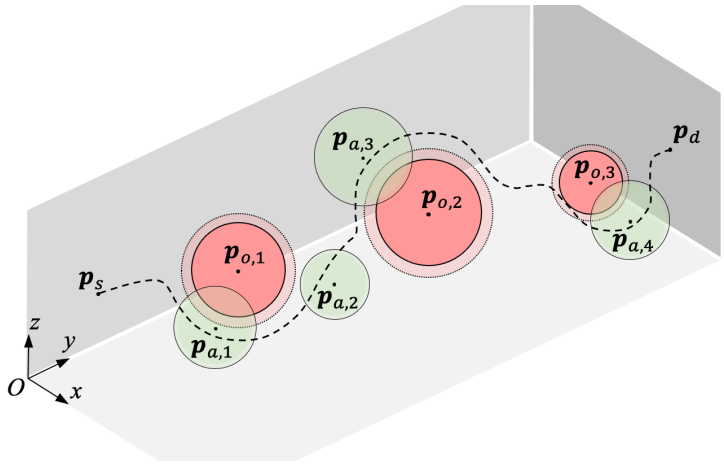


Figure 3.15 Example of a three-dimensional MAP with $m=3$ obstacles and $n=4$ local attractors. For clarity, the local attractors are represented only by the spheres with $R_{a,i}^{*}$. The dashed curve identifies a possible robot path

3.4.5 Summary of the MAP design steps

In this section, the steps to design the MAP are summarized in a schematic form to facilitate the practice. For clarity, a two-dimensional graphics is utilized to show the various steps, but the algorithms here proposed are intended for the most general case in three-dimensions. Since the case with multiple local attractors and obstacles can be solved with the same formulas by modelling each element separately, the analysis is related to generic $\boldsymbol{p}_{o,p}$ and $\boldsymbol{p}_{a,i}$. In Table 3.1, the algorithm to design the MAP is reported; the numbers in the left column refers to the design steps illustrated in Figure 3.16. If the MAP is static, i.e. if the obstacles and the local attractors do not move, one can use the steps 1-7; if the MAP is dynamic, i.e. if the elements move, one can follow the steps 1-9. In fact, at the initial time $t=t_0$, the rules to design the MAP are the same of the static case (steps 1-7); in addition, the steps 8-9 take into account the variability of $x'_{a,i}(t)$.

Table 3.1 MAP algorithm

1	>> Define $\boldsymbol{p}_s$ and $\boldsymbol{p}_f$ >> choose σ
2	>> localize the obstacle >> identify $\boldsymbol{p}_{o,p}$, $R_{o,p}$
3	>> choose $\lambda_{o,p}$, s_ϵ and $\beta_{o,p}$ >> use (3.17) to calculate $\gamma_{o,p}$ and find $R^*_{o,p}$ with (3.18)
4	>> place $\boldsymbol{p}_{a,i}$ outside $U^*_{o,p}$
5	>> *if* $\overline{\boldsymbol{p}_{a,i}\boldsymbol{p}_f} \cap U^*_{o,p} \neq 0$ >> calculate the $R_{a,lim}$ using (3.38)(3.39)
6	>> choose $R_{a,i} < R_{a,lim}$
5'	>> *elseif* $\overline{\boldsymbol{p}_{a,i}\boldsymbol{p}_f} \cap U^*_{o,p} = 0$
6'	>> choose $R_{a,i}$ >> *end*
7	>> choose $\mu_{a,i}$ and $\mu_\epsilon < 0.1466$ >> use (3.22) to calculate $\gamma_{a,i}$ and find $R^*_{a,i}$ with (3.23) >> calculate $x'_{a,i}$ and verify (3.41)(3.42) >> calculate $\tilde{\alpha}_{a,i}$ with (3.34) and choose $\alpha_{a,i} < \tilde{\alpha}_{a,i}$
8	>> *while* (3.41)(3.42) >> *if* $x'_{a,i}(t) \geq R^*_{a,i}(t_0)$ >> $\gamma_{a,i}(t) = \gamma_{a,i}(t_0)$ >> find $\tilde{\alpha}_{a,i}(t)$ with (3.34) and choose $\alpha_{a,i}(t) < \tilde{\alpha}_{a,i}(t)$
9	>> *elseif* $x'_{a,i}(t) < R^*_{a,i}(t_0)$ >> find $\gamma_{a,i}(t)$ by placing $R^*_{a,i}(t) = x'_{a,i}(t)$ in (3.23) >> find $\tilde{\alpha}_{a,i}(t)$ with (3.34) and choose $\alpha_{a,i}(t) < \tilde{\alpha}_{a,i}(t)$ >> *end* >> *end.*

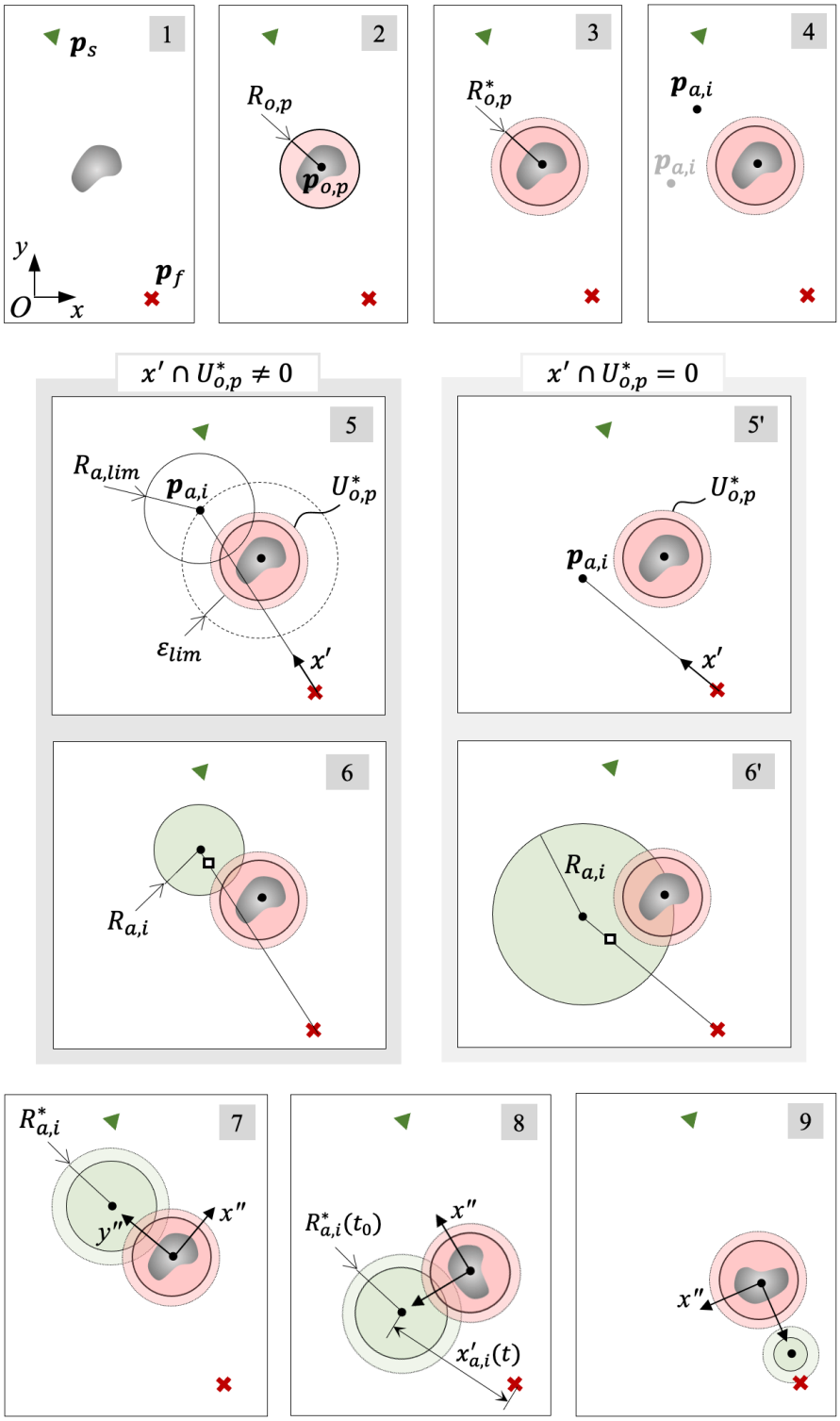


Figure 3.16 Design steps of the MAP algorithm

3.4.6 Trajectory generation and motion planning

Consider a 6dof robotic manipulator. Assume the task is given to the robot in terms of end-effector position and orientation. The goal is to reach the $\boldsymbol{p}_f$ by avoiding the obstacles, still controlling the orientation.

Concerning the position problem, the gradient tracking method is chosen to take full advantage from the potential field generated with the MAP. This simple and effective technique produces an exact tracking of the gradient lines and can be applied to any smooth artificial vector field ([65]). It consists in regarding the velocity vector rather than the acceleration vector as the variable under control. To move the end-effector towards the desired position $\boldsymbol{p}_f$ the linear velocity can be chosen as:

$$\dot{\boldsymbol{p}}_e = -v_e \frac{\nabla U_{t,MAP}}{\left\| \nabla U_{t,MAP} \right\|} \tag{3.43}$$

where $\nabla U_{t,MAP}$ is the gradient of the MAP and v_e is the desired scalar velocity. If $v_e = 0$ at the starting and final positions, a suitable choice for the linear velocity magnitude is ([65]):

$$v_e = \min \left(a_{max} t, v_{max}, (2 a_{max} e_p(t))^{\frac{1}{2}} \right) \tag{3.44}$$

where a_{max} is the maximum acceleration, v_{max} the maximum velocity and $e_p(t) = \left\| \boldsymbol{p}_f - \boldsymbol{p}_e(t) \right\|$ is the magnitude of the position error.

The orientation problem can be handled without requiring the MAP. Suppose that the actual and final orientation of the robot frame with respect to the world frame are identified by the rotation matrices $\boldsymbol{\mathcal{R}}_e$ and $\boldsymbol{\mathcal{R}}_f$ respectively. Let $\boldsymbol{\mathcal{Q}}_e(t) = \{\eta_e, \boldsymbol{\sigma}_e\}$ and $\boldsymbol{\mathcal{Q}}_f(t) = \{\eta_f, \boldsymbol{\sigma}_f\}$ represent the quaternions associated with the rotation matrices $\boldsymbol{\mathcal{R}}_e$ and $\boldsymbol{\mathcal{R}}_f$. The orientation error $\boldsymbol{e}_o = \boldsymbol{e}_o(t)$ between the actual and the final orientation is calculated using unit quaternions ([13]):

$$\begin{aligned} \boldsymbol{e}_o &= \eta_e \boldsymbol{\sigma}_f - \eta_f \boldsymbol{\sigma}_e - \boldsymbol{\mathcal{S}}(\boldsymbol{\sigma}_f)\boldsymbol{\sigma}_e \\ \boldsymbol{\mathcal{S}}(\boldsymbol{\sigma}_f) &= \begin{bmatrix} 0 & -\boldsymbol{\sigma}_f(3) & \boldsymbol{\sigma}_f(2) \\ \boldsymbol{\sigma}_f(3) & 0 & -\boldsymbol{\sigma}_f(1) \\ -\boldsymbol{\sigma}_f(2) & \boldsymbol{\sigma}_f(1) & 0 \end{bmatrix} \end{aligned} \tag{3.45}$$

where $\mathcal{S}(\circ)$ is the skew-symmetric operator. Then, the angular velocity of the end-effector is chosen as:

$$\dot{\boldsymbol{\phi}}_e = \boldsymbol{\omega}_e = \omega_e \frac{\boldsymbol{e}_o}{\|\boldsymbol{e}_o\|} \tag{3.46}$$

If ω_{max} is the maximum angular velocity and $\boldsymbol{\psi}_{max}$ is the maximum angular acceleration, a suitable value for the magnitude can be:

$$\omega_e = \min\left(\boldsymbol{\psi}_{max} t, \omega_{max}, (2\boldsymbol{\psi}_{max}\|\boldsymbol{e}_o(t)\|)^{\frac{1}{2}}\right) \tag{3.47}$$

The inverse kinematics algorithm based on the inverted Jacobian is utilized [13] and the resulting control law for the manipulator can be written as:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\dot{\boldsymbol{\chi}}_e = J^{-1}\begin{bmatrix}\dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e\end{bmatrix} \tag{3.48}$$

where $\dot{\boldsymbol{q}}$ is the (6 x 1) vector of the joint velocities and $\boldsymbol{J}$ represents the geometric Jacobian. The practical advantage of the control law (3.48) is that the magnitude of linear velocity of the robot is not dependent on the intensity of the gradient. In the specific case of the MAP, if α_a is reasonably close to $\tilde{\alpha}_a$, the gradient magnitude in the region nearby $\tilde{\boldsymbol{p}}'$ tends to zero. But the solution proposed by [65] applied to this case allows to pass through U_a^* without decelerating.

3.5 Discussion

The two algorithms presented in this chapter exploit the repulsive action in different manners.

The *robot links - human* collision avoidance algorithm use repulsive velocities on control points of the manipulator to generate an evasive motion across the entire robot body, depending on the minimum distances with the obstacle. The method is synthesized in a control law whose total effect is the sum of the velocities related to the planned trajectory with the ones generated locally by the repulsive action. The method has been proven in [12], [79], [80] and stands on similar studies that have already shown the effectiveness of repulsive velocities [73]–[75].

The MAP is a novel method that expands the capabilities of collision avoiance by conditioning the collision-free path through a potential field with multiple attractors. The optimal coexistence of multiple attractors has not yet been addressed by the literature. To reduce the complexity of this problem, the exact

solution has been presented by considering only the robot end-effector. The MAP, combined with the sliding mode [65], would drive the end-effector at the final point with a collision-free path that is affected by the local attractors.

The two algorithms use different control laws. The control law (3.10) not necessarily generates a collision free motion. For instance, the actual end-effector trajectory depends on the magnitude of the velocity related to the task and the intensity of the repulsive effects of each link. Since the task velocity is given, the parameters $v_{rep,max}$, α and ρ have to be tuned experimentally to guarantee a collision free trajectory. On the contrary, with (3.43) and (3.48) the end-effector directly tracks the collision free path, with a velocity law defined by v_e and ω_e.

In the results chapter, the strengths and the drawback of the MAP will be pointed analysing different tests. It is worth to mention that the most important limitation in comparison with the *robot links - human* collision avoidance algorithm is that with the current MAP theory the manipulator cannot avoid obstacles with each link. However, because the output of both the algorithms is a set of joint velocities $\dot{\boldsymbol{q}}_{set}$ for the manipulator, the MAP can potentially be extended by adding repulsive velocities to (3.48), with the same principle of (3.10). This will be addressed by future works.

At the end, as for human tracking algorithms, collision avoidance algorithms presented in this chapter can be grouped in a software block with inputs and outputs. Figure 3.17 shows how the human tracking block can be integrated with the collision avoidance block by means of distance calculation. More details will be given in the results chapter.

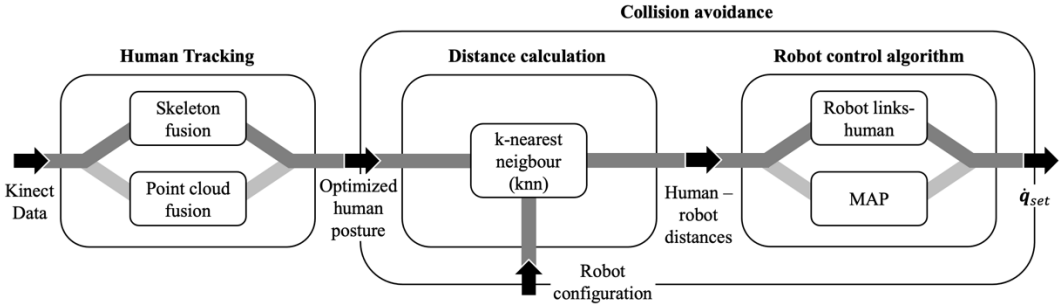


Figure 3.17 Block scheme of the integration between human tracking and collision avoidance algorithms

Chapter 4

Hand-over

Objects exchange between human and robot goes under the name of human-robot hand-over. In collaborative robotics, the hand-over is the action that best represents responsive collaboration as it requires human robot coordination in space and time [4].

From the technical point of view, the hand-over involves different challenges depending on the representation of the OTP (from *object transfer point*). If the OTP is a fixed point in the robot workspace, it is supposed that human and robot a priori know the place to meet for the object exchange. Instead, a reactive hand-over tracks the human hand as the OTP for a more natural gesture.

In contrast to collision avoidance, where the human is seen as a disturbance by the robot and vice-versa, the reactive hand-over consider the human hand as the robot target. For this reason, developing a reactive hand-over involve in robot trajectory and motion planning with dynamic target. However, more specific aspects like the robot velocity profile [84], the human-robot relative position at the OTP and the waiting time for the operator [85], can be fundamental. In particular, it will be seen that reducing the waiting time for the operator results in a more natural hand-over, in addition to an improved task time. For this reason, predicting the position of the human hand represents an important factor.

This chapter presents a reactive hand-over algorithm and a control strategy that can deal with the mentioned aspects. The hand-over algorithm specifically addresses *approach* and *reach* phases of a canonical hand-over task [86], which are the actions that involve the approach of the robot to the human hand until they

are close enough. In other words, the algorithm represents a solution to the trajectory and motion planning problems. In general, the hand-over also involves the *transfer* and the *retract* phases, where giver and receiver complete the object transfer and move back. The rules on how and when the hand-over algorithm is triggered within each phase are addressed by a dedicated hand-over strategy, which is presented at the end of the chapter.

Before describing the theory behind the algorithm, the state of the art is analysed. Concerning the existing robot trajectory and motion planning techniques in presence of a dynamic target, they are not discussed in a dedicated section since they can be basically reconducted to local methods reviewed in the previous chapter.

4.1 State of the art

The earliest studies on human-robot hand-over have been conducted in the nineties by applying human arm mathematical models [87] and human-human hand-over patterns [85] to simplified robot models. This has permitted to investigate human-robot preferences and to design the first hand-over control strategies [88]. In particular, [85] observed that bell-shaped velocity profile of the robot while approaching the human hand, as well as natural handing point, i.e. the OTP, positively affect human emotions during the task. The results of [85] are obtained using a one-dimensional robot. In [88], a fuzzy logic control is applied to a mobile manipulator with 3dof to adjust the OTP depending on the human arm motion. Even if [88] uses a two-dimensional problem to validate the control strategy, this work is still one of the first successful attempt of human-robot hand-over.

In the last two decades, different aspects of human-robot hand-over have been studied: psychological effects on the human worker, ergonomics, grasping and motion-planning.

By learning from natural hand-over, the psychological aspects due to the robot velocity addressed by [85] have been further investigated in [84], [89], [90]. The work [84] has confirmed that minimum jerk profile in spatial coordinates performs better than conventional trapezoidal velocity profiles in joint space for a robot manipulator. This result has been improved by the same authors in [89], where a trajectory generator that resembles human motion is obtained with a decoupled minimum jerk velocity profile. The study [90] investigated the robot

role as a giver and observed that predictability of robot actions makes the human more comfortable, increasing the chances of a fluent hand-over.

Ergonomics is mainly related to position and the orientation at which the object is presented and how the robot should be configured around the object. The studies [91], [92] observed that not only the OTP, but also the robot orientation should be adjusted to help object transferring. In [91] the robot delivers different objects through configurations learned from human preferences. A different method is described in [93], where feasible ergonomic configurations of the human arm are selected through cost learning problem.

Other works focused on the study of the grasping forces. For instance, [94] observed that humans apply pulling forces on the robot tool in specific directions when receiving an object. Thus, pulling forces are monitored in [94] to identify the appropriate moment for the robot to release the object. Hand-over interaction forces have been also registered in [95], where the results show that givers trigger their release on a relative force change.

The cited literature investigated specific features of human robot preferences. Results are obtained through experimental tests specifically designed to observe the features under study. Some of them used predetermined ([84], [89]–[91]) or fixed ([93]–[95]) hand-over position for the robot.

The most insightful contributions describe real-world hand-over applications requiring motion planning at a level that allows the robot to adapt to human movement ([96]–[102]). A reactive bidirectional path planner for hand-over actions is described in [96], [97]. The work [96] considers wearable sensors combined with active control to recognize human intention and optimize the hand-over task. In [97] AR (from *augmented reality*) tags are used to increase the vision system robustness. However, the use of wearable device limit the applicability of [96], [97] in an industrial scenario. In [98]–[102] a Kinect camera is used to track the human or the object to be delivered. In particular, [98] consider a domestic environment where a mobile manipulator receives objects from a human. In [98] the pose of human's arm is obtained from skeleton tracking, while a 3D perception module based on point cloud is capable of object detection; the robot recognizes human's intention from the pose of his arm and a reactive control strategy drives the robot towards the object. However, the perception system runs at $5\ fps$, which is limiting. In [99] a hybrid approach similar to [98] gives human and object position data using skeleton and point cloud. Depending on the object type and position, the robot hand-over configuration is chosen to orient the handle of the object towards the human torso.

However, the robot best trajectory is planned offline. In [100] a collaborative robot is tested as robotic assistant to a car mechanic; a dynamic movement primitives-based control is implemented to characterize the hand-over by means of spatial and time indexes. A major limitation of the method described in [100] is that hand tracking is performed through color image segmentation and requires the use of a glove to increase color contrast for the desired tracking precision. The work [101] presents an adaptive control strategy for a 6dof manipulator. In [101] the hand-over task is categorized into six different states that identify the activity for the giver and the receiver. Depending on human's activity, the best robot motion is determined based on a probability distribution obtained from data on human-human interactions. Even if the algorithm in [101] allows the robot to wait or slow-down to adapt to user's task demand, the discrete state-based control is not suitable for a reactive hand-over. The study [102] presents a control algorithm that permits reactive bidirectional hand-over operations. The experimental results of [102] shows that the waiting time (the time between the worker stops to move the hand and the hand-over task is per- formed) is long, reducing hand-over fluency and task efficiency

To reduce the waiting time, in some studies the robot anticipates human intentions using predictive models [101], [103]–[106]. In [101] the user's current state, e.g. *reaching* (the robot) or *retracting* (the arm), is predicted by observing skeleton joints and finding the activity that best matches human-human interactions data based on temporal and spatial features. A more precise model is described in [103], where the target of the human reaching phase is also estimated. However, [103] makes prediction among a predefined set of targets. In [104] the OTP is predicted through a joint torque cost function, which is limited by the assumptions of planar hand-over motion and 2dof human arm model. An OTP estimator based on a model trained with human-robot handover demonstrations is used in [105], but only the case of human giver is considered. The work [106] describes a method to predict the human hand position under the assumption of linear hand-over motion with bell-shaped profile.

4.2 Contribution to the current state of knowledge

The state of the art shows that most of the applications that provide human-aware control strategies considered the case of robot-to-human hand-over ([97], [99], [101], [103], [104], [106]). Only few of them studied human-to-robot ([98], [105]) and bidirectional ([96], [100], [102]) hand-over. Furthermore, the mentioned bidirectional applications do not consider prediction of the human motion.

The work [12] introduced a novel algorithm based on [39] and [107] that permits to have fast and reactive bidirectional hand-over considering not only the human's hand position, but also his forearm orientation.

This thesis presents an improved version of the algorithm in [12] and [107] which takes into account human motion prediction. The reader is referred to the latest publication [108]. The features that distinguish the improved algorithm from the state of the art can be summarized as following:

- it is reactive, i.e. it responds in real-time and adapts robot motion to human's intention;
- it is bidirectional, i.e. both robot and human can be giver or receiver;
- it works with 3D vision systems which deliver skeleton tracking, without requiring wearable sensors;
- it makes the hand-over comfortable by orienting the robot TCP according to human's forearm;
- it reduces the waiting time by anticipating human motion.

To the author knowledge, no one of the previous studies addressed all the listed features in a unique framework.

4.3 Improved hand-over algorithm

The hand-over algorithm implemented in this work is the result of the combination of several modules. A schematic representation of the algorithm is shown in Figure 4.1. Human position data is retrieved in terms of skeleton from the sensor fusion module. The prediction module estimates the position of the hand at future samples. This information, together with a convenient representation of the target in terms of virtual hand, is exploited by the trajectory generation and motion planning module, which uses local methods to drive the robot towards the dynamic target. The output of the algorithm is the control vector for the manipulator as joint velocities. In the following sections the prediction module as well as the trajectory and motion planning module are described.

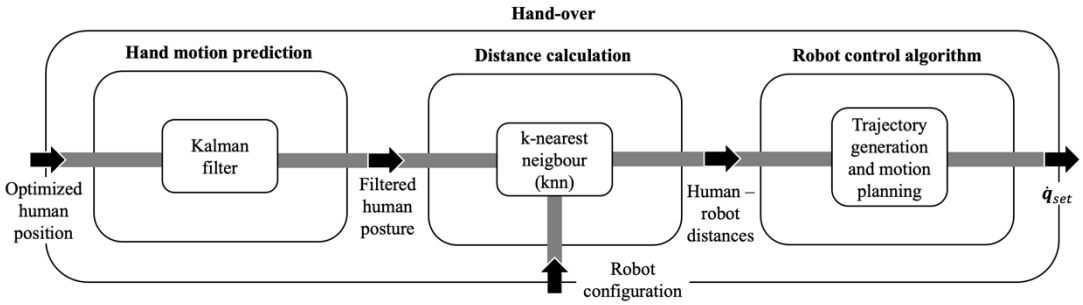


Figure 4.1. Block scheme of the hand-over algorithm

4.3.1 Human-hand position prediction

The optimized skeleton is quite robust to occlusions, but still suffers from jitters, which may affect the hand-over. In order to stabilize the robot target, a Kalman filter is applied to the optimal human hand joint, since it is proved to be an effective technique for a low-latency data smoothing in skeleton tracking [109]. Moreover, by modeling the human hand motion as a Wiener process [110] and by propagating the prediction stage of the Kalman filter, a prediction scheme is introduced to optimize the task time. In fact, in the previous works [39], [107], the limited velocity of the robot due to collaborative constraints resulted in a waiting time for the operator: the human reaches the OTP and waits for the robot before being able to hand-over the piece. In this work, the predicted position of the hand is estimated so that the robot, by slightly anticipating the target, is better synchronized with the human.

The assumption of linearity leads to the following discrete state-space representation for the dynamic target and the sensor:

$$\boldsymbol{x}_k = \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_k \tag{4.1}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{4.2}$$

Equation (4.1) represents the state equations and relates the state vector $\boldsymbol{x}_k$, at the discrete-time index k, to the state $\boldsymbol{x}_{k-1}$, at time index $k-1$, through the state transition matrix $\boldsymbol{F}_k$. The process noise vector $\boldsymbol{w}_k$ represents the fact that the dynamic of the target is not perfectly known and takes into account also the inaccuracy of the discrete representation of the continuous-time process. Measurement equations are collected in (4.2) and define the measurement vector $\boldsymbol{z}_k$ as a function of the state $\boldsymbol{x}_k$ by means of the measurement matrix $\boldsymbol{H}_k$. The measurement noise $\boldsymbol{v}_k$ is the uncertainty due to the imperfection of the sensor. It is

assumed that $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are white noises with zero mean and covariance matrix $\boldsymbol{Q} = E\{\boldsymbol{w}\boldsymbol{w}^T\}$ and $\boldsymbol{R} = E\{\boldsymbol{v}\boldsymbol{v}^T\}$, respectively.

Given Equations (4.1) and (4.2), the Kalman filter provides an unbiased and optimal estimate of the state-vector in two steps: the prediction stage and the updating stage. The prediction stage gives a priori information about the probability distribution of the state, before that any measure of the system is acquired:

$$\overline{\boldsymbol{x}}_k^- = \boldsymbol{F}_k \overline{\boldsymbol{x}}_{k-1}^+ \tag{4.3}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}_k \boldsymbol{P}_{k-1}^+ \boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{4.4}$$

where $\overline{\boldsymbol{x}}_k^-$ is the a priori estimate of the state and $\boldsymbol{P}_k^-$ denotes the state covariance matrix. Once the measurements are available, the update step is computed as follows:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \tag{4.5}$$

$$\overline{\boldsymbol{x}}_k^+ = \overline{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k \overline{\boldsymbol{x}}_k^-) \tag{4.6}$$

$$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{P}_k^- \tag{4.7}$$

where $\boldsymbol{K}_k$ is the Kalman gain while $\overline{\boldsymbol{x}}_k^+$ and $\boldsymbol{P}_k^+$ are the updated state vector and state covariance matrix. The superscript $+$ and $-$ in equations (4.5),(4.6),(4.7) indicate if the estimated statistics have been updated or not.

For the purpose of this work, the observed system is the human hand, whose position in the Cartesian space is indicated with $\boldsymbol{s}$. It is assumed that the hand acceleration is not exactly constant, and its changes are modeled by a continuous-time zero-mean white noise. The discrete-time state equation (4.1) with sampling period T is written considering:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{s} \\ \dot{\boldsymbol{s}} \\ \ddot{\boldsymbol{s}} \end{bmatrix}, \qquad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{I} & T\boldsymbol{I} & \frac{1}{2}T^2\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & T\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \qquad \boldsymbol{Q} = E\{\boldsymbol{w}\boldsymbol{w}^T\} = \begin{bmatrix} \frac{1}{20}T^5\boldsymbol{I} & \frac{1}{8}T^4\boldsymbol{I} & \frac{1}{6}T^3\boldsymbol{I} \\ \frac{1}{8}T^4\boldsymbol{I} & \frac{1}{3}T^3\boldsymbol{I} & \frac{1}{2}T^2\boldsymbol{I} \\ \frac{1}{6}T^3\boldsymbol{I} & \frac{1}{2}T^2\boldsymbol{I} & T\boldsymbol{I} \end{bmatrix} \tilde{q} \tag{4.8}$$

where $\boldsymbol{I}$ is the (3×3) identity matrix, $\boldsymbol{0}$ is the (3×3) null matrix, and $\tilde{q}$ is the process noise intensity. Introducing the measurement noise intensity $\tilde{r}$, equations (4.2) are described by:

$$\boldsymbol{z} = \boldsymbol{p}_H^*, \qquad \boldsymbol{H} = [\boldsymbol{I} \quad \boldsymbol{0} \quad \boldsymbol{0}], \qquad \boldsymbol{R} = E\{\boldsymbol{v}\boldsymbol{v}^T\} = \boldsymbol{I}\tilde{r} \tag{4.9}$$

At the end, the predicted state at sample $k+1, \dots, k+n_k$, being n_k the prediction index, is obtained by propagating the updated state through the transition matrix:

$$\overline{\boldsymbol{x}}_{k+1}^- = \boldsymbol{F}_k \overline{\boldsymbol{x}}_k^+, \dots, \overline{\boldsymbol{x}}_{k+n_k}^- = \boldsymbol{F}_k \overline{\boldsymbol{x}}_{k+n_k-1}^- \tag{4.10}$$

and the estimated hand position vector at time index $k+n_k$ is:

$$\boldsymbol{p}_{H,n_k}^e = \overline{\boldsymbol{x}}_{k+n_k}^-(1{:}3) \tag{4.11}$$

In Figure 4.2, the output of the optimization problem of data fusion and the predicted human hand positions resulting from Kalman filter are represented.

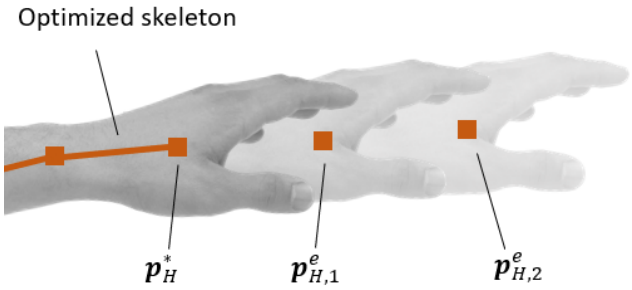


Figure 4.2 Optimized and predicted hand position

The sampling period is the Kinect refresh time rate, which corresponds to $T = 33\ ms$. The prediction index is chosen as $n_k = 2$. It means that the prediction is made with a time horizon of $n_k T = 66\ ms$.

Concerning the order of magnitude for the Kalman filter parameters $\tilde{q}$ and $\tilde{r}$, in general it can be deduced with some prior analysis. However, for the best values, it is required to carry out some experimental tests and choose the ones that lead to the best practical result.

By observing equation (4.8), the process is modeled so that the changes in the acceleration for each coordinate over a sampling period T are on the order of $\sqrt{\tilde{q}T}$. As a consequence, higher values of $\tilde{q}$ would produce a more responsive filter; on the other hand, one can prefer lower values of $\tilde{q}$ for signal smoothing. In this work, various experiments have been conducted and the value $\tilde{q} = 150\ m^2/s^5$ seems to balance these two effects, producing a filtered signal which preserves the information of the original one.

The order of magnitude for the parameter $\tilde{r}$ can be deduced by analyzing the error of the sensor. For example, the authors of [111] apply Kalman filter to Kinect measurements to improve the reliability of motion tracking. The results show that the root-mean-square error in the displacement estimation between measurement and ground truth data varies within $0.02\ m$ and $0.2\ m$, where the higher values show during occlusions. However, multiple Kinect techniques can significantly improve the human pose estimation [38], [112]. In this work, the optimized position of the hand $\boldsymbol{p}_H^*$ is considered in the measurement equations (4.9). This takes into account occlusions and joint-to-joint distances by means of (2.2). After some experiments, the measurement noise $\tilde{r} = 0.002\ m^2$ has been selected, which refers to the reasonable displacement error for each coordinate of $0.045\ m$.

4.3.2 Trajectory generation and motion planning

Without loss of generality, consider an anthropomorphic manipulator as the collaborative robot. The motion that ensures convergence of the TCP position and orientation at the target is obtained through a method similar to the APF-based collision avoidance algorithms presented in Chapter 3. The only difference is that the target is dynamic, as it is calculated from the human hand position and the forearm orientation.

In particular, it was observed that the robot should move towards the front of the hand in order to facilitate the piece exchange. The reason is in the size of the exchanged object, which must be considered. Thus, a virtual point at a certain distance from the hand is introduced. This point is named virtual hand and is indicated as $\boldsymbol{p}_{VH}$. The virtual hand $\boldsymbol{p}_{VH}$ represents a projected point of the estimated hand position $\boldsymbol{p}_{H,2}^e$ in the direction of the forearm axis $\boldsymbol{a}_{FA}$, defined as:

$$\boldsymbol{a}_{FA} = \frac{\boldsymbol{p}_W^* - \boldsymbol{p}_E^*}{\|\boldsymbol{p}_W^* - \boldsymbol{p}_E^*\|} \tag{4.12}$$

where $\boldsymbol{p}_W^*$ and $\boldsymbol{p}_E^*$ denote the optimized position vectors of the wrist and the elbow of the operator. The forearm axis and the significant points are shown in Figure 4.3. Notice that only the hand position $\boldsymbol{p}_{H,2}^e$ is predicted since, during the experimental test, the prediction scheme presented in the previous section revealed weak results with the elbow and wrist joints. Therefore, the virtual hand is built from the predicted signal, but the forearm axis is referred to the optimized elbow and wrist skeleton joints, not to the predicted ones. In the results section it

will be shown that this compromise does not represent a limit to achieve the hand-over. On the contrary, it is simple and produces evident results.

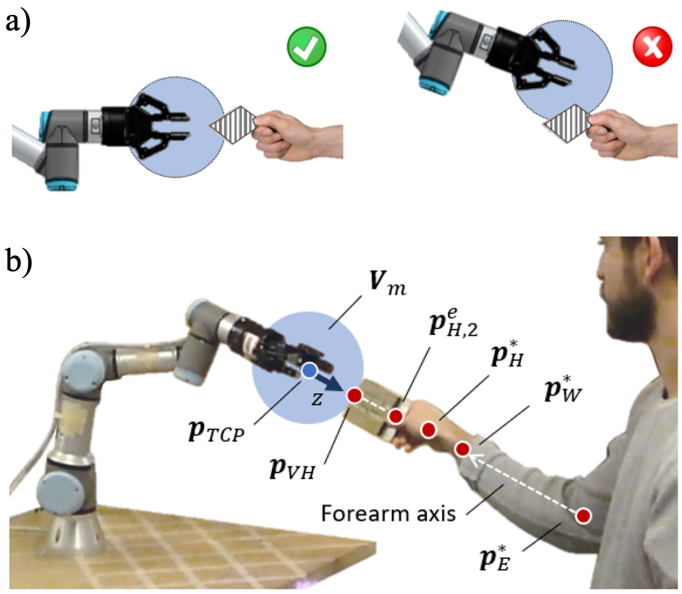


Figure 4.3 a) Desired hand-over. b) Virtual hand and forearm axis

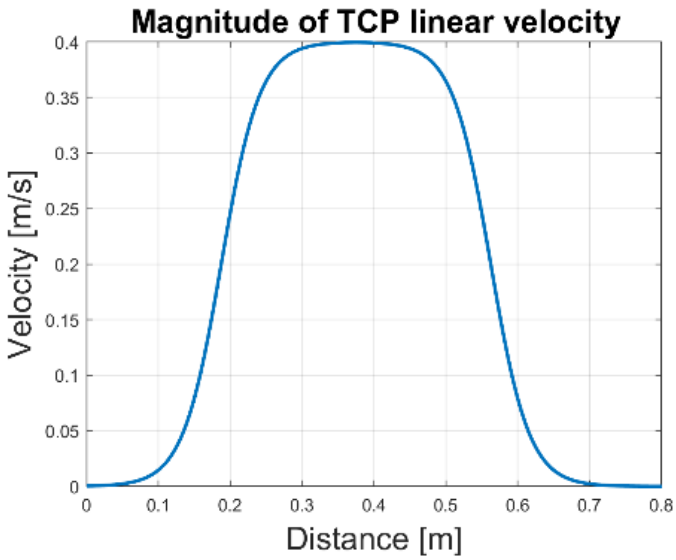


Figure 4.4 Magnitude of the attractive linear velocity of the TCP versus the distance between virtual-hand and robot TCP.

The position problem is addressed with a local planner based on attractive velocities. Let $\boldsymbol{e}_{p,TCP}$ denote the position error with respect to the dynamic target:

$$\boldsymbol{e}_{p,TCP} = \boldsymbol{p}_{VH} - \boldsymbol{p}_{TCP} \tag{4.13}$$

To drive the TCP at the target $\boldsymbol{p}_{VH}$, an attractive velocity vector $\boldsymbol{v}_{att}$ is introduced as:

$$\boldsymbol{v}_{att} = v_{att} \frac{\boldsymbol{e}_{p,TCP}}{\|\boldsymbol{e}_{p,TCP}\|} \tag{4.14}$$

$$\begin{cases} v_{att} = \dfrac{v_{att,max}}{\left(1 + e^{\left((\rho_{att,1} - \|\boldsymbol{e}_{p,TCP}\|)\frac{2}{\rho_{att,1}} - 1\right)\alpha_{att,1}}\right)} & , if \ \|\boldsymbol{e}_{p,TCP}\| \leq \rho_{att,1} \\ v_{att} = \dfrac{v_{att,max}}{\left(1 + e^{\left((\|\boldsymbol{e}_{p,TCP}\| - \rho_{att,1})\frac{2}{\rho_{att,2}} - 1\right)\alpha_{att,2}}\right)} & , if \ \|\boldsymbol{e}_{p,TCP}\| > \rho_{att,1} \end{cases} \tag{4.15}$$

Equation (4.15) gives a bell-shape velocity profile (see Figure 4.4) for a human-like motion, as reported in [84], [85]. In particular, $v_{att,max}$ is the maximum magnitude of the attractive velocity, $\rho_{att,1}$ is the distance at which the magnitude of the velocity reaches the maximum value and $\rho_{att,2}$ defines the length of the right branch of the curve. $\alpha_{att,1}$ and $\alpha_{att,2}$ are the shape factors of the two branches of the curve.

To define the orientation of the virtual hand, it is assumed that the worker's hand is aligned with the forearm axis during the object exchange. To obtain an ergonomic human–robot hand-over, the robot desired orientation is chosen in order to align the z-axis of the TCP with the forearm axis, as shown in Figure 4.3b. In this way, the orientation of the gripper allows a more fluent hand-over operation. The angular velocity vector that permits to obtain the desired TCP orientation is calculated as:

$$\boldsymbol{\omega}_{TCP} = \boldsymbol{K}_o \boldsymbol{e}_{o,TCP} \tag{4.16}$$

where $\boldsymbol{e}_{o,TCP}$ is the quaternion difference between the desired orientation of the TCP and the actual one. The TCP desired pose is:

$$\dot{\boldsymbol{\chi}}_{TCP} = \begin{bmatrix} \boldsymbol{v}_{att} \\ \boldsymbol{\omega}_{TCP} \end{bmatrix} \tag{4.17}$$

To obtain the desired motion, the robot is controlled in terms of joint velocities. The control law is similar to (3.1) and corresponds to the control law of the inverse kinematics algorithm with Jacobian inverse:

$$\dot{\boldsymbol{q}} = J^{-1}(\boldsymbol{q})(\dot{\boldsymbol{\chi}}_{TCP}) \tag{4.18}$$

which is asymptotically stable since $\boldsymbol{K}_o$ is positive definite [13].

4.4 Hand-over strategy

The hand-over algorithm is part of a higher-level hand-over strategy that has been developed to establish the time relationship between the phases of a canonical hand-over process.

Figure 4.5 shows a schematic representation of the collaborative workspace. Assume the operator is close to the robot and can access anytime to the robot workspace. To characterize the hand-over task, the collaborative volume and the meeting volume are introduced. The collaborative volume is defined as the intersection of human's and robot workspace. The meeting volume, indicated with V_m, is a sphere centered at the robot TCP and represents the volume in which the hand-over takes place.

In Figure 4.6, a scheme of the human–robot hand-over task is shown, where the main tasks are inside the boxes, while the subtasks are underlined. In the middle, the events that trigger the subtasks are labelled. The operator and the robot can be both giver and receiver in order to consider a wide range of possible hand-over applications. The human task begins when the operator moves the hand towards the workspace of the robot to reach a desired point. The robot starts to move when the hand of the operator enters in the collaborative volume. The manipulator follows the human through the hand-over algorithm and it stops its following motion when the distance between the TCP and the hand of the operator is less than the radius of the meeting volume; then, the human adjusts his pose so that the hand-over is accomplished. At the end, the operator moves his hand outside the workspace, and the robot returns to its home configuration through a point-to-point motion generated offline in the joint space.

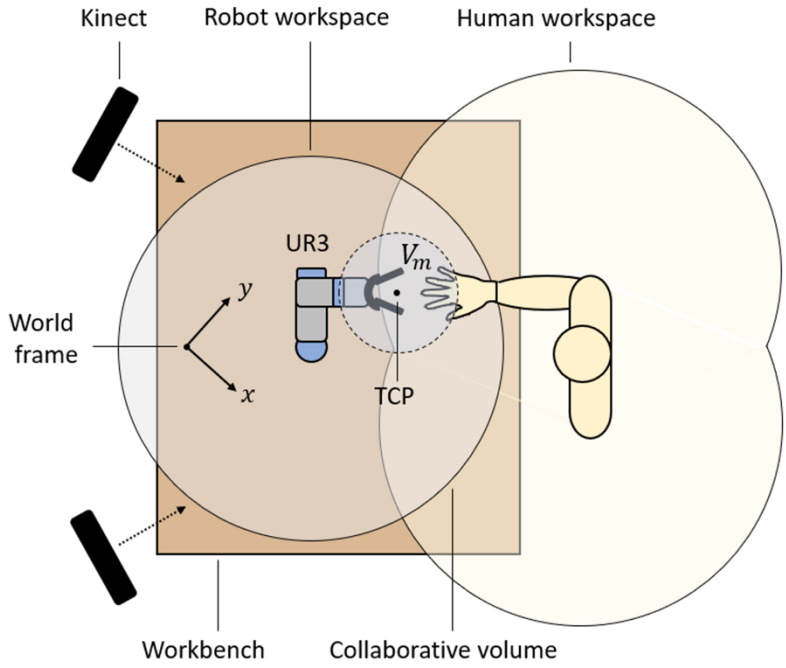


Figure 4.5 Collaborative workspace

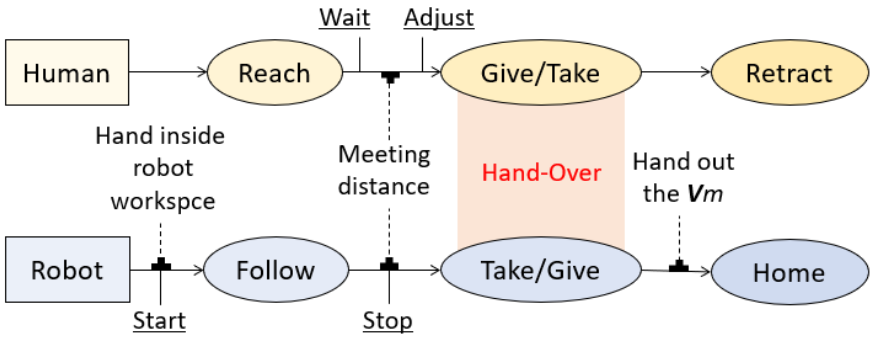


Figure 4.6 Human-robot hand-over sequence

Chapter 5

Results

Different tests have been carried out to prove the effectiveness of collision avoidance and hand-over algorithms. Depending on the grade of novelty, some algorithms are directly validated with a real robot, others are previously verified in a simulation environment.

For instance, the effectiveness of the basic collision avoidance algorithm with skeleton tracking have been already discussed in [12] by means of simulation tools and experimental tests. However, [12] does not show the advantages of collision avoidance in a real-world collaborative task. In this chapter, a section is dedicated to demonstrate how collision avoidance can be used in a collaborative assembly task to guarantee safety and to improve the task time. Another aspect that has not been investigated in [12] is the application of the basic collision avoidance algorithm with the human point cloud. In the following, this case will be analyzed with both simulation and experimental tests. The reason is that by modifying the input of the algorithm, i.e. the way the person is tracked, some parameters requires to be tuned. This step will be done by simulation.

The MAP algorithm is the core of the dissertation. Both simulation and experimental tests will be shown. A prior simulation is fundamental to visualize how the insertion of local attractors modifies the potential field and to analyze the effect of the MAP parameters on the gradient vector field. Successively, the experiments with the real robot are required to validate the control law and to demonstrate how the MAP can be used in collaborative robotics scenarios.

The hand-over algorithm contains a revision on how the target position is estimated. This modifies the input of the algorithm and that does not affect the control law of the manipulator. The effect of prediction on task efficiency compared to the basic version of the algorithm will be proved in a real human-robot hand-over.

The first section of this chapter describes the simulation environment and the robotic cell utilised in the experimental tests. Successively, the outcomes are grouped in two main sections, distinguishing collision avoidance and hand-over tests. The observed strenghts and drawbacks of each algorithm are pointed with a discussion at the end of the relative section.

5.1 Tools and methods

The aim of this section is to present the hardware/software solutions and the control architecture that have been used to obtain a reactive robot motion according to the algorithms presented in the previous chapter. The algorithms are basically pieces of software that elaborate the inputs to produce some outputs in terms of numbers. However, the robot response will depend also on the hardware and software integration, i.e. on tools and techniques used to compute those numbers. Moreover, so far the collaborative robot has been addressed as a system consisting in a generic manipulator that receives commands and executes motion. In this section, the robot is identified as a specific component of a control architecture so that more details on the robot type as well as on the robot controller are given.

5.1.1 Simulation environment

The proof of concept and the parameter tuning of each algorithm has been done in Matlab 2019b running in Windows 10 pro. The computer used for simulation is equipped with i7-7500U processor and 16 GB RAM. To improve efficiency and flexibility of simulation codes, Corke Robotics Toolbox has been utilized [82]. It is a collection of Matlab libraries specifically designed for robot modelling, which provides easy tools and functions to handle direct and inverse kinematics, as well as simulation and animation.

The simulation program is a Matlab script composed of three main sections: human/objects detection, control algorithm and robot model. Human/objects detection and control algorithm sections contain a list of commands that solve the mathematics presented in the previous chapters. The robot motion is calculated

considering the kinematics of existing collaborative robots. In the following sections, the robot models and the simulation program are described.

5.1.1.1 Robot modelling

Two different collaborative robots have been considered for simulation: the LBR iiwa 14 by KUKA [6] and the UR3 by Universal robots [113].

The LBR iiwa 14 is natively a 7dof manipulator and has a spherical workspace with radius of $1.360\ m$ at the end-effector (Figure 5.1). Because of the rounded edge design, human-sized workspace, redundancy, payload ($14\ kg$), and built-in torque sensors at each joint, the LBR iiwa 14 is one of the most flexible collaborative robots on the market. These features made it the first choice when selecting the robot for simulation in the first part of the research, when a real robot was not available.

After the purchase of the most affordable UR3 from DIMEAS (department of mechanical and aerospace engineering) the attention has been moved on the real robot and the UR3 model has been also considered for simulation before the experimental tests. Compared to the LBR iiwa 14, the UR3 has 6dof, a smaller workspace of $0.5\ m$ at the end-effector (Figure 5.1) and a reduced payload of $3\ kg$.

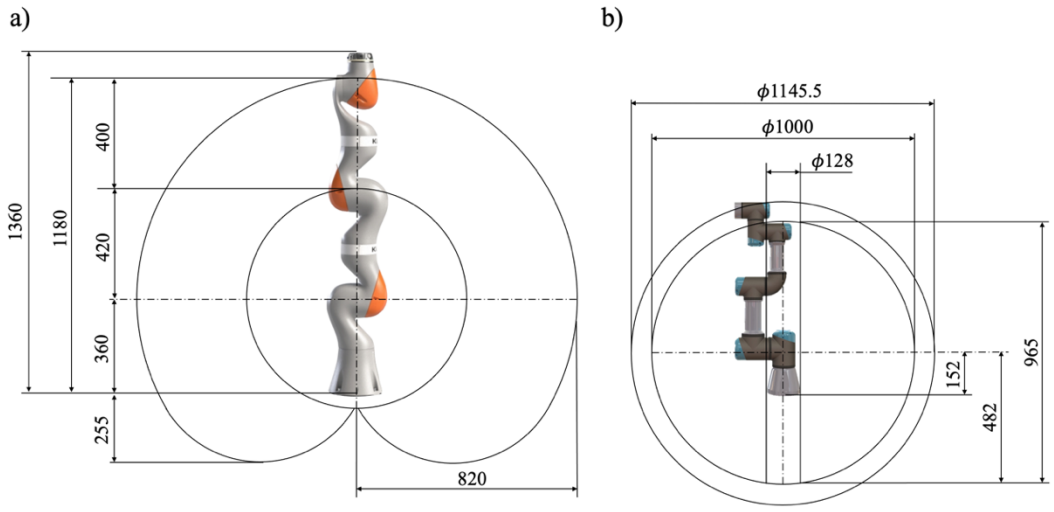


Figure 5.1 Workspace sizes of the robots LBR iiwa 14 (a) and UR3 (b)

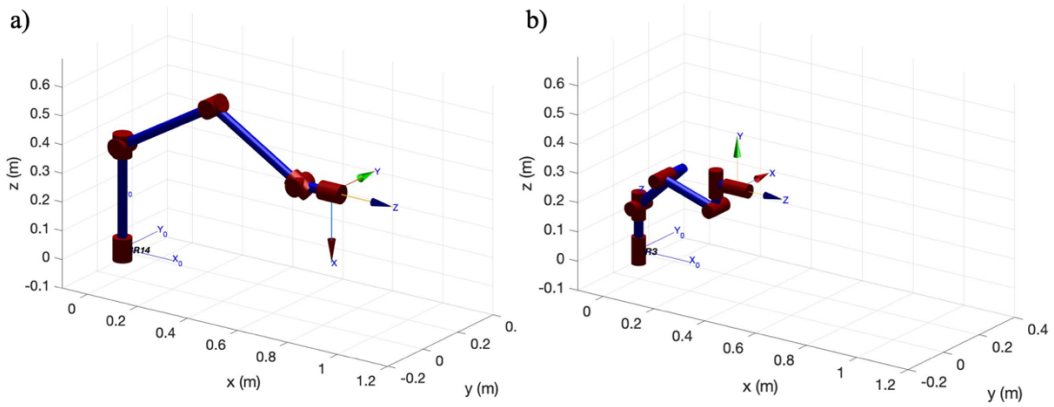


Figure 5.2 Models of robots LBR iiwa 14 (a) and UR3 (b) built in Matlab with the Corke Robotics Toolbox

Table 5.1 Denavit Hartenberg parameters according to modified convention

UR3				
Link	$\alpha_{DH}\ [rad]$	$a_{DH}\ [m]$	$d_{DH}\ [m]$	$\theta_{DH}\ [rad]$
1	0	0	0.1519	$-\pi$
2	$-\pi/2$	0	0	0
3	0	0.24365	0.11985	0
4	0	0.21325	0.00945	0
5	$-\pi/2$	0	0.0834	0
6	$\pi/2$	0	0.0834	0
LBR iiwa 14				
Link	$\alpha_{DH}\ [rad]$	$a_{DH}\ [m]$	$d_{DH}\ [m]$	$\theta_{DH}[rad]$
1	0	0	0.36	0
2	$-\pi/2$	0	0	0
3	$\pi/2$	0	0.42	0
4	$\pi/2$	0	0	0
5	$-\pi/2$	0	0.4	0
6	$-\pi/2$	0	0	0
7	$\pi/2$	0	0.163	0

Because the prior tests with the LBR iiwa 14 model have been made by considering the 3^{rd} joint fixed, so not exploiting the redundancy of the 7dof, the models of the two robots have similar kinematics and the simulated results obtained with the LBR iiwa 14 have been easily transferred to the UR3 model by modifying the DH (from *Denavit-Hartenberg*) parameters and by adapting the task to the reduced size of the workspace. In Figure 5.2, the robot models built with the Corke Robotics Toolbox according to modified DH parameters of Table 5.1 are shown.

5.1.1.2 Software structure

The simulation script consists of a *loop* structured as in Figure 5.3. At each simulation step, the human position is sampled from data acquired with the vision system layout of Chapter 2. Alternatively, in collision avoidance applications, a preliminary test with simple objects can be considered by defining the position and the geometry of the obstacle at each sample. A distance calculation sub-section takes the human and the robot position data to compute the minimum distances. In particular, a *knnsearch* (from *k-nearest-neighbors*, see [114]) is performed between the set of points identifying the human position (skeleton or point cloud) and the significant points related to the robot, depending on the control algorithm. The robot motion is obtained by first order integration. If the controller of the manipulator runs at the frequency f_c, for each step k one can compute:

$$\boldsymbol{q}_k = \boldsymbol{q}_{k-1} + \dot{\boldsymbol{q}}_{set,k} \frac{1}{f_{\boldsymbol{c}}} \tag{5.1}$$

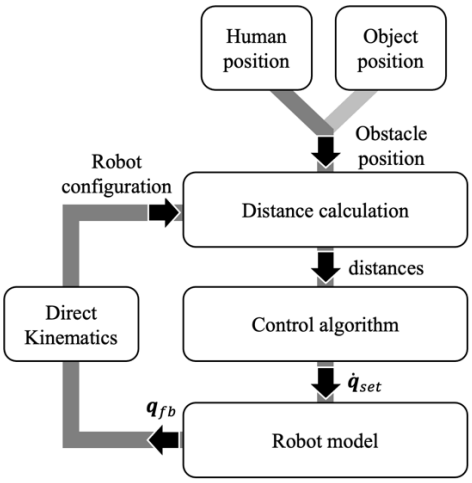


Figure 5.3 Simulation software structure

5.1.2 Experimental set-up

A collaborative robotic cell has been designed to prove the effectiveness of the control algorithms in a real scenario. A picture of the experimental set-up is reported in Figure 5.4. The collaborative cell is built from the vision system presented in Chapter 2. In the following sections, the components and the control architecture are described.

Figure 5.4 Laboratory set-up of the collaborative robotic cell

5.1.2.1 Workspace design

The robotic cell can be divided in two main areas (Figure 5.5): the collaborative workspace and the computing area. The collaborative workspace is composed of a workbench, a collaborative robot UR3 equipped with a Robotiq 2F-85 gripper [115] and 2 Kinect. The layout of the multiple Kinect have been chosen according to the best performance of the vision system ([38], [40]). The robot is mounted in front of the operator to facilitate the range of movements ([116]). The computing area is composed of the robot controller, 3 computers and a HUB. The hardware specifications of each component are listed in Table 5.2. The controller terminates at the HUB through an ethernet cable. At the same manner, the HUB hosts three computers so that the controller and the computers are physically connected under the same local network.

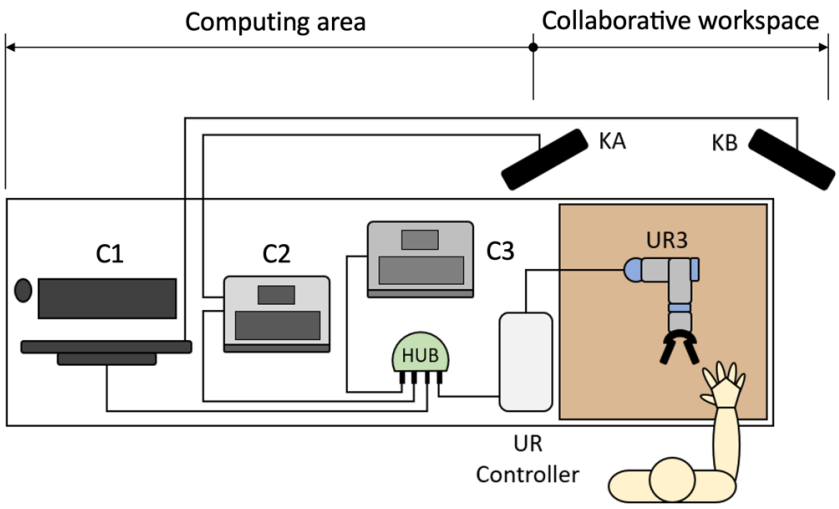


Figure 5.5 Schematic of the robotic cell

Table 5.2 Hardware specifications

Component	Specifications
C1	i7-6700, 32 GB RAM, NVIDIA GTX 970
C2	i7-7500U, 16 GB RAM, NVIDIA GTX 950M
C3	i7-6700 HQ, 16 GB RAM, NVIDIA GTX 960M
KA, KB	Microsoft Kinect v2
HUB	Tp-link Wireless N Router, 300 Mbps
Cables	4 x CAT 5e Ethernet

5.1.2.2 Software and control architecture

The control architecture of the system is schematized in Figure 5.6. Each computer runs Matlab 2019b.

C1 and C2 acquire and merge signals from two Kinect sensors in a master-slave configuration, as described in Chapter 2. To synchronize C1 and C2, the master sends a trigger to the slave through the HUB and they acquire Kinect frames at the same time.

Optimized human data position is sent from C1 to C3 at 30 Hz. C3 also receives the robot feedback from the UR controller. Further information related to the feedback data can be found in [117]. The kinematic model of the UR3 is

implemented in C3, which uses the feedback data to obtain the actual configuration of the robot. Human-robot distances are computed with the *knnsearch* Matlab function, which is the same used for simulations.

Human and robot positions are the input of the control algorithm, that calculate the signal for the UR3 according to collision avoidance and hand-over strategies, i.e. trajectory planning and motion control techniques presented in Chapter 3 and Chapter 4. In each case, the output is the vector of joint velocities, calculated with a frequency of $62.5\ Hz$.

The control vector is sent to the UR Controller as input of URScript commands *speedj* and *stopj* [118]. These commands are sent as strings to the UR Controller. The collaborative robot UR3 receives commands from the robot controller with a frequency of $125\ Hz$.

The URScript command is a high-level control of the UR3 and permits to benefit of the default safety functions of the robot. In this way, the robotic cell is monitored with another safety tool beyond the collision avoidance algorithm. For example, UR safety functions allow to set the maximum values of the TCP velocity and of the joint velocities. Furthermore, a safety function permits to define planes that the TCP of the collaborative robot must not cross. Two planes have been defined. One plane is parallel to the table and positioned $50\ mm$ above it; the other plane is vertical and positioned in front of the Kinect sensors. In this way, the TCP does not collide with the table and the Kinect v2 sensors. Violations of safety functions lead to a stop of the UR3.

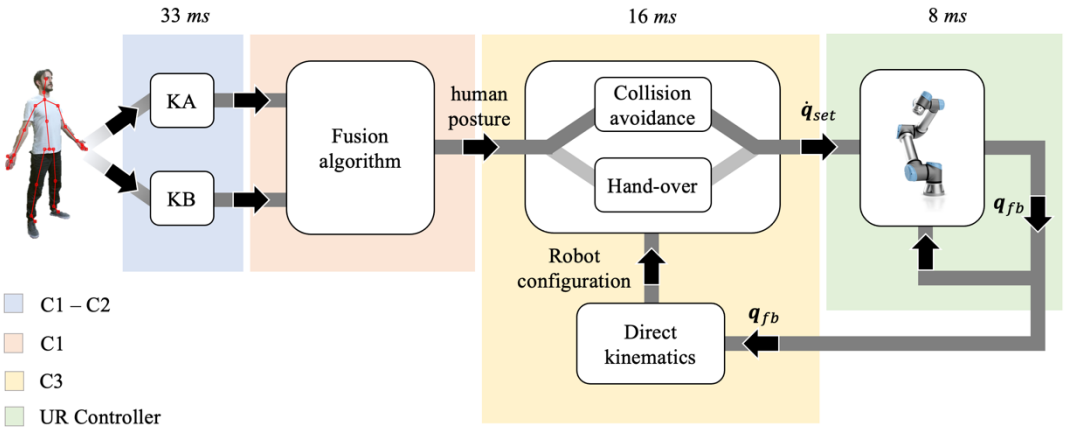


Figure 5.6 Control architecture of the robotic cell

5.2 Collision avoidance

5.2.1 Application of the *robot links - human* algorithm to collaborative assembly task

This section investigates how the robotic cell driven by 3D vision system and the basic collision avoidance algorithm can be used to exploit the advantages of the highest level of collaboration in a specific assembly task. The idea is to employ a human worker in the added value operations of the task, i.e. assembly, while letting a robot collect the parts to be assembled and deliver them on the assembly table, where the human operates. Thanks to the collision avoidance algorithm, the operator can access the shared workspace anytime, without being influenced by the presence of the robot.

5.2.1.1 Workspace and task design

The case study is an assembly cell which receives three parts to be assembled as input and produces an assembly made of the same parts. The parts consist of cubes of different sizes, while the assembly is obtained by sticking the cubes together, as shown in Figure 5.7.

The entire task can be divided in two main operations: collecting the components to be assembled and assembling them. Concerning the first operation, the components are supposed to be stored so that a robot can easily pick them up. The second operation is supposed to be too complex or too variable for a full automation solution, so that the human presence is required. For this reason, the operator should be able to take the parts to be assembled from a certain area which must be a portion of the robot workspace, since the robot has the role to provide the parts to the operator. With this aim, the workspace of the robot is divided in two zones, as shown in Figure 5.8. The parts are stored in the “pick zone”, where no human presence is expected. In the “collaborative zone”, where the human and robot can potentially come to contact as their workspaces intersect, the operator takes the components to be assembled.

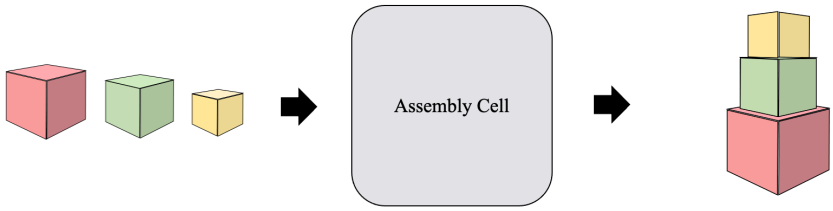


Figure 5.7 Assembly process

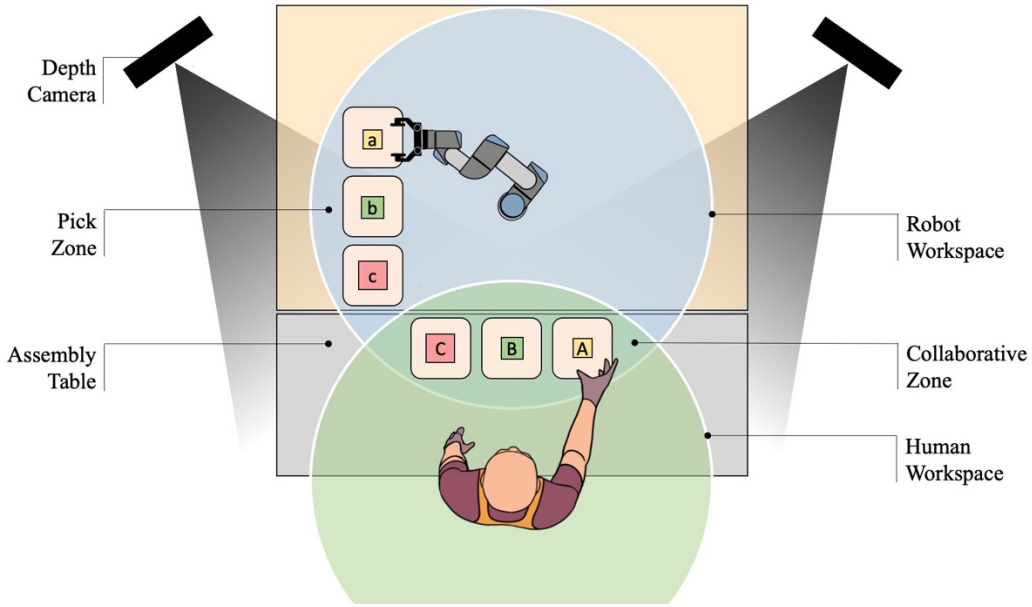


Figure 5.8. Collaborative assembly cell

The human worker and robot perform seven operations in parallel to carry out the assembly task, as reported in Figure 5.9. The task, which is also identified as a cycle, starts when the operator takes the cube positioned in station A. As the human hand enters in the collaborative zone, the robot task begins by replacing the cube in A with a new one from station a. In parallel, the operator puts the glue on the first cube, takes the second cube by station B, and prepares both cubes to be glued. The next step for the robot is taking a cube from station b and placing it in station B, to replace the one taken by the operator. Meanwhile, the human picks up the last cube in station C and assembles the tower of cubes. Finally, the robot replaces the cube in station C and returns in the home position, while the worker checks the edges of the assembly to remove the exceeding glue before delivering.

Concerning the human task, the phases in which he takes the components happen in the collaborative zone, while the other phases are carried out on the assembly table, outside of the robot workspace. On the other hand, the “pick” stages of the robot task identify the robot motion in the pick zone, i.e. outside the collaborative zone, while the “place” stages identify the robot motion within the collaborative zone. The execution of these operations involves a responsive collaboration. In fact, as the human and robot carry out the subtasks in parallel, the operator can access the collaborative zone at any time, even if the robot is placing a component. The experimental setup specifically modified to carry out the assembly task is shown in Figure 5.10.

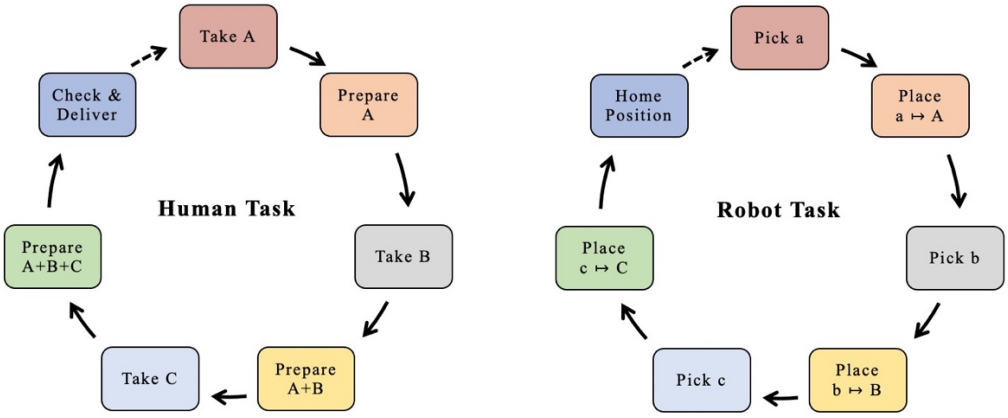


Figure 5.9 Assembly task phases

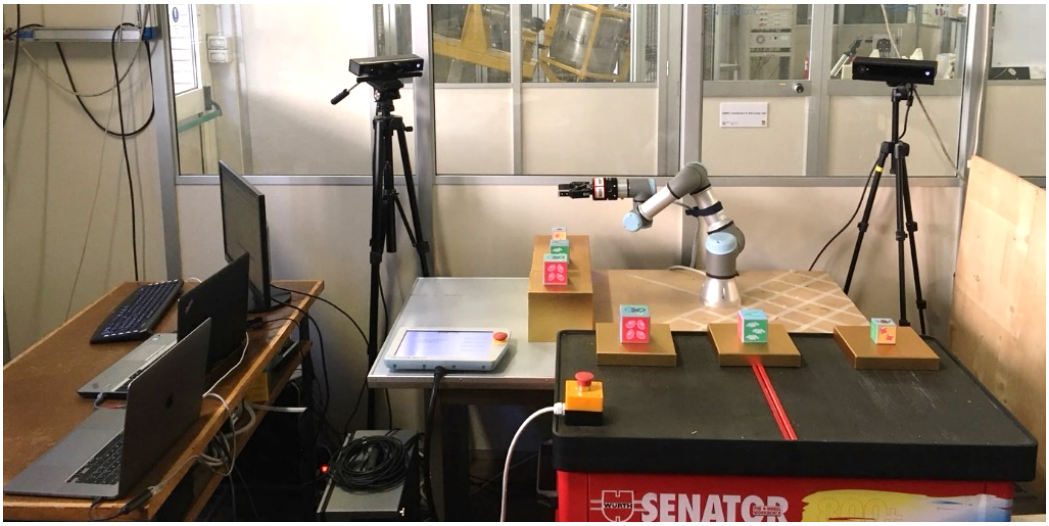


Figure 5.10 Experimental layout of the collaborative assembly cell

5.2.1.2 Robot control strategy

The TCP motion is planned by combining planar trajectories that trigger and terminate at strategical nodes. When the robot moves above the stations of the components from the pick zone to the collaborative zone, the TCP planned trajectory is planar at $z = 0.3\ m$, where z is measured from the robot mounting table. To grasp or release the cubes, the robot follows a vertical path. The strategical nodes are defined as the intersection points between the plane $z = 0.3\ m$ and the vertical paths.

For example, in Figure 5.11 the main directions of the robot planned trajectory in the collaborative zone are shown, together with the x and y axis of the robot base. Regardless of the cube to be placed, the robot approaches the

collaborative zone always from point $\boldsymbol{p}_C$, then it navigates towards the station to be refilled. Once the component has been replaced, the robot drives backwards through the same path, returns in $\boldsymbol{p}_C$ and leaves the collaborative zone. However, due to the human presence, the planned trajectory may deviate from the rectilinear paths. In fact, if the operator takes a component and gets close to the robot, which is going to replace another one, the risk of collision occurs. Then, the robot planned trajectory is modified through the collision avoidance algorithm.

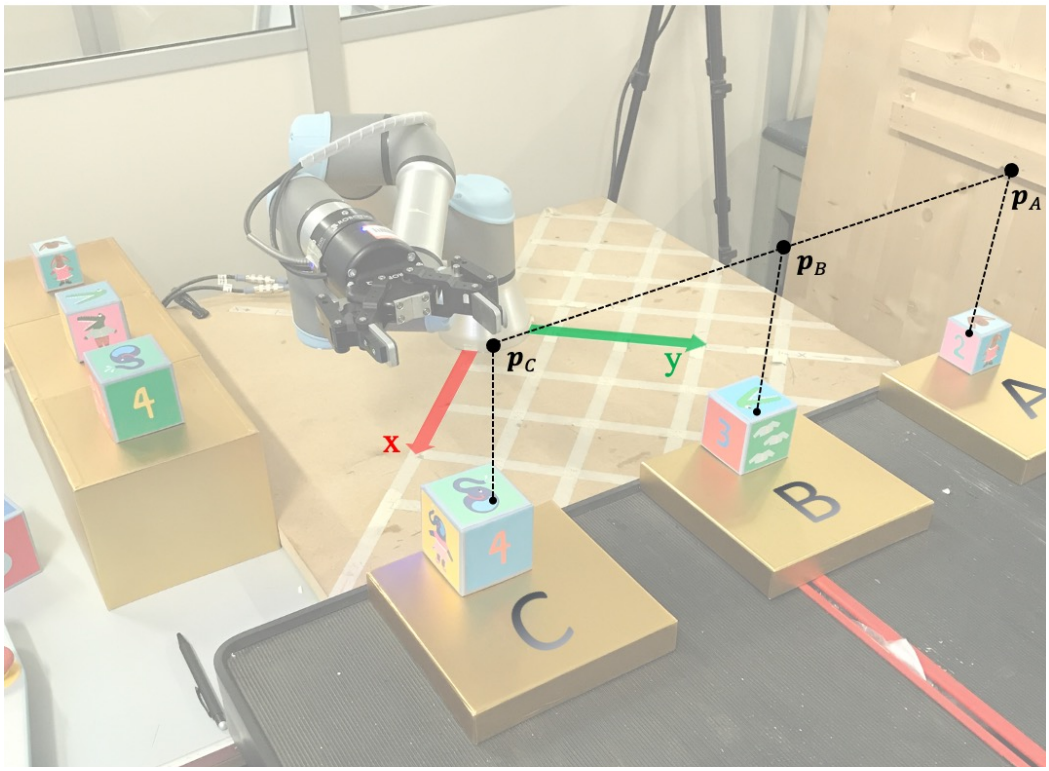


Figure 5.11 Planned trajectory in the collaborative zone

In particular, to avoid undesired contacts between the worker and the moving links of the manipulator, nine points divided in five sets have been assumed within the robot, as seen in Figure 5.12. The positions of these reference points are calculated by direct kinematics, exploiting the feedback joint degrees of freedom provided by the UR controller. The distances between the human joints and the points on the UR3 robot are then evaluated using the *knnsearch* function. Given the human-robot distances $\boldsymbol{d}_{Li-human}$, the repulsive velocities $\boldsymbol{v}_{L_i}$ are calculated for each cluster using the equations (3.7),(3.8).

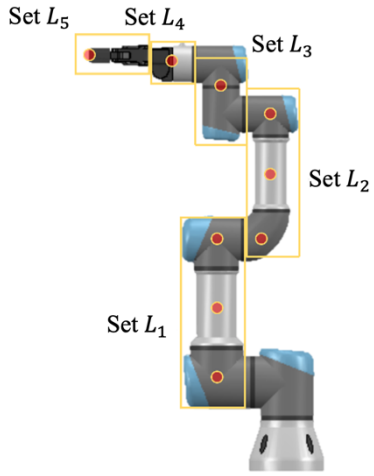


Figure 5.12 Control points considered on the UR3 robot

The magnitude $v_{rep,max}$ has been modelled as in Figure 3.2. Concerning the value of $\rho = 0.4\ m$ (that determines the distance between the control points on the robot and the skeleton that triggers the repulsive effect), even if it may seem quite conservative, takes into account the real dimensions of the human body and robot. In fact, $\boldsymbol{d}_{Li-human}$ are point-to-point distances, so the human body and robot dimensions have been included in the reference distance ρ. Moreover, ρ also considers the position uncertainties of the operator and robot resulting from measurement tolerances of the sensing devices according to what is required by the standards. Concerning $v_{rep,max}$ and α, their values are chosen considering the TCP velocity during the pick-and-place phases, to obtain evasive movements of the robot from the operator which were smooth and seamless.

It is important to highlight that the values of the three parameters $v_{rep,max}$, ρ and α used in the experimental phase result in a safe collision avoidance in the sense that they can prevent the collision with the operator, even if the algorithm described in this section does not fully respect the equations related to the SSM presented in [10]. In the technical specification, the value of the minimum human-robot distance is not constant and depends on several terms. The velocity of the human operator is one of the terms that modify the value of the minimum distance. In this task, a constant reference distance ρ was adopted and the influence of the human velocity was taken into account choosing sufficiently precautionary values of the three parameters involved in the calculation of the repulsive velocities.

5.2.1.3 Results and discussion

The effectiveness of the proposed collaborative layout is evaluated in terms of safety and productivity. Concerning the safety, the experimental tests prove that the collision avoidance algorithm can quickly drive the robot through alternative trajectories in order to avoid the human.

Figure 5.13 shows an example of collision avoidance that happens when the robot is placing $a \mapsto A$ and the operator is taking B. For each frame, the simulation environment is reported in the robot base reference frame. In the latter, the robot links are represented as blue lines, together with the nine control points. The optimized skeleton of the human upper body is colored in red and made of 15 significant joints. The robot planned trajectory is also shown as the rectilinear dashed path connecting $\boldsymbol{p}_C$ to $\boldsymbol{p}_A$, while the solid line is the trace of the actual collision avoidance trajectory.

Observing Figure 5.13, the robot approaches the collaborative zone from point $\boldsymbol{p}_C$ and starts following the rectilinear planned path towards $\boldsymbol{p}_A$, while the human is still applying the glue on the first cube (Frame 1). Then, the operator finishes preparing A (Frame 2) and takes B, while the robot is crossing the relative station, so that the robot reacts backwards to avoid the human hand (Frame 3). As the hand retracts, the robot returns on the planned path (Frames 4 and 5) and reaches $\boldsymbol{p}_A$, while the human is preparing $A + B$ (Frame 6).

The collision avoidance trajectory is analyzed in Figure 5.14. The robot planned trajectory is also shown with the dashed line, together with the strategical points. To give an idea of the repulsive effect, the repulsive velocity vectors acting on the TCP are reported on the actual path at significant samples. The repulsive effect increases nearby $\boldsymbol{p}_B$, which is the point above station B where the human takes the cube. Here, the actual path bends backwards due to the presence of the human hand. The operator takes the cube within a time interval of $1\ s$, so the repulsive effect is localized only on a limited portion of the collision avoidance path.

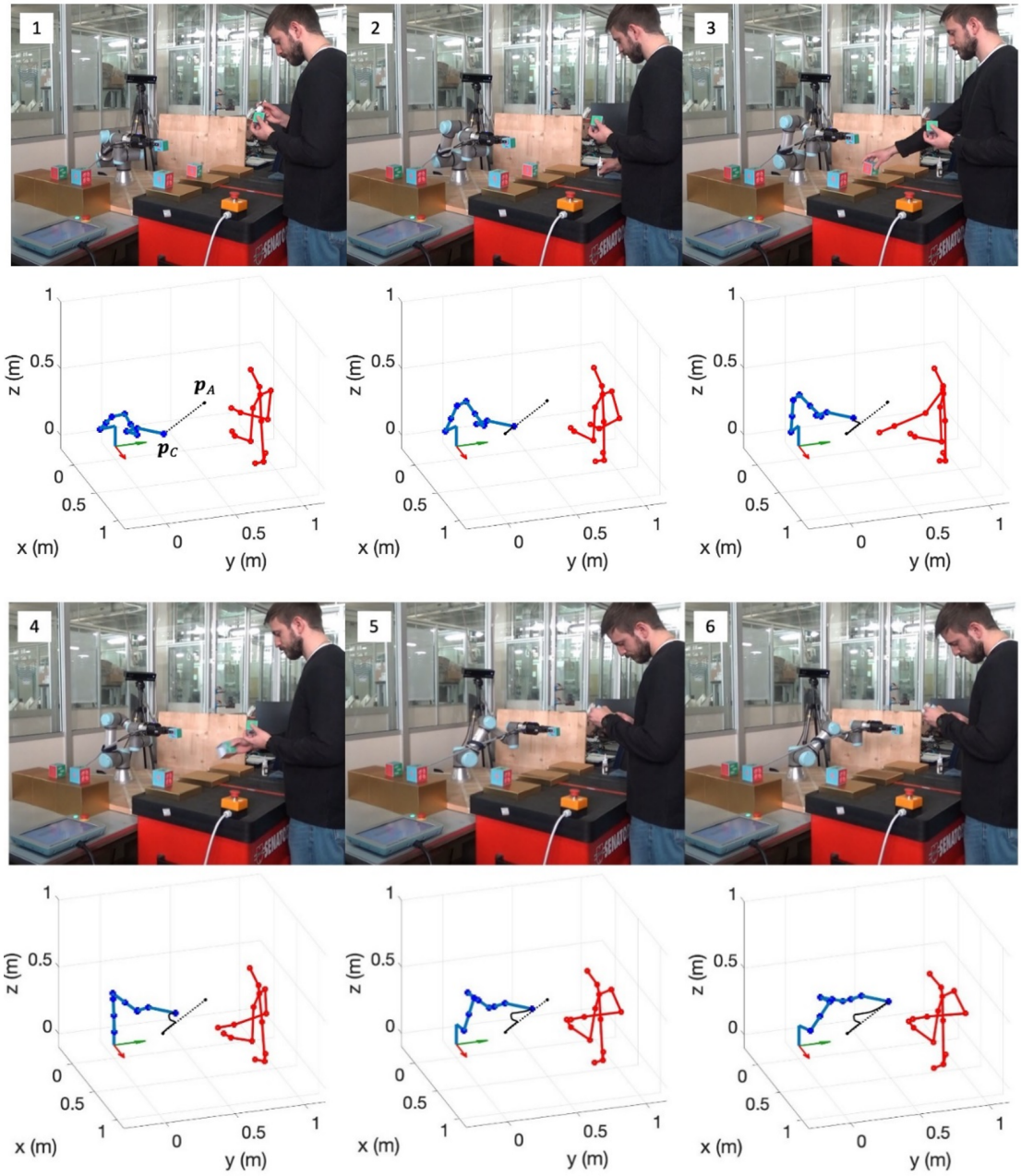


Figure 5.13 Frames of the collision avoidance during the "Place a↦A" stage

The possibility to carry out the task with the highest level of collaboration brings significant advantages in terms of task times. At the top of Figure 5.15, the workflow diagram of the proposed assembly cell is shown. In this case, which represents a responsive collaboration, the human and the robot complete their task in the same time interval, resulting in a mean cycle time of 112 s. Human and

robot tasks run in parallel and the robot collaborates while the human executes value added operations, i.e. actions that carry on the assembly process.

To stress the importance of this result, it is assumed that the same assembly cycle is carried out by considering the lowest level of collaboration. The latter identifies a sequential collaboration, where the human and robot workspaces intersect, but without the possibility for the human to access the collaborative zone if it is already occupied by the robot, and vice-versa. With this more restrictive safety constraint, the workflow diagram would be the one in the middle of Figure 5.15. In fact, when the operator finishes preparing A, he cannot enter in the collaborative zone until the robot accomplishes the "Place $a \mapsto A$" phase. After that, the robot leaves the collaborative zone and the operator can take B. This generates a first delay of approximately 23 s on the human's task. A second delay of approximately 3 s would be at the end of the "Prepare $A + B$" phase, since the robot has not finished yet the "Place $a \mapsto A$" stage and is still in the collaborative zone, therefore, the human must wait before he takes C. The mean cycle time, in this case, is 138 s. Comparing the latter result with the previous case, the major result is that the highest level of collaboration optimizes productivity by saving up to the 18% of the cycle time.

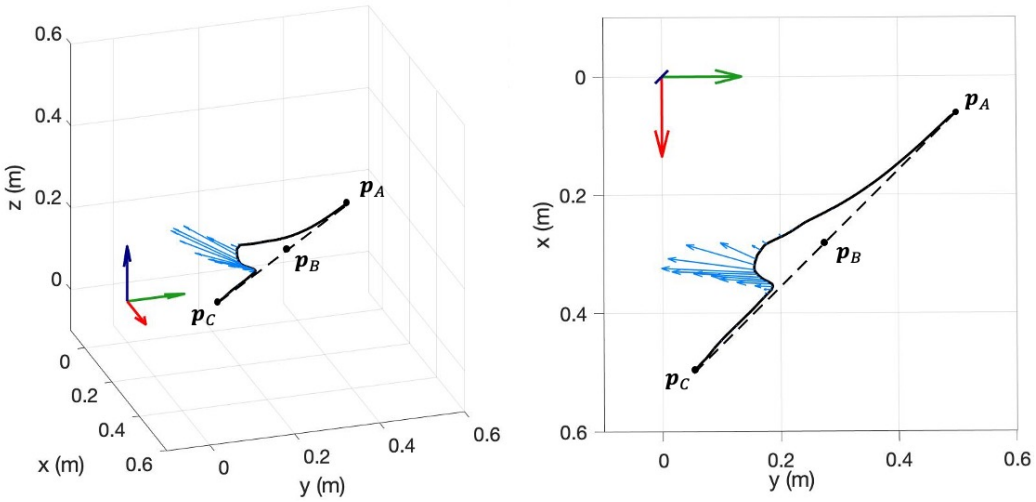


Figure 5.14 Collision avoidance trajectory during the Place a↦A stage

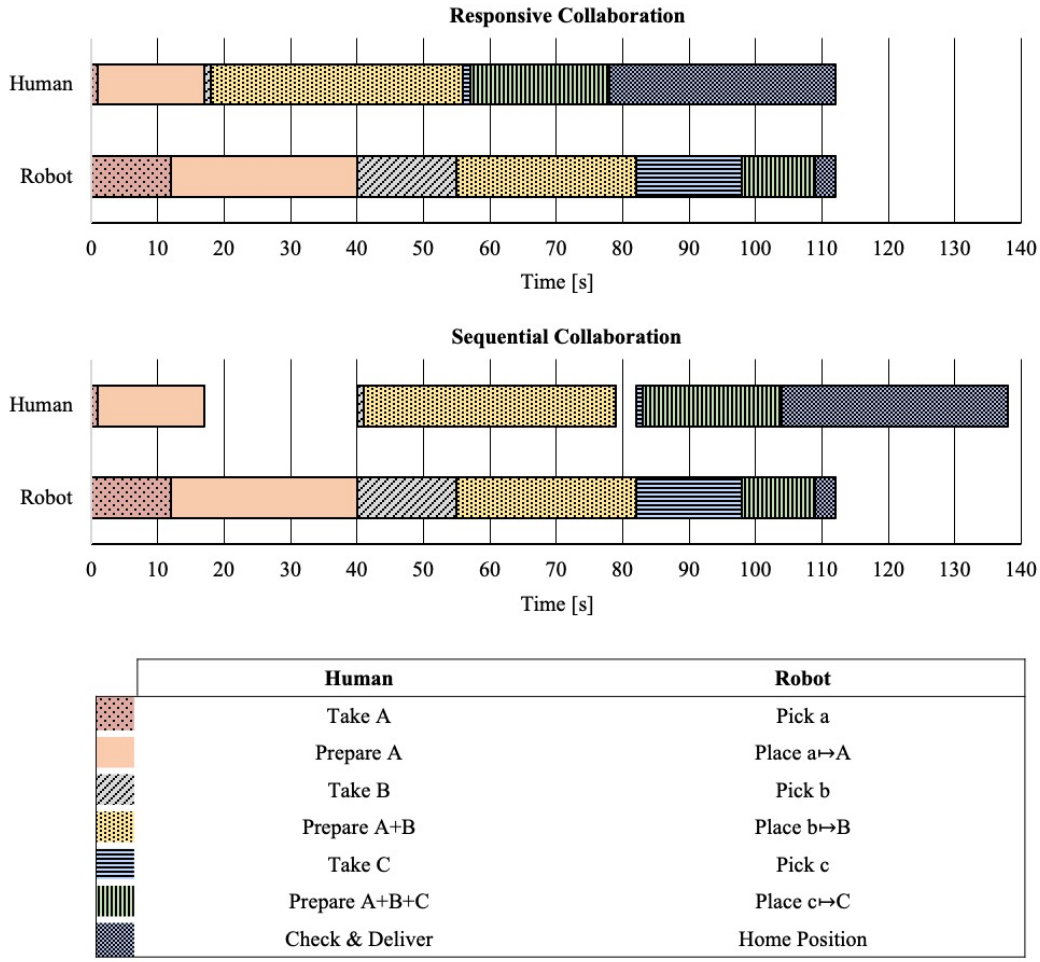


Figure 5.15 Workflow diagram of the assembly cycle for responsive and sequential collaborations

5.2.2 Point cloud-based collision avoidance

In this section, the *robot links - human* collision avoidance algorithm is applied representing the human size by means of point cloud. The influence of the point cloud density on distances calculation time and the possibility to use the convex hull instead of the point cloud are investigated by simulations. Therefore, an experimental demonstration of the algorithm is proposed.

5.2.2.1 Simulation

Simulation tests are carried out considering the LBR iiwa 14 model, with the hybrid setup of Figure 5.16. Human motion is acquired experimentally by the 3D vision system, while the robot motion is simulated in Matlab. The human moves handling objects on the table as it would perform an assembly task, while the

simulated robot is moving with its base positioned at the world frame, which is built from the physical markers.

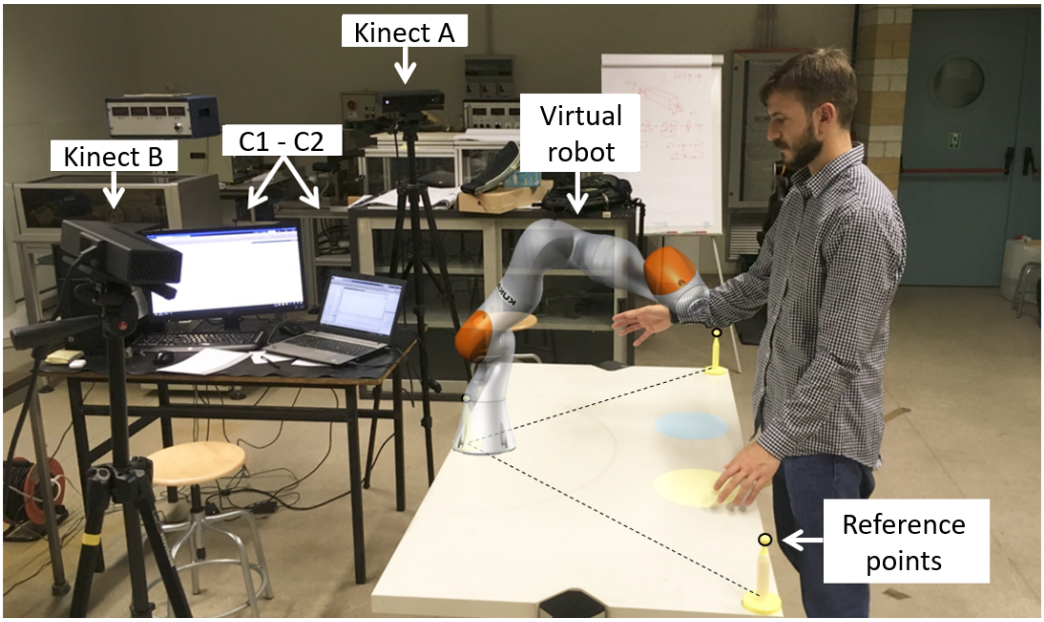


Figure 5.16 Set-up for human motion data acquisition and robot simulation

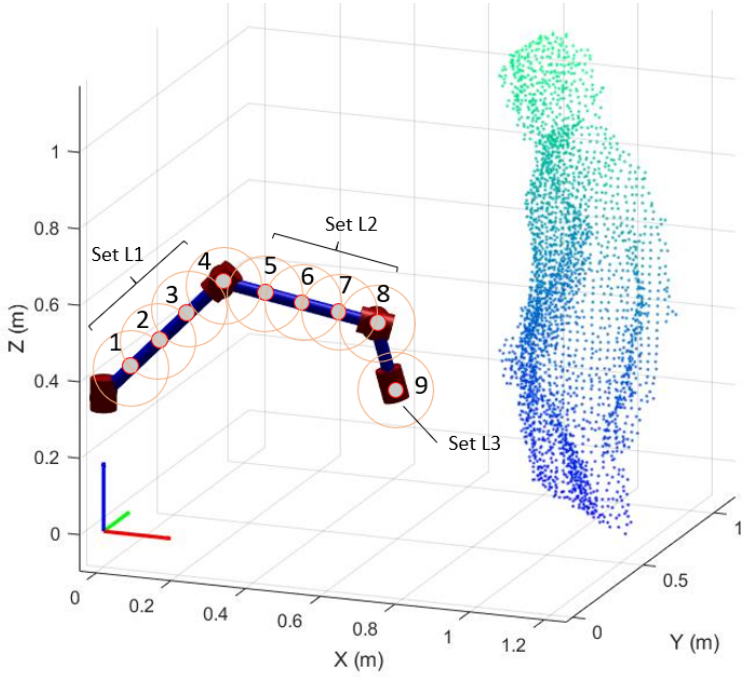


Figure 5.17 Simulation environment for collision avoidance with the point cloud

To interface the point cloud with the robot, two different techniques are used. The first one simply considers the point cloud. The minimum distance between all the human points and a discrete number of control points on the robot is calculated using the *knnsearch* function. The second approach is based on the convex hull of the point cloud generated by means of triangulation; here the minimum distances are estimated using mesh to point algorithm [119]. The second approach excludes the case in which the robot would choose a path between the human hand and his body, representing a more conservative approach. The simulation environment with the acquired point cloud and the robot model is shown in Figure 5.17.

The first test is carried out with the point cloud using a grid step of 0.03. The frames of the test are reported in Figure 5.18. The human reaches the workbench with his right arm, while the serial manipulator is pointing towards the final task configuration. When the hand crosses the robot planned trajectory, the end-effector starts to move away from the obstacle, choosing a path which depends on the shape of the obstacle and its relative position from the human operator. In Figure 5.19 the resulting collision-free path of the end-effector is depicted.

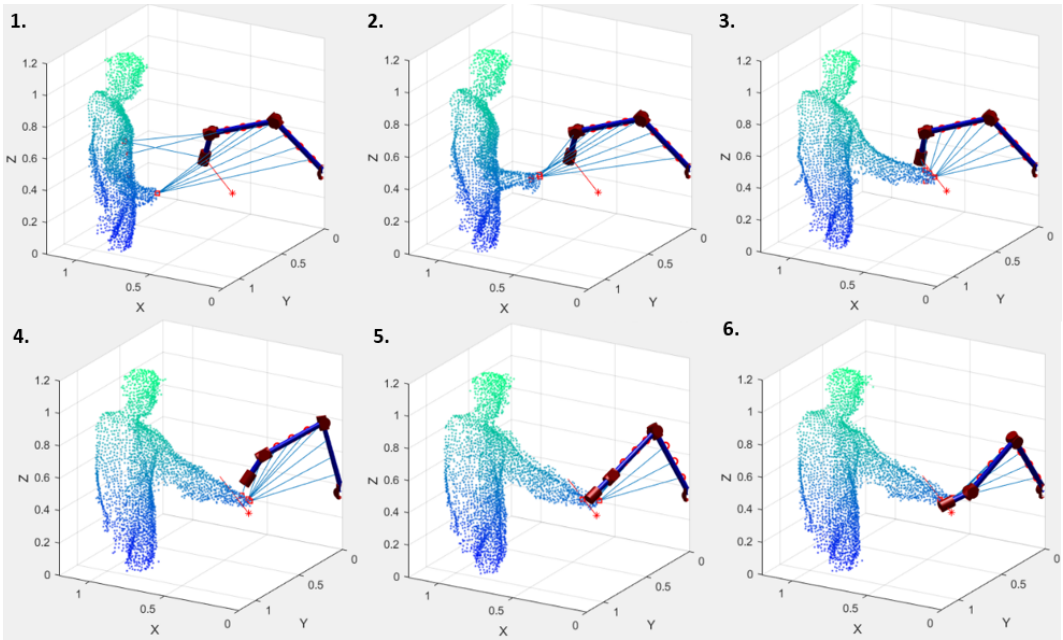


Figure 5.18 Frames of the simulation with the point cloud

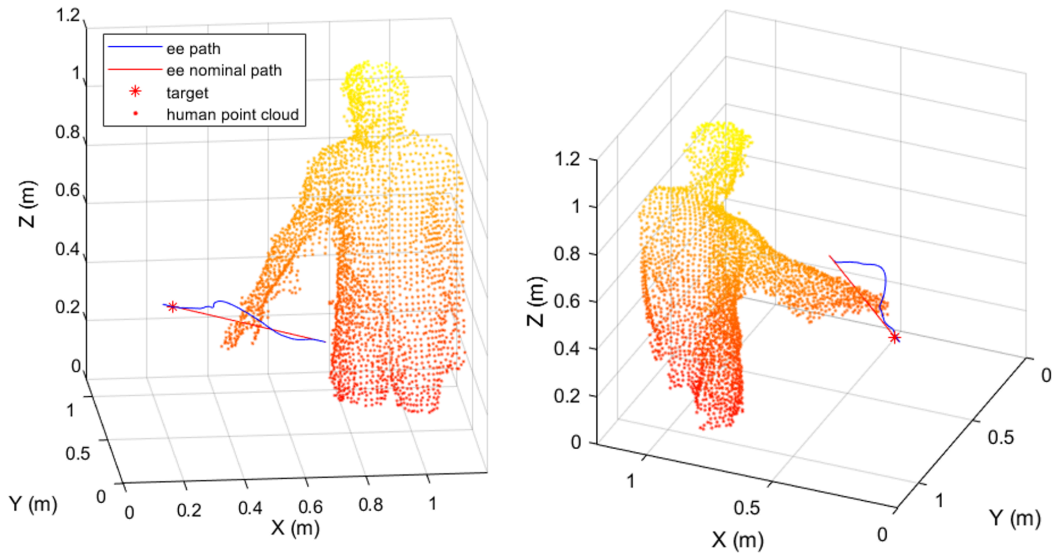


Figure 5.19 Robot path resulting from the simulation with the point cloud

A second test is performed by considering the human convex hull, represented as a mesh. The simulation is run with the same human motion acquired in the test with the point cloud. Figure 5.20 represents a comparison of distance calculation in the starting frame of the two tests, in which the human is going to move his hands towards the robot. The convex hull approach is more conservative as expected since the robot is detected at a lower distance from the mesh. This aspect is confirmed by the analysis of the end-effector paths for the two cases (Figure 5.21). If the human is represented by a triangle mesh, the convex hull fills the space between the human hand and his body (see Figure 5.20). The resulting distance vectors from the control points on the robot are different both in module and direction.

Focusing on the end effector, this can be observed also in Figure 5.22a, where the normalized repulsive velocity vectors of the end effector $\boldsymbol{v}_{L3}/v_{rep,max}$ is plotted for each test sample. The blue vectors refer to the point cloud approach while the red ones to the convex hull. When the obstacle is represented by means of point cloud, the robot drives its end-effector choosing to pass above the hand, in a resulting path which is closer to the human body. On the other hand, the mesh generates a repulsive velocity vector which deforms the path backward. Therefore, when safety is a crucial factor, it can be useful to compute distances from a convex hull rather than a point cloud.

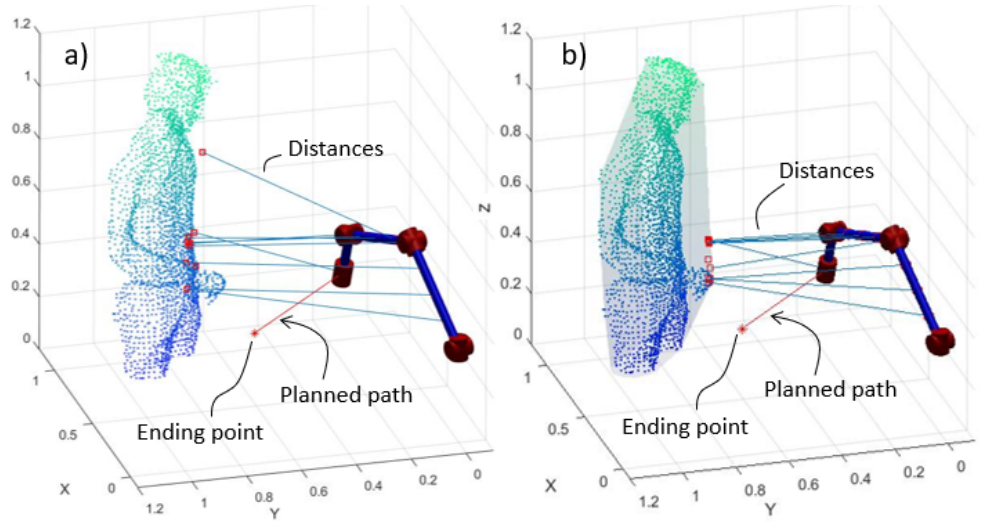


Figure 5.20 Minimum distances calculation by means of point cloud (a) and convex hull (b)

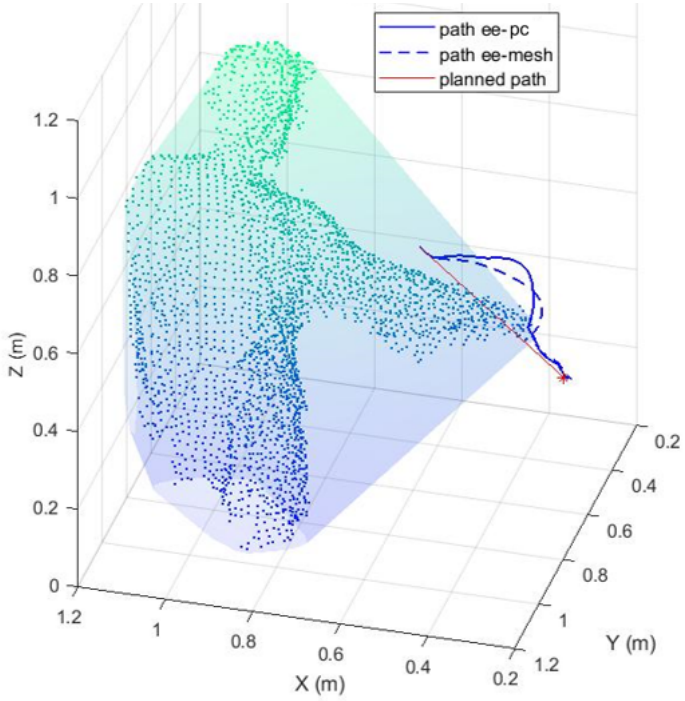


Figure 5.21 Collision free paths of the end-effetor

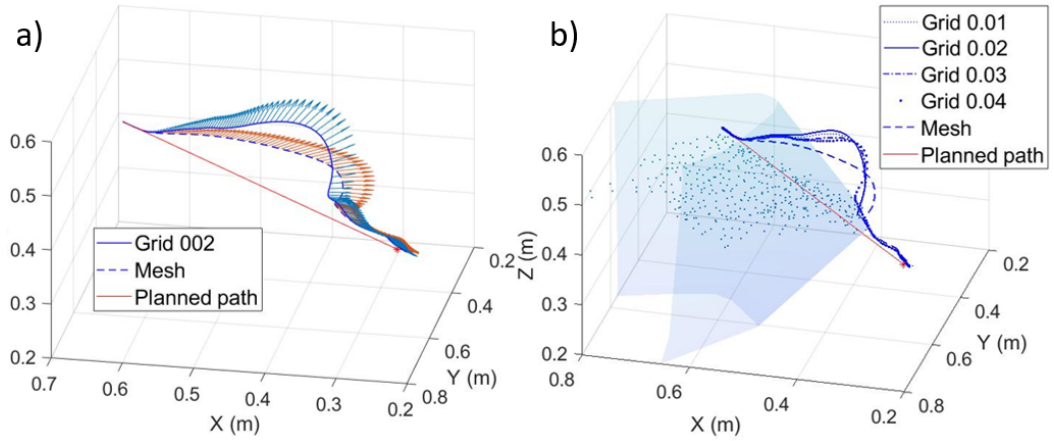


Figure 5.22 a) Effect of the repulsive velocity. b) influence of different downsampling grid steps on the end-effector path

The influence of downsampling on the end-effector paths when the point cloud is used to compute distances is presented in Figure 5.22b. The curves are very similar in all cases, which means that the grid step does not significantly influence the robot trajectory. This suggests selecting the larger grid step in order to reduce the computation time.

For instance, the total times of human data fusion, human-robot distance calculation and collision avoidance algorithm is reported in Table 5.3 for significant values of the grid step. Observing the calculation times during experiments, building the triangle mesh and calculate distances, or computing distances with *knnsearch*, has the same weight in terms of time processing. It means both point cloud and convex hull can be chosen to detect the human motion with an efficient data flow.

Table 5.3 Calculation times (in ms)

	Point Cloud		**Convex Hull**	
	grid 0.02	grid 0.03	grid 0.03	grid 0.04
Fusion algorithm	21.8	19.5	19.4	17.1
Distance calculation	2.5	1.4	2.5	1
Collision avoidance	5	5	5	5
Total time	29.3	25.9	26.9	23.1

Finally, Figure 5.23 describes the effect of the grid step (indicated with different line types) on computed distance values for each link set (drawn with different colors). In particular, the minimum distance from each link set is plotted

as a function of the task time. The distance values are similar in all cases, which means that downsampling does not significantly influence the relative position between the human body and each of the robot links.

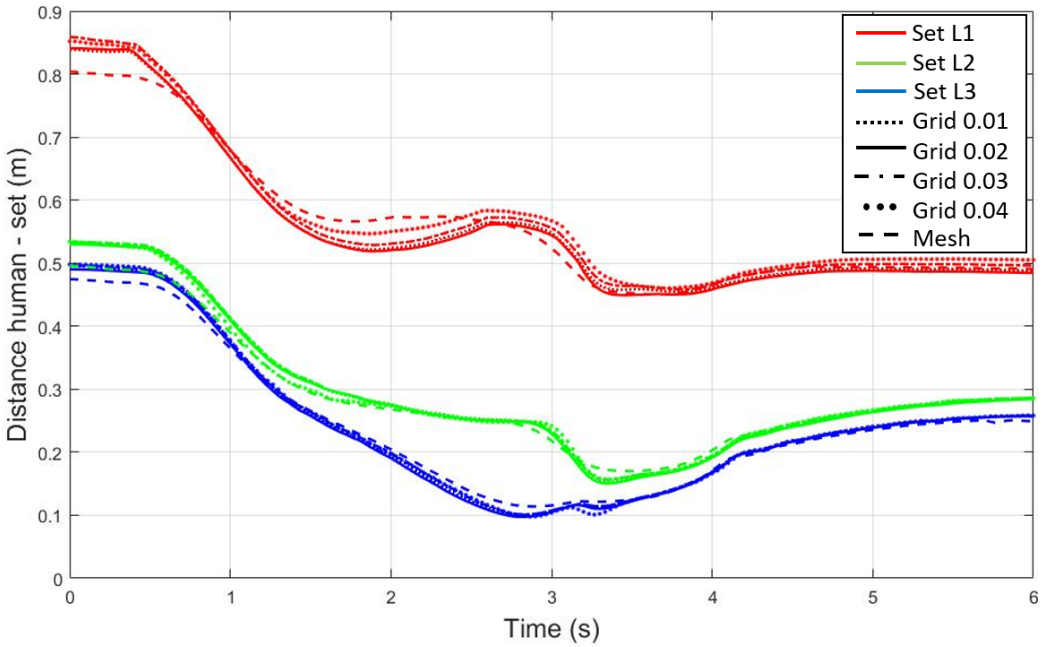


Figure 5.23 Minimum distance of each link set from the human

5.2.2.2 Experimental test

A test similar to the ones of the simulation environment has been carried out with the collaborative robotic cell. The robot UR3 is represented by means of 9 control points, divided in 5 sets (see Figure 5.12). The end-effector is programmed to follow a rectilinear path, as in Figure 5.20. During the robot motion, the human operator intercepts the robot planned motion from different directions.

The case with the hand approaching the robot tool frontally is depicted by the frames of Figure 5.24. In the same figure, for each frame, the Matlab calculation environment with the robot control points and the minimum distances is shown. In Figure 5.25, the human hand intercepts the robot wrist from the top. In both cases, the robot responds to human motion according to the collision avoidance algorithm.

However, real word experiments put in evidence some limitations. In fact, when the operator is close to the robot, the human point cloud segmentation becomes noisy. For the most part, this can be fixed by excluding the points of the point cloud proximal to the robot, as the robot configuration is given by feedback

data and direct kinematics. But the presence of false positives can still occur. Therefore, to improve the quality of human segmentation, a more aggressive denoising step is required, which means a larger processing time. For instance, in the tests of Figure 5.24 and Figure 5.25, the performance of fusion algorithm dropped from 30 Hz to 20 Hz. This caused that the robot did not respond effectively at faster huma motion.

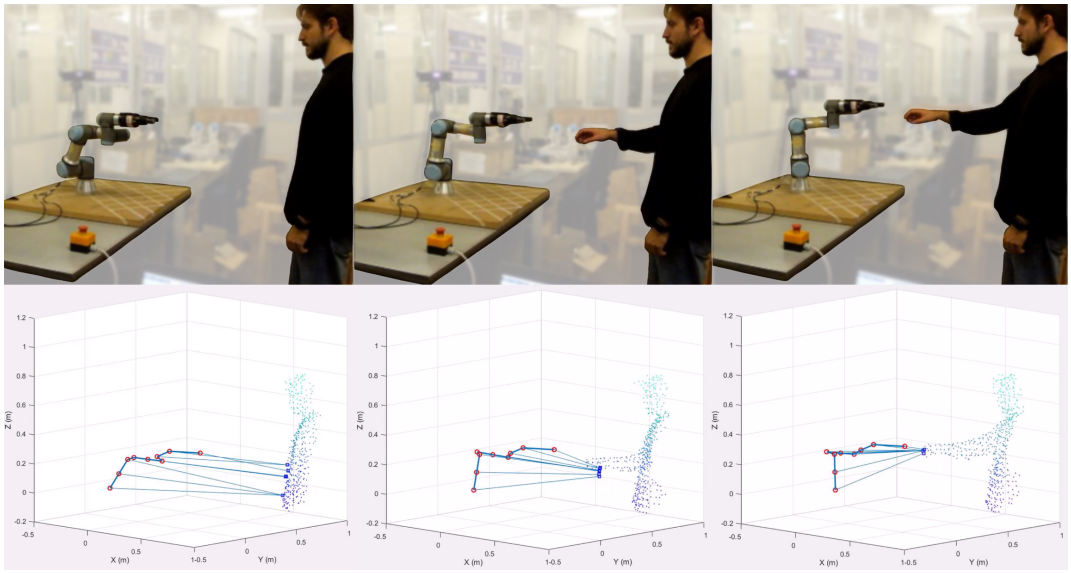


Figure 5.24 Frames of the experimental test of collision avoidance algorithm with the human point cloud approaching the robot TCP from the front

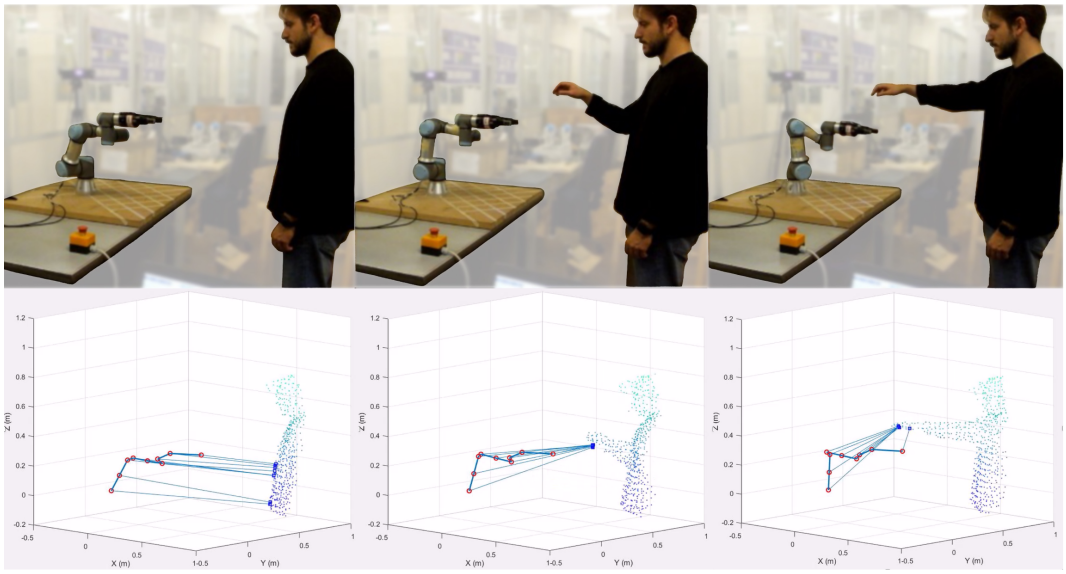


Figure 5.25 Frames of the experimental test of collision avoidance algorithm with the human point cloud approaching the robot wrist from the top

For safety concerns, further experiments will be addressed by future works. In fact, a more efficient implementation of the algorithm at a lower-level programming language is currently under development. The aim is to speed up calculation to keep the frequency at 30 Hz.

5.2.2.3 Discussion

The simulations have shown that the basic collision avoidance algorithm can be applied to the human point cloud with the same methodology used for the skeleton. The main difference involves the human tracking algorithm and the distance calculation. The *knnsearch* function is still a good way to calculate minimum distances, since it can deal with thousands of points within $2.5\ ms$. The results of simulation indicate that robot can safely avoid the point cloud and that the robot collision-free path is slightly affected by the point cloud density. Moreover, the possibility to use a convex hull of the point cloud has been investigated. The results have shown that the convex hull can be intended as a less accurate but more safe approach.

The preliminary experimental test has proven that the methodology is capable of dealing with a real robot with some limitations. The point cloud segmentation suffers the presence of the real robot and a stronger denoising step is required. The major drawback is that the algorithm slows down. This aspect will be addressed in future works, together with the experimental comparison between collision avoidance with skeleton, point cloud and convex hull.

5.2.3 MAP

5.2.3.1 Simulation

The analysis will focus on the position problem since it is produced by the MAP. Thus, the robot path will be discussed in depth. However, a graphical representation of the manipulator will be used to give an idea on the motion of the robot frame. For example, in Figure 5.26, the manipulator is depicted at different times; the initial pose is the most transparent, the actual pose has the maximum opacity and the desired pose is identified by $\boldsymbol{p}_f$ and the attached frame. In all tests, the robot starts with the x_e-axis perpendicular to the xy-plane and with the z_e-axis intersecting the z-axis. The subscript "e" indicates the frame centred in $\boldsymbol{p}_e$ that defines the pose of the end-effector. The final orientation is chosen by rotating about z through an angle δ. Concerning the simulation parameters, all the results are obtained by setting $v_{max} = 0.15\ m/s$, $a_{max} = 0.15\ m/s^2$, $\omega_{max} =$

$0.3\ rad/s$, $\psi_{max} = 0.15\ rad/s^2$ and $f_c = 60\ Hz$. These values are quite conservative and will produce a motion compatible with the joint limits of the manipulator. The MAP calculation time, which corresponds to calculate (3.43), is not discussed since it has the order of magnitude of $10^{-5}\ s$.

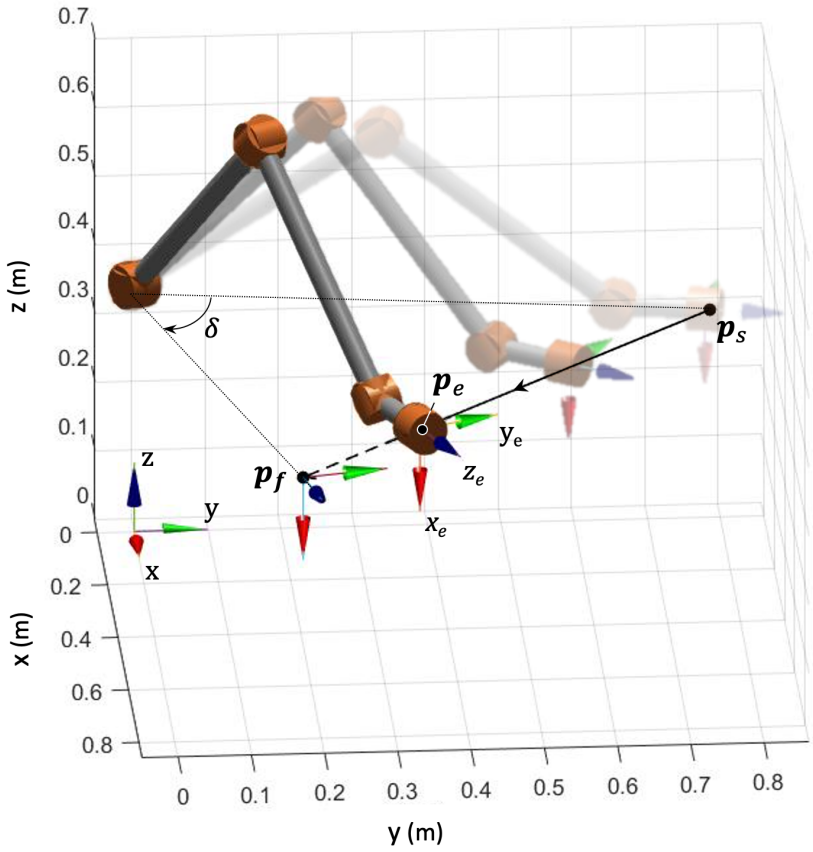


Figure 5.26 Matlab simulation environment built with the Corke Robotics Toolbox. The significant points refer to test 1. Without any obstacle or local attractor, the end effector would trace a rectilinear path since the gradient of U_f is radial

5.2.3.1.1 Single obstacle

The first test, named Test 1, is made considering a spherical steady obstacle and a local attractor placed on the side. This is basically the case of Figure 3.3b, which is the example used to introduce the MAP. This test is also utilized to show in practice the steps of the MAP algorithm.

By following Table 3.1 and Figure 3.16, the MAP algorithm of Test 1 is summarized in Table 5.4. Concerning the choice of the design parameters, all the test presented in the results section are carried out by considering $\sigma = 1$ for the

global attractor intensity; each obstacle is modelled with $\lambda_o = 0.2$, $s_\epsilon = 10^{-2}$, $\beta_o = 0.5$; each attractor is characterized by $\mu_a = 0.1$ and $\mu_\epsilon = 10^{-2}$. Notice that, for example, in test 1 it is $s_\epsilon^2 = 10^{-4} < 98.3 = \beta_o^2 \gamma_o / e$ and the condition mentioned in Appendix C is verified.

Table 5.4 MAP algorithm - Test 1

1	>> $\boldsymbol{p}_s = [0.1765\ 0.7735\ 0.37]^T m,\ \boldsymbol{p}_f = [0.7735\ 0.1765\ 0.37]^T m$
	>> $\sigma = 1$
2	>> steady obstacle
	>> $\boldsymbol{p}_o = (\boldsymbol{p}_s + \boldsymbol{p}_f)/2$, $R_o = 0.075\ m$
3	>> $\lambda_o = 0.2$, $s_\epsilon = 10^{-2}$ and $\beta_o = 0.5$
	>> $\gamma_o = 1069$, $R_o^* = 0.1286\ m$
4	>> $\boldsymbol{p}_a = [0.3689\ 0.3689\ 0.37]^T m$ (so that $\|\boldsymbol{p}_a - \boldsymbol{p}_o\| = 2R_o$)
5'	>> $\overline{\boldsymbol{p}_a \boldsymbol{p}_f} \cap U_{o,p}^* = 0$
6'	>> $R_a = 2R_o + R_o^*$
7	>> $\mu_a = 0.1$, $\mu_\epsilon = 10^{-2}$
	>> $\gamma_a = 98.43$, $R_a^* = 0.36\ m$
	>> $x_a' = 0.448\ m > R_a^*$
	>> $\tilde{\alpha}_a = 0.0564$, $\alpha_a = 0.99\tilde{\alpha}_a$

The shape of the resulting functions for the obstacle and the local attractor are the ones traced with the solid lines in Figure 3.6. In Figure 5.27 the resulting potential field $U_{t,MAP}(x, y, z)$ for $z = 0.37\ m$ is shown. The plane $z = 0.37\ m$ contains the starting position, the obstacle and the attractors. Therefore, this two-dimensional representation of $U_{t,MAP}$ has a significant value and can be used to figure out the resulting robot path.

The results of Test 1 are collected in Figure 5.28. For a comparison, the gradient vector field of the potential U_{fo} for $z = 0.37\ m$ is depicted in Figure 5.28a. Figure 5.28b shows the gradient vector field of the MAP at $z = 0.37\ m$. As can be seen, the gradient lines in proximity of the obstacle curve towards $\boldsymbol{p}_a$. Figure 5.28c and Figure 5.28d show the results in terms of the robot path, with

different starting position. In particular, $\boldsymbol{p}_{s1}$ is aligned with $\boldsymbol{p}_o$ and $\boldsymbol{p}_f$. In this case, thanks to the MAP, the robot can avoid the obstacle without suffering the classical saddle point presented in Figure 3.4a. Moreover, the starting positions $\boldsymbol{p}_{s2}$ and $\boldsymbol{p}_{s3}$ are not aligned with $\boldsymbol{p}_o$ and $\boldsymbol{p}_f$; by observing the gradient lines in Figure 5.28a, with these starting positions and in the absence of $\boldsymbol{p}_a$, the robot would avoid the obstacle on the opposite side. But the influence of the local attractor pushes the robot on the same side of $\boldsymbol{p}_a$.

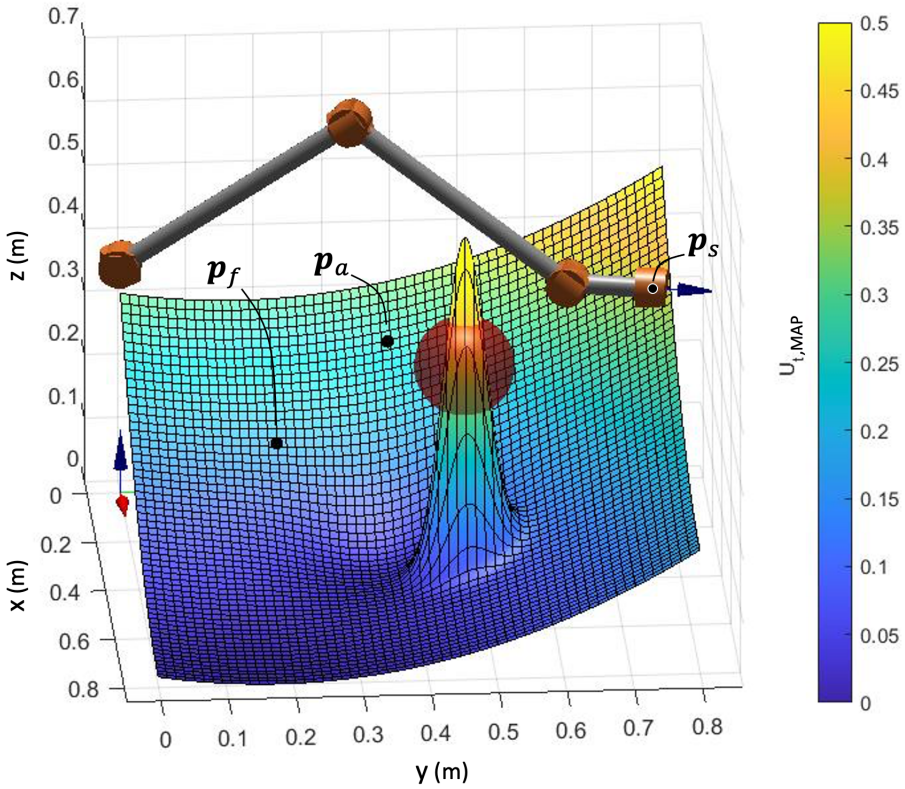


Figure 5.27 Simulation environment of Test 1 with the robot in starting position. The sphere has a radius R_o = 0.075 m and represents the obstacle. The $U_{t,MAP}(x, y, z)$ for z = 0.37 m is also plotted

In the second test, the same local attractor of Test 1 is displaced in different positions around the sphere to force the robot passing above and below the obstacle. In particular, $\boldsymbol{p}_a$ is placed so that $\|\boldsymbol{p}_a - \boldsymbol{p}_o\| = 2R_o$ and $x'_a = 0.448\ m$. Thus, the parameters for the local attractor and the obstacle are the same of the previous test. Figure 5.29 shows the results. As can be seen, in each case the MAP is able to condition the robot path.

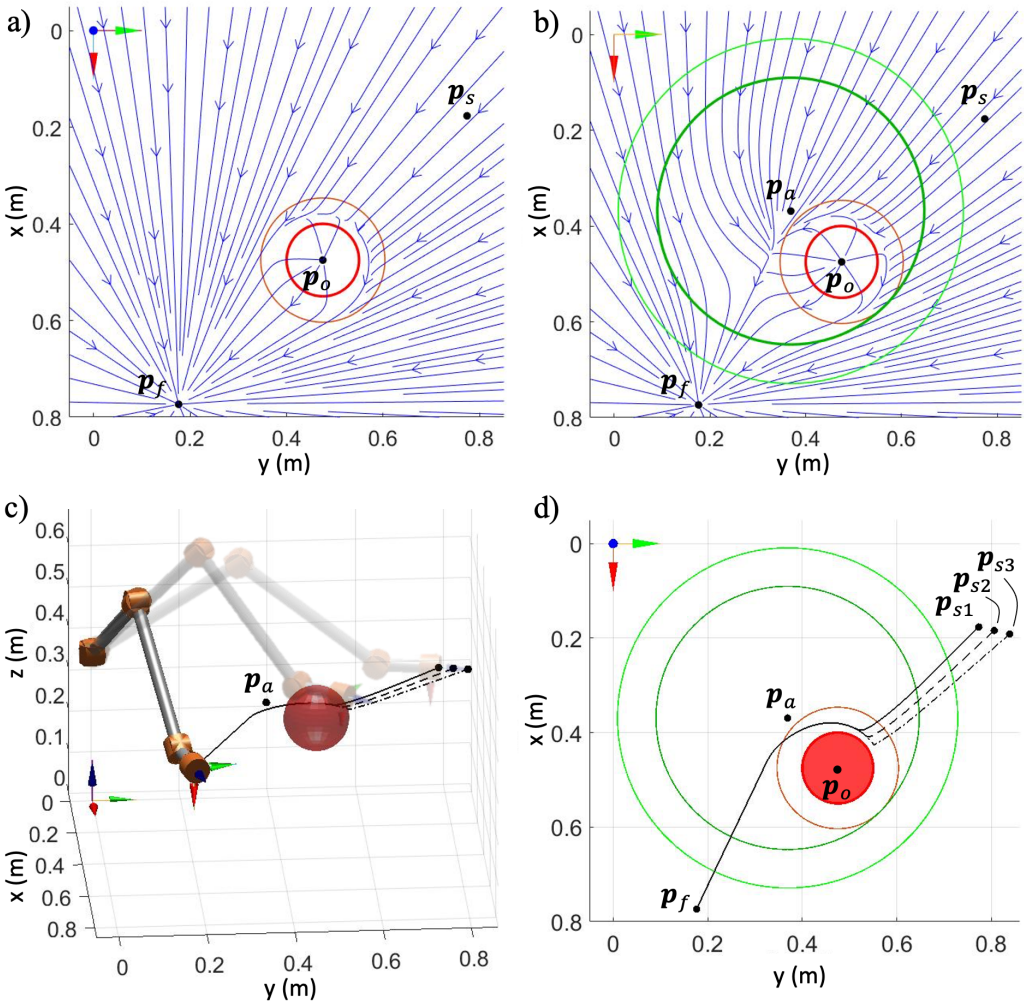


Figure 5.28 Results of Test 1. a) gradient lines of U_{fo} at $z = 0.37\ m$; the outer circle with centre $\boldsymbol{p}_o$ has radius R_o^*, while the inner one indicates the obstacle contour. b) Gradient lines of $U_{t,MAP}$ at $z = 0.37\ m$; the outer circle with centre $\boldsymbol{p}_a$ has radius R_a^*, while the inner one has radius R_a. c) The sphere represents the obstacle with radius R_o. The manipulator is depicted in 3 different poses to give an idea of the robot motion related to the solid line path; the different styles of the path lines correspond to three different starting positions. The robot frame rotates through $\delta = 60.3°$. d) Comparison of the robot paths. The top view is significant because the paths lay on the plane defined by $\boldsymbol{p}_s$, $\boldsymbol{p}_o$ and $\boldsymbol{p}_f$. The labels $\boldsymbol{p}_{s1}$, $\boldsymbol{p}_{s2}$ and $\boldsymbol{p}_{s3}$ identify the three different starting positions

The last test with single steady obstacle is performed to show how it is possible to avoid the obstacle using multiple exponential attractors. The obstacle has the same position and radius of Test 1 and Test 2. The starting point and the final position are kept the same as well. Two different attractors $\boldsymbol{p}_{a,1} = [0.2923\ 0.5172\ 0.37]^T m$ and $\boldsymbol{p}_{a,2} = [0.5811\ 0.3689\ 0.37]^T m$ are utilised to

generate the conditioned path. Since the segment $\overline{\boldsymbol{p}_{a,1}\boldsymbol{p}_f}$ crosses U_o^*, using (3.38)-(3.39) one can calculate $R_{a,lim} = 0.1317\ m$. In this example, $R_{a,1} = 0.13\ m$ is chosen and the constraints (3.40)-(3.41) are verified for the first attractor. In fact, it results $\gamma_{a,1} = 452$, $R_{a,1}^* = 0.168\ m$, $x'_{a,1} = 0.5895\ m$ and $\tilde{\alpha}_{a,1} = 0.042$. On the other hand, for $\boldsymbol{p}_{a,2}$, $\tilde{\varepsilon}_2 = 0$ since the segment $\overline{\boldsymbol{p}_{a,2}\boldsymbol{p}_f}$ does not intersect U_o^*; it is selected $R_{a,1} = 0.11\ m$. Then $\gamma_{a,2} = 631.3$, $R_{a,2}^* = 0.1422\ m$, $x'_{a,2} = 0.2721\ m$ and $\tilde{\alpha}_{a,2} = 0.0151$. Thus, equations (3.40)-(3.41) are verified for the second attractor. Finally, condition (3.42) which prevents the overlapping of $U_{a,1}^*$ and $U_{a,2}^*$ is also satisfied.

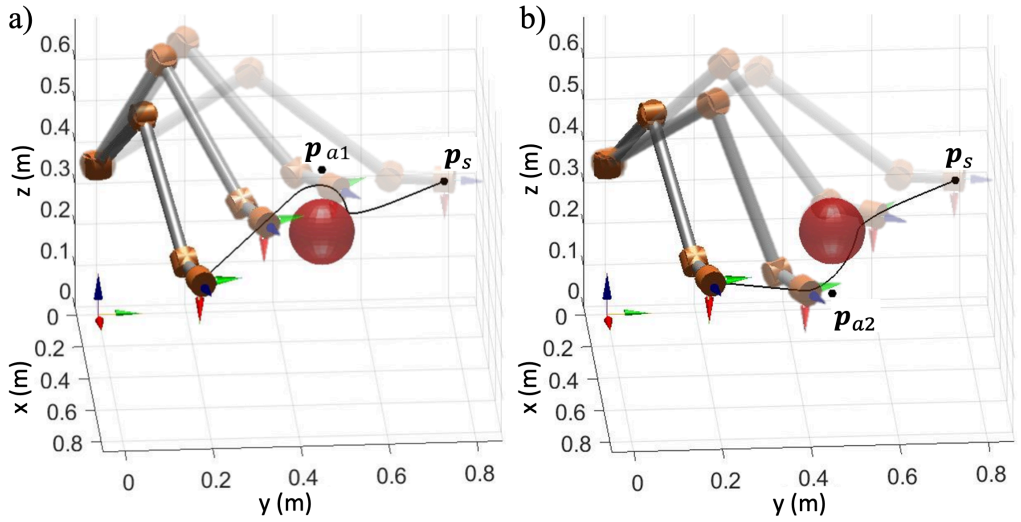


Figure 5.29 Results of Test 2. The manipulator is depicted in 4 different poses to give an idea of the robot motion related to the solid path line; the robot frame rotates through $\delta = 60.3°$. In a) and b) the local attractor is placed above and below the obstacle, respectively

In Figure 5.30a the resulting potential field for $z = 0.37\ m$ m is shown. In this test, the influence of α_a. Figure 5.30b shows the resulting paths of the robot for different pairs of $\alpha_{a,1}$ and $\alpha_{a,2}$. If they are chosen so that $\alpha_{a,1} = 0.99\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.99\tilde{\alpha}_{a,2}$ the effect of the attractors is maximized. But the path has a high curvature near the point where the saddle would show if $\alpha_{a,1} = \tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = \tilde{\alpha}_{a,2}$. This is not desirable. As discussed in Guldner and Utkin [65], in high curvature regions the real robot may require high torques, that could exceed the robot limits. The solid line in Figure 5.30b indicates the path for $\alpha_{a,1} = 0.8\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.6\tilde{\alpha}_{a,2}$. Therefore, by regulating α_a one can obtain smoother paths.

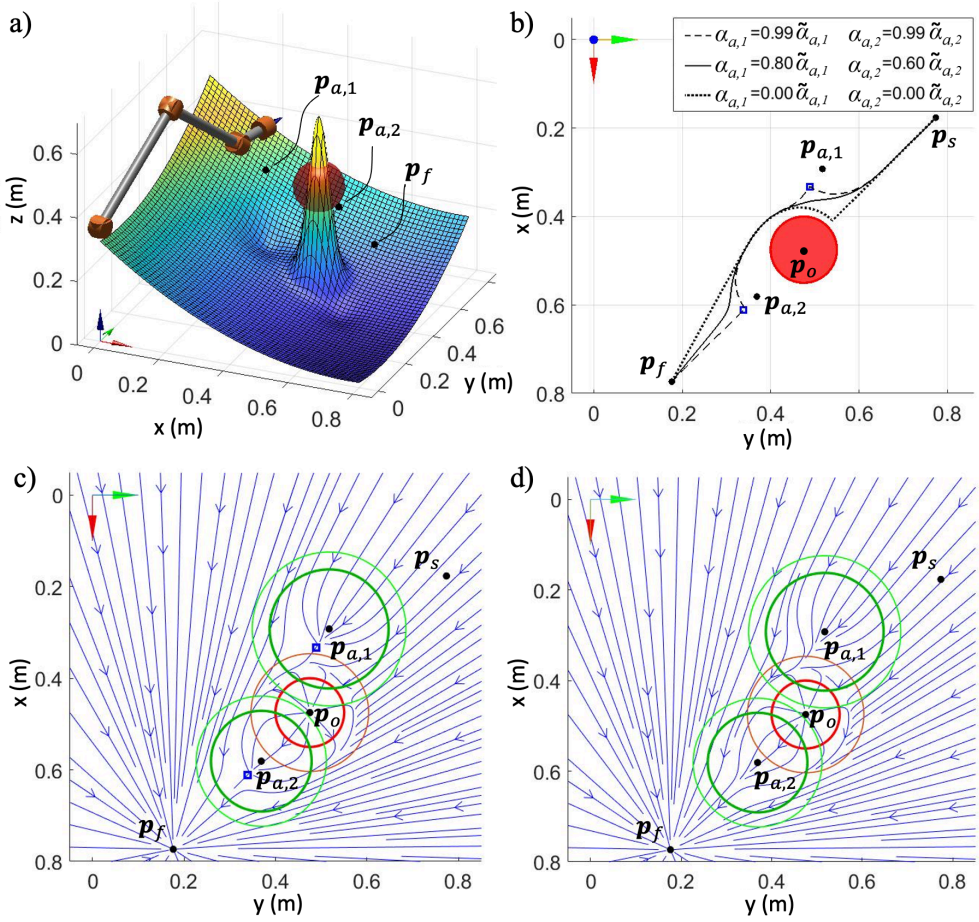


Figure 5.30 Results of Test 3. a) Simulation environment with the significant points and the manipulator in the starting position; in this test, the robot frame rotates through $\delta = 60.3°$. In the same graph, the MAP for $z = 0.37\ m$ is plotted to show the deflections obtained with $\alpha_{a,1} = 0.99\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.99\tilde{\alpha}_{a,2}$. b) Comparison of the robot paths for different values of $\alpha_{a,1}$ and $\alpha_{a,2}$; the squares indicate the saddle points related to the local attractors if $\alpha_{a,1} = \tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = \tilde{\alpha}_{a,2}$. c) MAP gradient lines for $z = 0.37\ m$, $\alpha_{a,1} = 0.99\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.99\tilde{\alpha}_{a,2}$. d) MAP gradient lines for $z = 0.37\ m$, $\alpha_{a,1} = 0.80\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.60\tilde{\alpha}_{a,2}$

5.2.3.1.2 Multiple obstacles

A fourth test is carried out to discuss the case with multiple obstacles. The robot starting and desired position are slightly different from the one of the previous tests, in order to have a wider movement of the robot. This allows to place more obstacles on the robot path still obtaining a motion within the robot joint limits. It is $\boldsymbol{p}_s = [0.1299\ 0.8201\ 0.37]^T m$, $\boldsymbol{p}_f = [0.8201\ 0.1299\ 0.37]^T m$and $\delta = 72°$. The geometry and the positions of the obstacles and the local attractors are summarized in Table 5.5, together with the tuned parameters. They are chosen so

that the robot avoids the obstacles in different manners. The attractor $\boldsymbol{p}_{a,1}$ forces the robot to pass above the obstacle $\boldsymbol{p}_{o,1}$; the attractor $\boldsymbol{p}_{a,2}$ guides the robot on the side of the obstacle $\boldsymbol{p}_{o,2}$; finally, attractor $\boldsymbol{p}_{a,3}$ steers the robot below the obstacle $\boldsymbol{p}_{o,3}$. Notice that for all the attractors it is the case $\tilde{\varepsilon} = \varepsilon$, thus the corresponding R_a it is limited by (3.39).

Table 5.5 MAP parameters and geometry - Test 4

p, i	**Obstacle**	**Attractor**
1	$\boldsymbol{p}_{o,1} = \boldsymbol{p}_s + \frac{2}{9}(\boldsymbol{p}_f - \boldsymbol{p}_s)$ $R_{o,1} = 0.05\ m$ $\gamma_{o,1} = 2.405 \cdot 10^3$ $\beta_{o,1} = 0.5$ $R^*_{o,1} = 0.0878\ m$	$\boldsymbol{p}_{a,1} = \begin{vmatrix} 2.032 \\ 7.468 \\ 4.228 \end{vmatrix} 10^{-1}\ m$ $R_{a,1} = 0.07\ m$ $\gamma_{a,1} = 1.559 \cdot 10^3$ $\alpha_{a,1} = 0.023$ $R^*_{a,1} = 0.0905\ m$
2	$\boldsymbol{p}_{o,2} = \frac{1}{2}(\boldsymbol{p}_s + \boldsymbol{p}_f) + \begin{vmatrix} 3 \\ 3 \\ 0 \end{vmatrix} 10^{-2}\ m$ $R_{o,2} = 0.08\ m$ $\gamma_{o,2} = 9.395 \cdot 10^2$ $\beta_{o,2} = 0.5$ $R^*_{o,2} = 0.1366\ m$	$\boldsymbol{p}_{a,2} = \begin{vmatrix} 3.265 \\ 5.7 \\ 3.7 \end{vmatrix} 10^{-1}\ m$ $R_{a,2} = 0.096\ m$ $\gamma_{a,2} = 8.288 \cdot 10^2$ $\alpha_{a,2} = 0.0323$ $R^*_{a,2} = 0.1241\ m$
3	$\boldsymbol{p}_{o,3} = \boldsymbol{p}_s + \frac{4}{5}(\boldsymbol{p}_f - \boldsymbol{p}_s)$ $R_{o,3} = 0.065\ m$ $\gamma_{o,3} = 1.423 \cdot 10^3$ $\beta_{o,3} = 0.5$ $R^*_{o,3} = 0.1124\ m$	$\boldsymbol{p}_{a,3} = \begin{vmatrix} 5.669 \\ 3.646 \\ 3.153 \end{vmatrix} 10^{-1}\ m$ $R_{a,3} = 0.1\ m$ $\gamma_{a,3} = 7.638 \cdot 10^2$ $\alpha_{a,3} = 0.0168$ $R^*_{a,3} = 0.1292\ m$

The resulting path is shown in Figure 5.31, from different views. For a comparison, the top and the side views also contain the standard path, i.e. the one obtained without the use of the attractors and with only the repulsive effects. As can be seen, the standard path is flat and lies on the plane $z = 0.37$ m. Here the robot passes all the obstacles on the same side (see the top view). On the contrary, the path obtained with the MAP is three-dimensional.

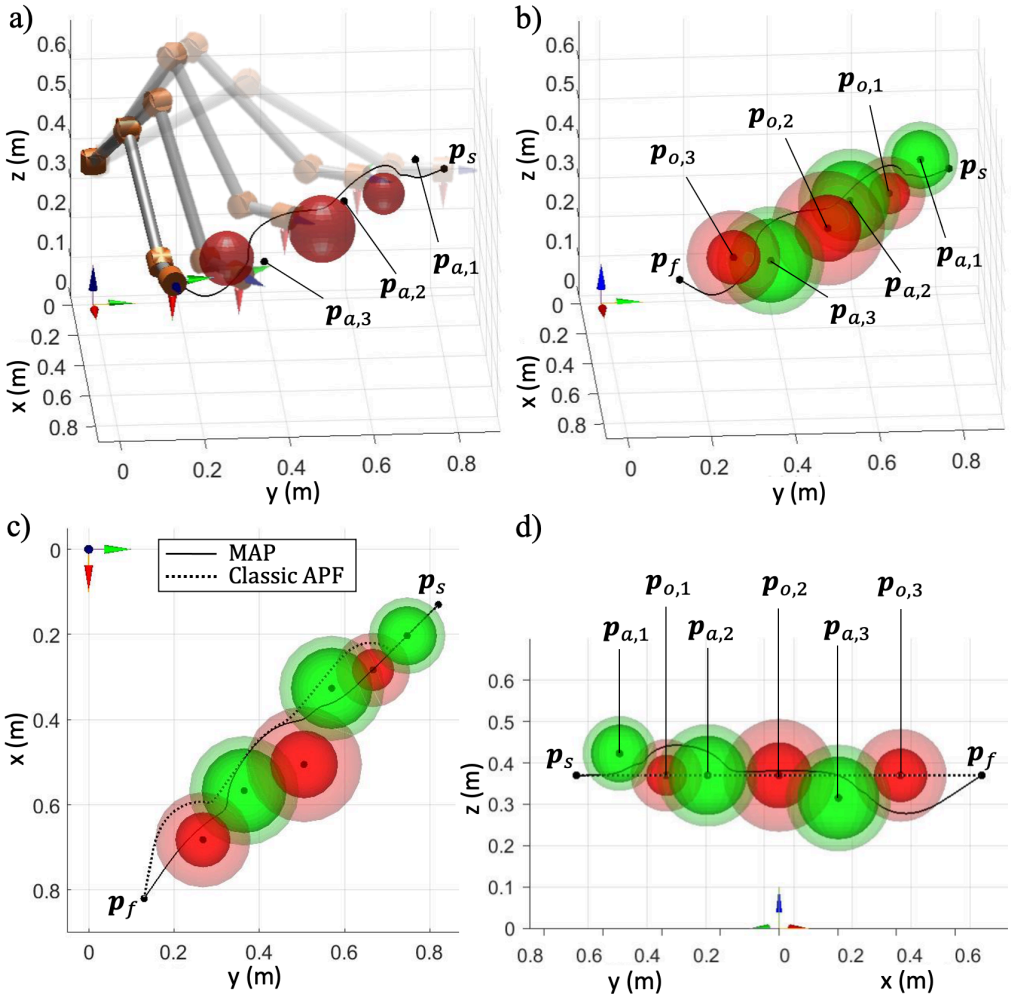


Figure 5.31 Results of Test 4. a) The manipulator is depicted in 5 different poses to give an idea of the robot motion related to the solid path line; the robot frame rotates through $\delta = 72°$. b) In addition to the robot path, the obstacles and the attractors are identified with the spheres of radius $R_{o,p}$, $R^*_{o,p}$, $R_{a,i}$ and $R^*_{a,i}$. c) - d) Comparison of the robot path obtained with the MAP with the one resulting without the MAP (named "Standard"), observed from the top and the lateral views

5.2.3.1.3 Dynamic obstacle

The last two tests, named Test 5 and Test 6, are carried out to show the case with the dynamic obstacle. Table 5.6 summarizes the dynamic MAP algorithm of Table 3.1 applied to Test 5. The case study of Test 5 is similar to the one described in Figure 3.12 and Figure 3.13. The local attractor is fixed to the obstacle frame $O'' - x''y''z''$ which in Figure 5.32 is indicated with red, green and blue arrows centred in $\boldsymbol{p}_o(t)$.

Table 5.6. Dynamic MAP algorithm - Test 5

1	>> $\boldsymbol{p}_s = [0.134\ 0.846\ 0.37]^T m,\ \boldsymbol{p}_f = [0.846\ 0.134\ 0.37]^T m$
	>> $\sigma = 1$
2	>> dynamic obstacle
	>> $\boldsymbol{p}_{o,t_0} = \boldsymbol{p}_s + (\boldsymbol{p}_f - \boldsymbol{p}_s)/3$, $R_o = 0.075\ m$
3	>> $\lambda_o = 0.2$, $s_\epsilon = 10^{-2}$ and $\beta_o = 0.5$
	>> $\gamma_o = 1069$, $R_o^* = 0.1286\ m$
4	>> $\boldsymbol{p}_{a,t_0} = [0.1951\ 0.6728\ 0.37]^T m,\ \boldsymbol{x}_a'' = [0\ 5/2R_o\ 0]^T$
5	>> $\overline{\boldsymbol{p}_{a,t_0}\boldsymbol{p}_f} \cap U_{o,p}^* \neq 0$
	>> $R_{a,lim} = 0.1317\ m$
6	>> $R_{a,t_0} = 0.99 R_{a,lim}$
7	>> $\mu_a = 0.1$, $\mu_\epsilon = 10^{-2}$
	>> $\gamma_{a,t_0} = 449.5,\ R_{a,t_0}^* = 0.1685\ m$
	>> $x_{a,t_0}' = 0.845\ m > R_{a,t_0}^*$
	>> $\tilde{\alpha}_{a,t_0} = 0.062,\ \alpha_{a,t_0} = 0.8\tilde{\alpha}_{a,t_0}$
8	>> *while* (3.41)(3.42)
	>> *if* $x_{a,t}' \geq R_{a,t_0}^*$
	>> $\gamma_{a,t} = \gamma_{a,t_0}$
	>> find $\tilde{\alpha}_{a,t}$ with (3.34), $\alpha_{a,t} = 0.8\tilde{\alpha}_{a,t}$
9	>> *elseif* $x_{a,t}' < R_{a,t_0}^*$
	>> find $\gamma_{a,t}$ by placing $R_{a,t}^* = x_{a,t}'$ in (3.23)
	>> find $\tilde{\alpha}_{a,t}$ with (3.34), $\alpha_{a,t} = 0.8\tilde{\alpha}_{a,t}$
	>> *end*
	>> *end.*

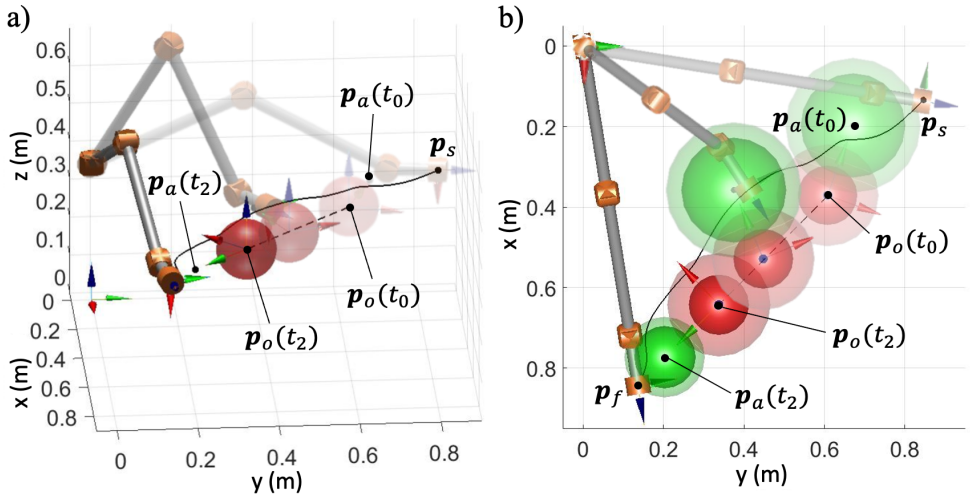


Figure 5.32 Results of Test 5. The solid line indicates the robot path, the dashed line represents the obstacle path. The manipulator and the obstacle are depicted in 3 different poses to give an idea their motion; the robot frame rotates through $\delta = 72°$ while the obstacle frame rotates through $155°$ about the z'' axis. a) and b) show the same motion from different views. In the top view, the active regions at significant times are also depicted

At time t_0 the obstacle is in position $\boldsymbol{p}_{o,t_0}$ with the obstacle frame oriented as in Figure 5.32. The position of the attractor in the world frame is $\boldsymbol{p}_{a,t_0}$ and the distance from the desired position is $x'_{a,t_0} = 0.845\ m$; this state is also indicated in Figure 3.13, where are pointed the corresponding γ_{a,t_0} and $\tilde{\alpha}_{a,t_0}$.

For $t_0 < t < t_2$ the obstacle moves with a rectilinear path (Figure 5.32); in addition, within the same time interval, it rotates through 155 degrees about the z''-axis. The obstacle stops in $\boldsymbol{p}_{o,t_2} = \boldsymbol{p}_s + 5/7\,(\boldsymbol{p}_f - \boldsymbol{p}_s)$ and the attractor terminates the rotation in $\boldsymbol{p}_{a,t_2} = [0.7726\ 0.2024\ 0.37]^T m$, at the distance $x'_{a,t_2} = 0.1\ m$ from $\boldsymbol{p}_f$.

During the motion, the attractor parameters $\gamma_{a,t}$ and $\tilde{\alpha}_{a,t}$ are dynamically adjusted according to the steps 8-9 in Table 5.6, that can be used to obtain the curves in Figure 3.13. Concerning the decay parameter, it goes from $\gamma_{a,t_0} = 449.5$ to $\gamma_{a,t_2} = 1269$, while the intensity varies between $\tilde{\alpha}_{a,t_0} = 0.062$ and $\tilde{\alpha}_{a,t_2} = 0.0032$. In Figure 5.32, the resulting robot path is planar, since the obstacle and the attractor move in the plane $z = 0.37\ m$.

The final test, identified as Test 6, is carried out with the same initial conditions of Test 5 but considering a 90 degrees rotation of the obstacle about the x'' axis within the time interval $t_0 < t < t_2$.

At the final time t_2, the attractor stops above the obstacle in $\boldsymbol{p}_{a,t_2} = [0.6393\ 0.3386\ 0.5575]^T\ m$ at the distance $x'_{a,t_2} = 0.3461\ m$ from $\boldsymbol{p}_f$. In this test, since $x'_{a,t_2} > R^*_{a,t_0}$, the decay parameter of the attractor keeps constant and equal to $\gamma_{a,t_0} = 449.5$, i.e. $R_{a,t}$ does not change as can be seen in Figure 5.33a. The intensity varies from $\tilde{\alpha}_{a,t_0} = 0.062$ and $\tilde{\alpha}_{a,t_2} = 0.0231$. In particular, for each $x'_{a,t}$, it is selected $\alpha_{a,t} = 0.8\ \tilde{\alpha}_{a,t}$.

Comparing the paths of Test 5 and Test 6 (Figure 5.33b), Test 6 produces a three-dimensional path, which derives from the different motion of the attractor. In fact, here the robot passes above the obstacle before reaching the goal.

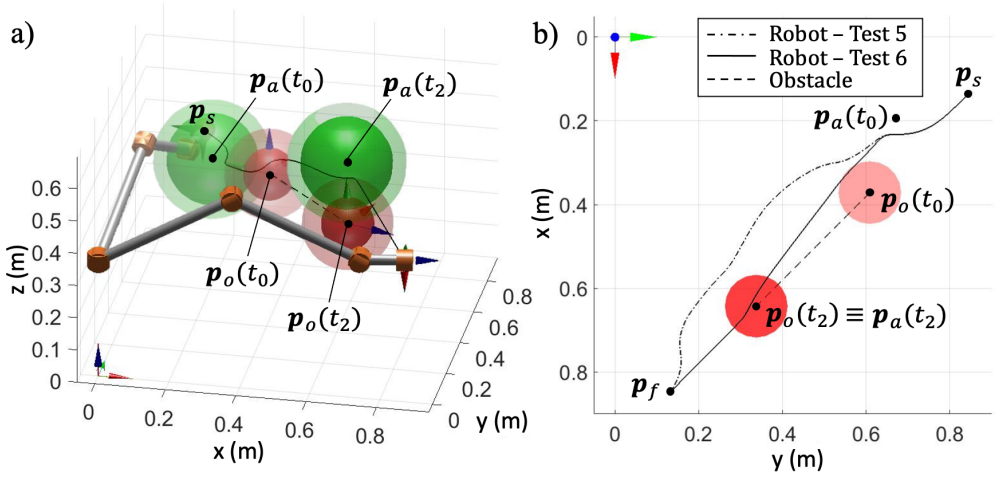


Figure 5.33 Results of test 6. a) The robot, the obstacle and the local attractor are depicted at the initial and final times; the robot frame rotates through $\delta = 72°$ while the obstacle frame rotates through 90° about the x'' axis. b) Robot path from the top view; in the same figure, the path of test 5 is also shown for comparison

5.2.3.2 Experimental tests

To validate the MAP, the algorithm has been applied to the robot UR3 in different scenarios. First, simple tests with a virtual steady obstacle have been carried out to verify that the robot responds as seen in the simulation environment. Furthermore, a final test considering the human hand as the dynamic obstacle is proposed to prove the effectiveness of the MAP for collaborative robotics.

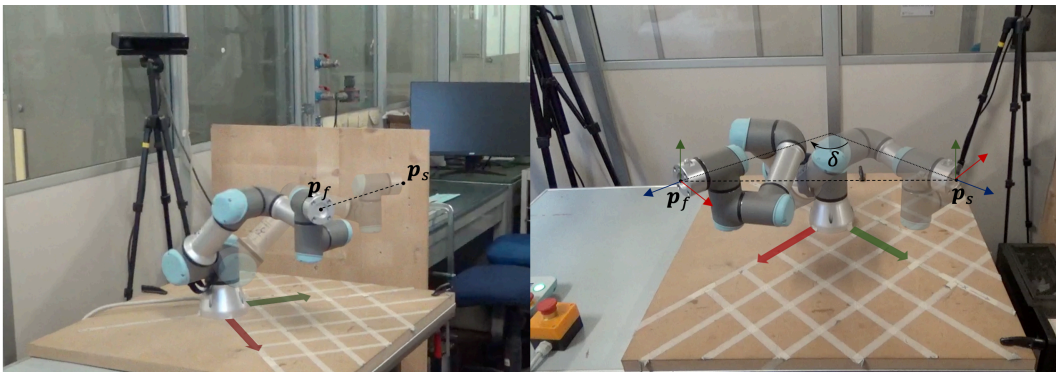


Figure 5.34 Experimental workbench. The robot is depicted with different opacity at the starting and final configurations. The world frame is also indicated, positioned at the robot base

In each test, the end-effector starts from $\boldsymbol{p}_s = [0.1133\ 0.5696\ 0.3381]^T m$ and navigates towards $\boldsymbol{p}_f = [0.5696\ 0.1133\ 0.3381]^T m$, while rotating about the z-axis through an angle $\delta = 90°$ (Figure 5.34).

5.2.3.2.1 Virtual steady obstacle

The first test is named Test 1e (where "e" stands for "experimental") and is similar to Test 1 presented in the simulation chapter. A virtual steady object with radius $R_o = 0.075\ m$ is placed at the middle of the starting and final end-effector positions (Figure 5.35). The term "virtual" indicates that the obstacle is not physical, but is considered in the Matlab environment. For instance, the control architecture is the same of Figure 5.6, except that the optimized skeleton signal is substituted by the spherical obstacle position and the distances are computed in Matlab without requiring any sensor. The MAP is built with the same parameters of Table 5.4, with different starting and final positions that have been modified according to the size of the UR3 workspace.

In Figure 5.36 the results of test 1e are shown, comparing the robot path obtained with the MAP and the classic APF. In this particular case, the paths are similar because local attractor $\boldsymbol{p}_a$ is placed at the same z coordinate of $\boldsymbol{p}_o$, $\boldsymbol{p}_s$ and $\boldsymbol{p}_f$. The MAP produces a smoother path (Figure 5.36c), while the classic APF path oscillates nearby the classic saddle point. The reason is the high curvature of the gradient lines of classic APF, as observed in the similar example of Figure 5.28. This prove that the MAP can prevent undesired robot motion due to the classic saddle point.

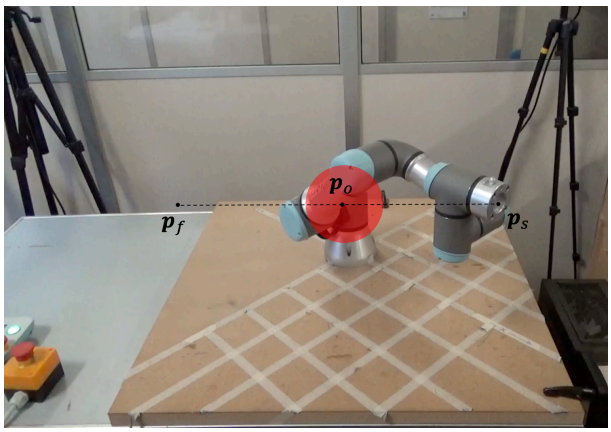


Figure 5.35 The red sphere is centered in p_o and represents the obstacle. The dashed line indicates the path that the end effector follows in the absence of the obstacle

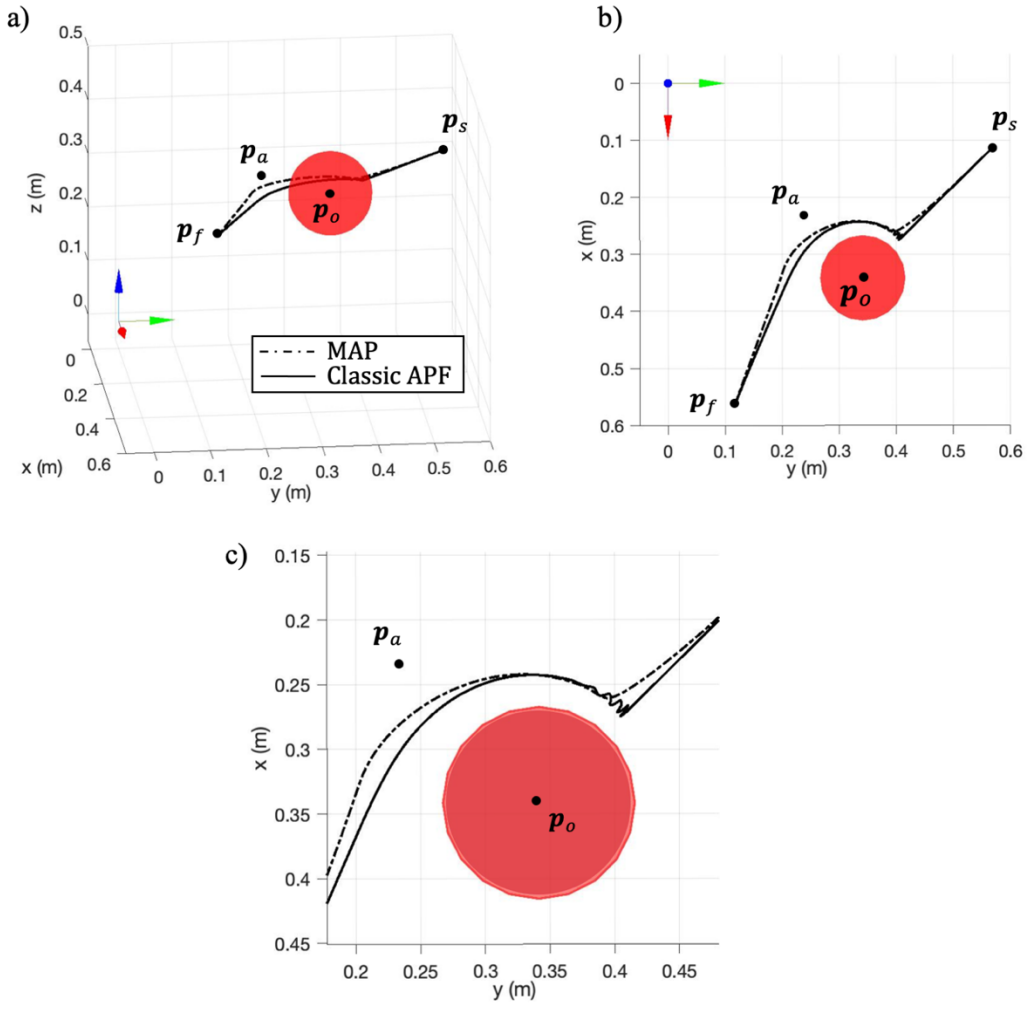


Figure 5.36 Results of Test 1e. a) and b) represent the same result from different viewing angles. c) detail of the top view

A second test, named Test 2e has been carried out by moving the local attractor $\boldsymbol{p}_{ai}$ in different positions around the obstacle (Figure 5.37). In each case, the local attractor conditionates the robot path, which bends precisely on the side of $\boldsymbol{p}_{ai}$ (see Figure 5.37c). In Figure 5.37 the robot path obtained by using the simulated model of UR3 is plotted. The paths resulting from simulation are almost identical to the ones measured in the experiments, so that no differences are appreciable in the figure. This proves the reliability of the simulation environment.

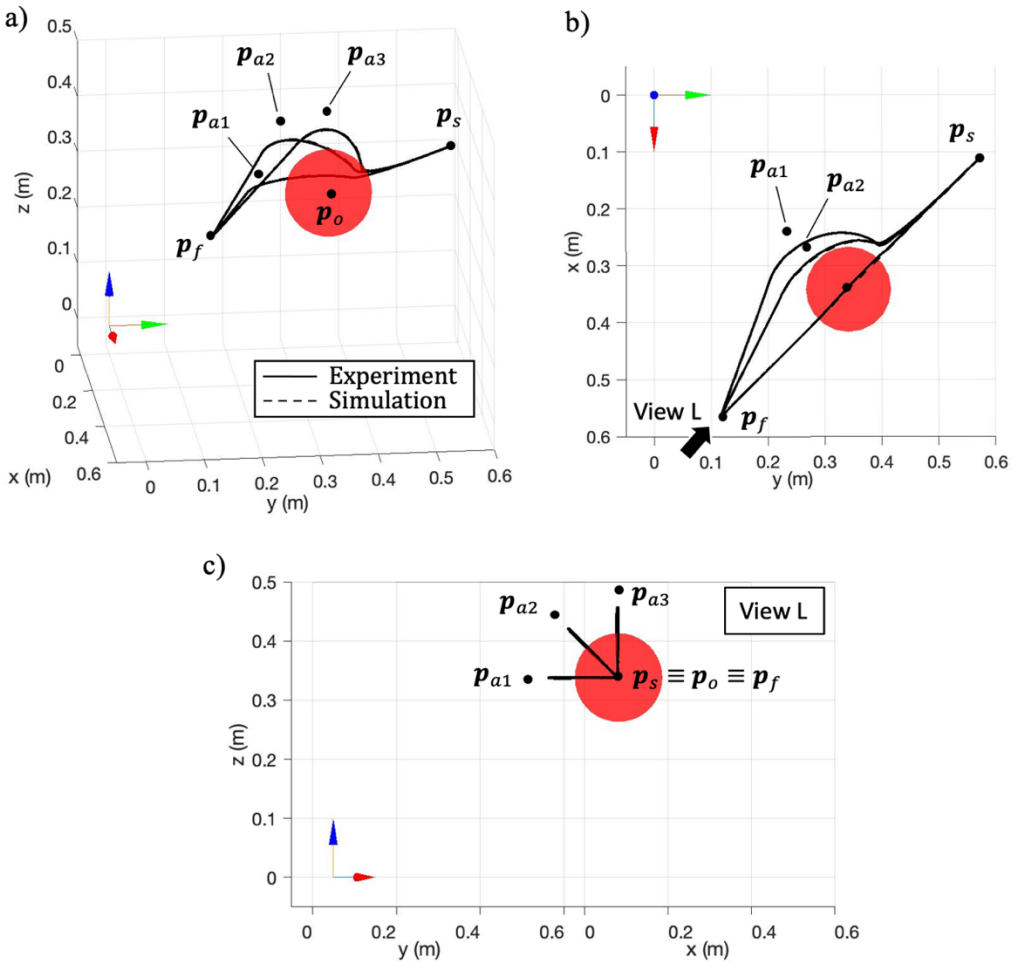


Figure 5.37 Results of Test 2e. a), b) and c) represent the same result from different viewing angles. In particular, c) is the lateral view, whose direction is identified by the arrow in b) and the label “View L”

The last test with virtual object is named Test 3e and considers the case with single obstacle and multiple attractors (Figure 5.38). The results confirm the observations made for Test 3 (Figure 5.30). However, experiments better show the effect of high curvature regions of gradient lines. In these regions, the robot oscillates due to dynamics. Smoother paths can be obtained by regulating the intensity of the local attractor, as can be seen from the curve $\alpha_{a,1} = 0.80\tilde{\alpha}_{a,1}$ and $\alpha_{a,2} = 0.60\tilde{\alpha}_{a,2}$.

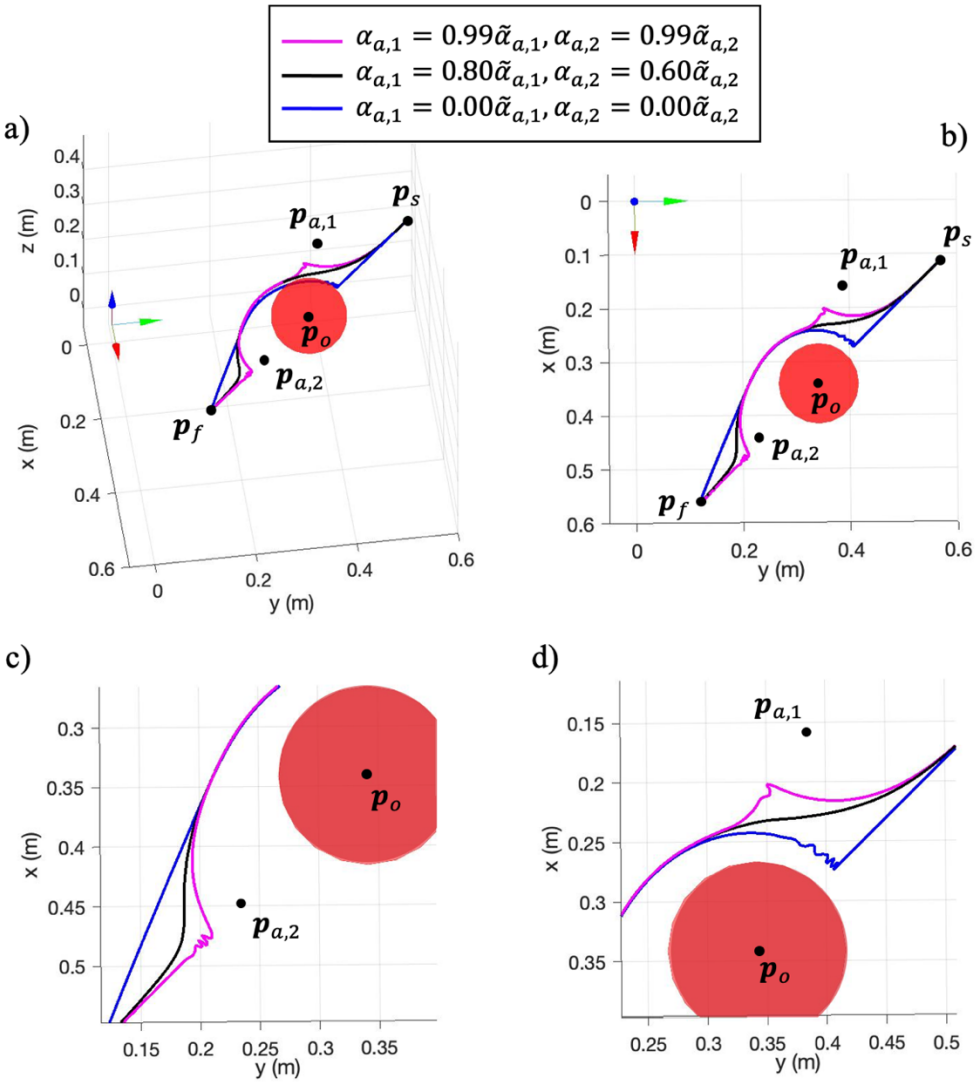


Figure 5.38 Results of Test 3e. a) and b) represent the same result from different viewing angles. c) and d) are details of the top view

5.2.3.2.2 Application to collaborative robotics

The MAP can be applied to collaborative robotics to influence the robot path while avoiding the human. In Figure 5.39, the robot starts its motion from the starting position and points towards the final configuration, as depicted in Figure 5.34. The operator is tracked by the 3D vision system as a skeleton. The human hand is approximated with a sphere with radius R_H centered at the optimized human hand position $\boldsymbol{p}_H^*$. A local attractor $\boldsymbol{p}_a$ is placed at a distance $2R_H$ from $\boldsymbol{p}_H^*$ and moves in order to keep the attraction in front of the hand (Figure 5.39). The human operator puts his hand within the end-effector actual and final configurations and the robot reacts with a collision-free path that crosses the hand on the front side.

The path is analyzed in Figure 5.40. In the same figure, the path resulting from classic APF in the same conditions is also reported. The classic APF path is obtained by saving human position data from the MAP test of Figure 5.39 and by replicating the test considering the same motion as input of the collision avoidance algorithm. Therefore, the paths of Figure 5.39 refer to the same human motion. Notice that the sphere (hand) and the attractor in Figure 5.39 slightly move during the robot motion, thus the picture is only representative of a generic frame. With the classic APF, the end-effector choose a path that passes above the sphere. This means that the human operator would have seen the robot crossing his hand from the top side. On the contrary, the MAP is able to keep the robot path in front of the hand, for a more comfortable collision avoidance.

In general, the possibility to drive the robot towards desired regions nearby any human body part make the collision-free path predictable. In the proposed example, the human worker knows a priori that the robot would choose the path in front of the hand, regardless of the robot approaching direction. In Figure 5.41 the results of a test similar to Figure 5.40 are reported. The robot paths of Figure 5.41 are obtained translating human data of the previous test by $0.1\ m$ along the z-axis. In this way, the robot would have approached the hand from below. However, the MAP would have still driven the end-effector on the front side of the hand.

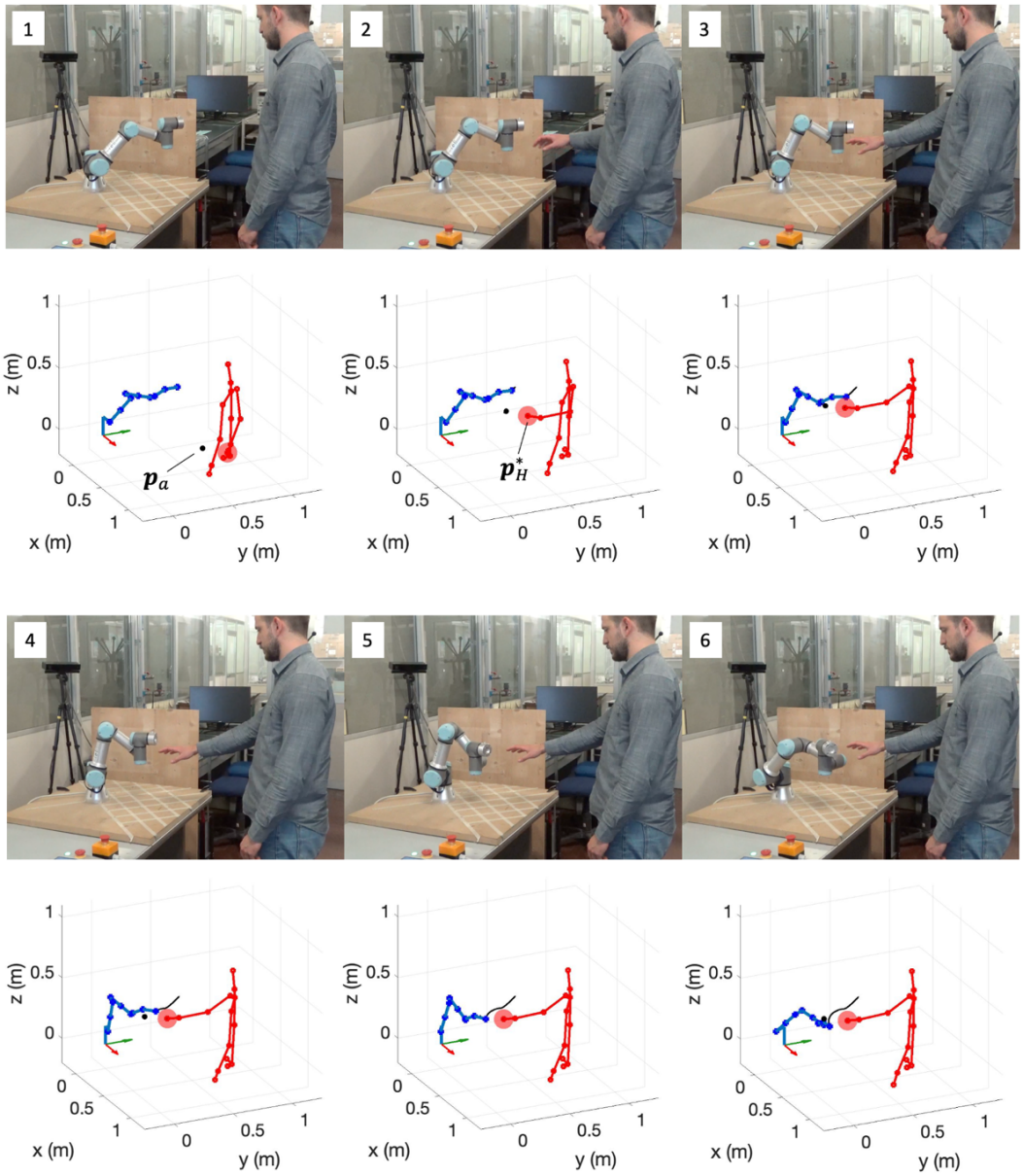


Figure 5.39 Frames of the experimental test of the MAP with the obstacle centered at the right hand joint p_H^* of the optimized skeleton. The attractor moves with p_H^* and is indicated with a black dot in the pictures of the Matlab calculation environment, where the collision-free path of the end-effector is also plotted

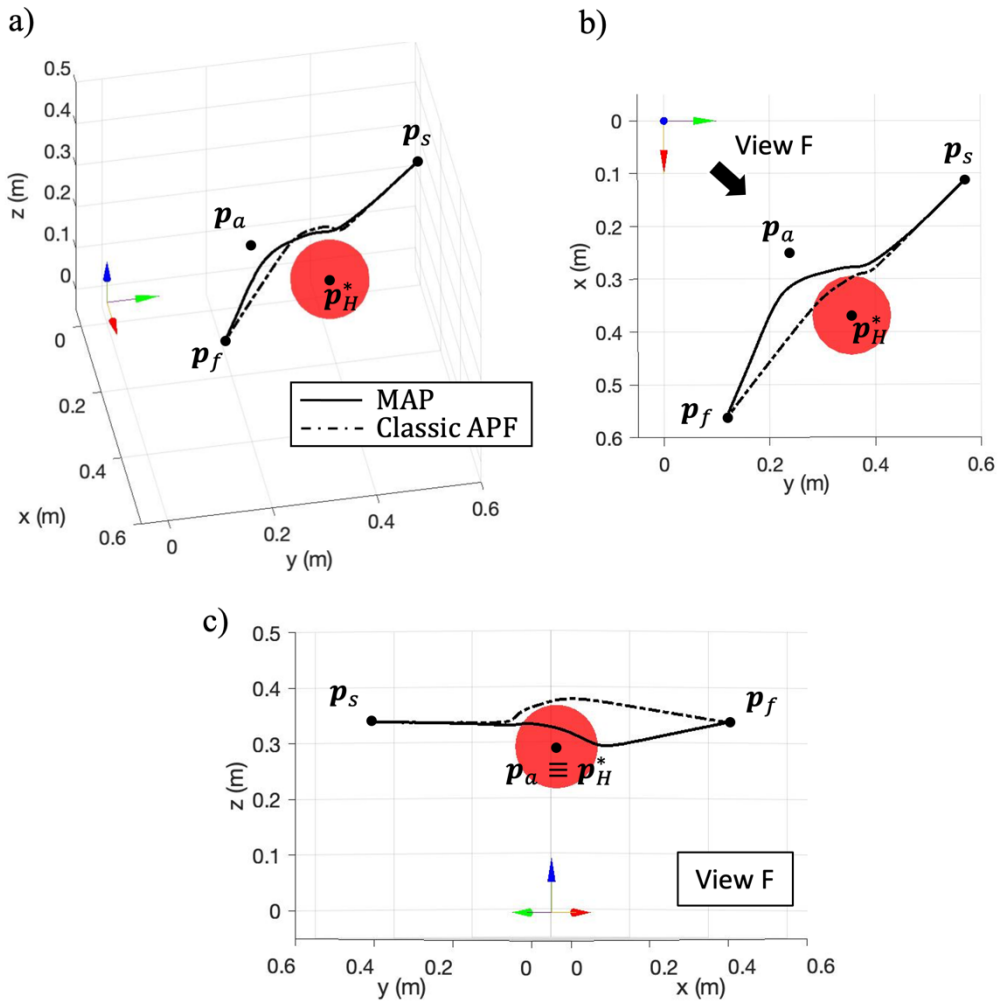


Figure 5.40 Results of the test reported by the frames in Figure 5.39. a), b) and c) represent the same result from different viewing angles. In particular, c) is the front view, whose direction is identified by the arrow in b) and the label “View F”

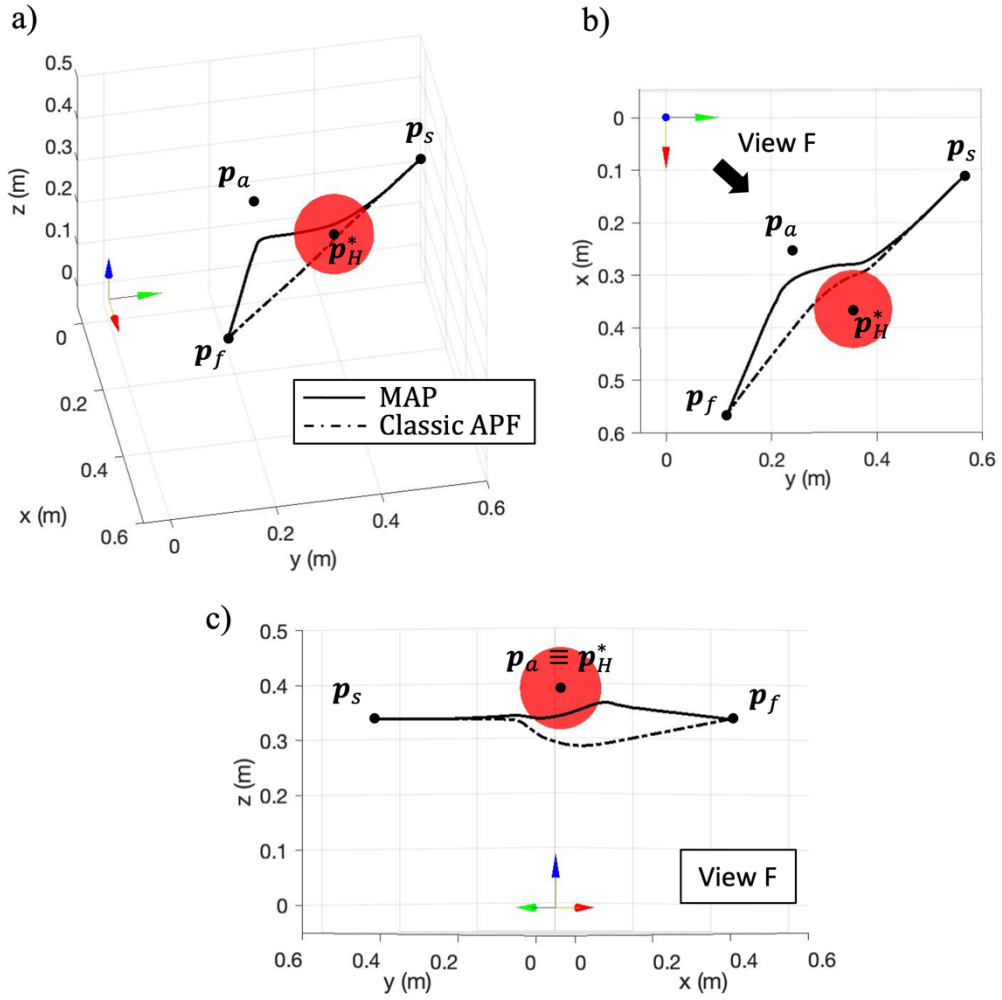


Figure 5.41 Results obtained by translating the obstacle (skeleton) by $0.1\ m$ along the z-axis. a), b) and c) represent the same result from different viewing angles. In particular, c) is the front view, whose direction is identified by the arrow in b) and the label "View F"

5.2.3.2 Discussion

The MAP theory has been described and the results proved that it is a promising method to perform collision avoidance by driving the robot through preferred regions. The method combines the well-known approach of artificial potential field to the use of elementary functions to handle obstacle avoidance with original form. The strengths of the MAP are summarized in the following points:

1. It is a potential field and benefits of all the advantages of the robot navigation algorithms based on artificial potential fields, such as easiness, low computation time and wide applicability.

2. The maximum admissible deflection nearby the local attractor can effectively bend the gradient lines of the potential field.
3. The local minima related to the local attractor are prevented and the global minimum is unperturbed.
4. The high curvature regions of the gradient lines can be adjusted by acting on the single parameter α_a.
5. By opportunely placing the local attractor, the problem of the classical saddle point due to the obstacle repulsive potential can be solved.
6. The solution does not depend on the potential field related to the obstacle and it can be extended to more complex repulsive potential fields.
7. The exact solution is provided in closed formula, so that the potential field can be sculpted by simple calculation of the optimal parameters.
8. Because of the previous point, the MAP can be adjusted dynamically. In practice this requires the only knowledge of the distance x'_a.
9. The MAP can be applied in three-dimensions, with any number of obstacles or local attractors.
10. Combining multiple local attractors, it is possible to avoid collision with complex three-dimensional trajectories.

Concerning the limitations, two points can be discussed. The first point is that the MAP stands on assumptions (3.40),(3.41),(3.42) so that the elements must be combined according to geometrical constraints; this can be intended as a weakness, since one cannot combine the elements at will. For this reason, the method to design the MAP has been described step by step and the main phases have been collected in a unique algorithm to deal with static and dynamic MAP. At the end, when it comes to the practice, this aspect reduces to simple math calculation. The second point is related to the previous one in addition to the fact that the maximum intensity of a local attractor is limited by $\tilde{\alpha}_a$. So, one can observe that not only the geometry of the MAP is constrained, but also the effect of a local attractor is limited. Even if this point is a theoretical limitation, the results show that in practice the MAP has tangible effects so that it can deal with different scenarios, even considering complex cases with multiple obstacles and attractors, or with dynamic obstacles.

Finally, one more aspect can be discussed to address further studies. By choosing the local attractor position $\boldsymbol{p}_a$ as a design variable, i.e. as the point the

robot should be attracted by, a slight error is admitted. In fact, the theory reveals that the attraction is towards the saddle point $\tilde{\boldsymbol{p}}$, which does not coincide exactly with $\boldsymbol{p}_a$. To attract the robot within a certain region, this aspect is not relevant as shown in the results section. But if one wants the robot to navigate exactly towards a certain local attractive point, the design phase of the MAP can be revised to consider $\tilde{\boldsymbol{p}}$ as the design variable, instead of $\boldsymbol{p}_a$.

5.3 Hand-Over

5.3.1 Experimental test

Two different kinds of tests have been conducted to verify the effectiveness of the hand-over control strategy. The first category of tests, named "Prediction" tests, has been performed to show the differences between the hand-over approach with the prediction scheme and the one without prediction. In the second test case, named "TCP pose" tests, the possible advantages of considering the pose of the TCP instead of the only positions have been studied.

5.3.1.1 Prediction test

In this kind of tests, the operator moves his hand inside the robotic workspace, and the robot starts to move. The object exchange is performed when the hand of the worker and the TCP of the robot are inside the meeting volume, as schematized in Figure 4.5.

The robot is driven both with the information related to the virtual hand pose obtained considering or not the prediction. In the former test, the hand-over is performed without prediction and the Kinects raw skeleton data are saved. Then, for a reliable comparison, the same movement of the worker has been used in the test with prediction. In particular, the saved skeleton data are processed, and the command joints velocity set, calculated from the trajectory planning relative to the predicted hand, is sent to the robot. This permitted to replicate the same test conditions.

In Figure 5.42, an example of frames of the movements that the operator performed are shown. In Figure 5.43, the distance between the hand and the shoulder center and the norm of the velocity of the TCP in both cases are presented. By observing the graphs, the shape of the predicted and the standard curves are quite similar, only shifted in time. This proves that the proposed Kalman filter based on Wiener process has a significant practical effect in this application. In the case with the prediction, the virtual hand entered in the robot

workspace $0.25\ s$ before the case without the velocity calculation. This implies that the robot started moving earlier with the velocity calculation, as can be seen in Figure 5.43. The virtual hand stopped its motion after $7\ s$. After the stop, the operator waited the arriving of the robot to perform the object exchange. The waiting time was $0.93\ s$ with the prediction and $1.17\ s$ without. In fact, Figure 5.43 shows that the TCP of the robot finished to move at $7.93\ s$ with the velocity calculation and at $8.17\ s$ without it. The results show clearly that the waiting time (w.t. in the Figure 5.43) is reduced with the prediction of the virtual hand position. This is an interesting and promising result and shows that the approach with the velocity calculation is effective and reduces the waiting time of the worker, obtaining a fast hand-over operation.

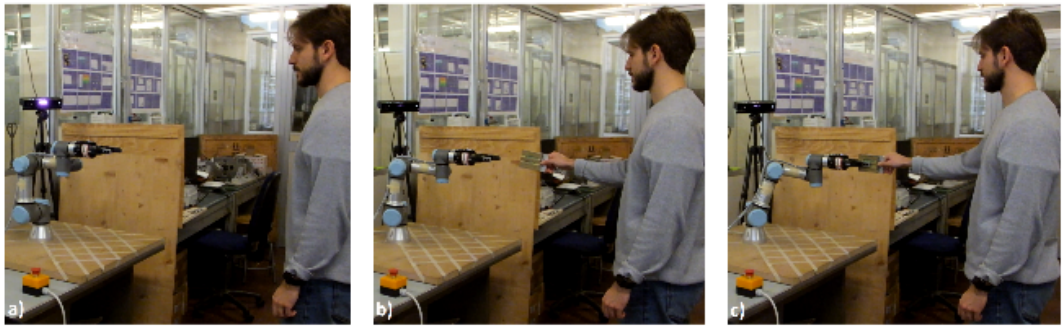


Figure 5.42 a) The hand of the operator is outside the workspace of the robot, and the manipulator is in its home position; b) the hand enters in the workspace, and the robot starts to move; c) the TCP is inside the meeting volume and the hand-over is performed

5.3.1.2 TCP pose test

In this subsection, the results of tests show the ability of the robot to adapt to the pose of the hand and how this adaptability is useful for a fluent and ergonomic human– robot hand-over. Human and robot have to pass each other a plate which is exchanged with a casual orientation.

In the first test, the worker changed the orientation of the forearm while the TCP of the robot was inside the meeting volume. In Figure 5.44, it is possible to see how the robot can modify the TCP pose according to the pose of the human-hand.

The second test highlighted the differences between a hand-over where only the hand position was used to plan the trajectory and one where the pose information was processed. In Figure 5.45, the frames of the hand-position case are shown. It is clear how the operator must adapt his arm in order to give/take the plate with the proper orientation.

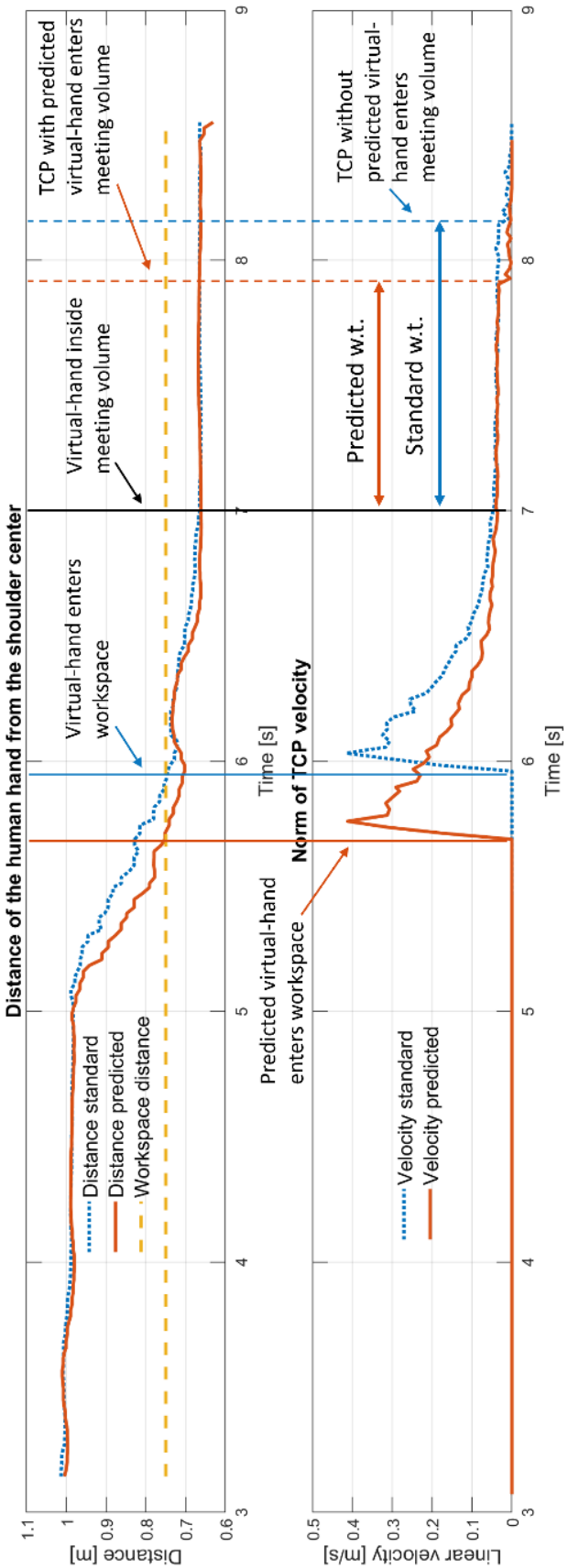


Figure 5.43 Norm of the distance between the virtual hand and the shoulder center and norm of the TCP linear velocity vector. The solid lines are the cases with predicted position; the dotted lines are the standard cases

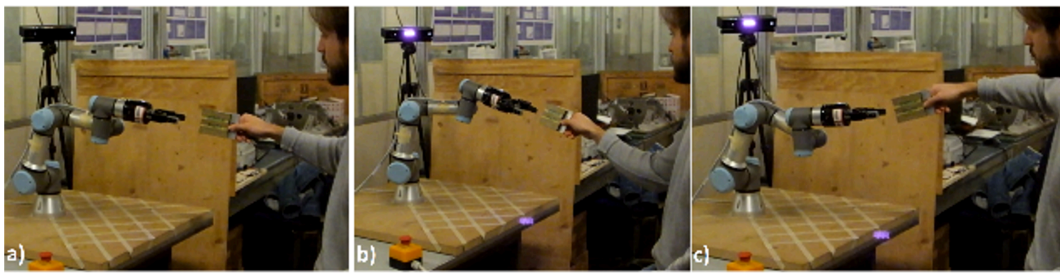


Figure 5.44 The robot is able to modify its pose in order to align its z-axis with the forearm axis

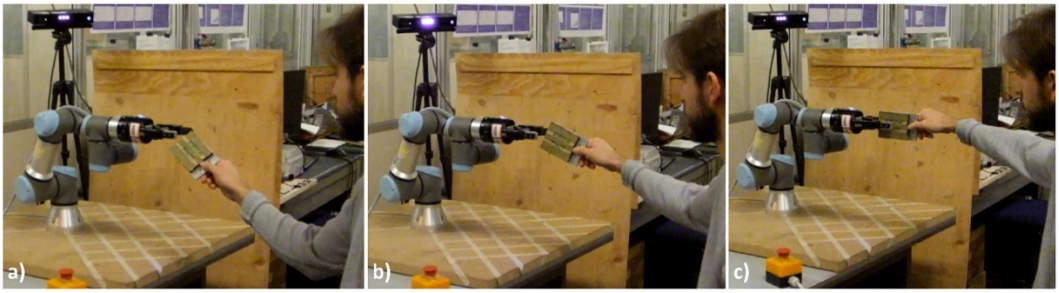


Figure 5.45 a) The TCP of the robot is inside the meeting volume and stops to move, but its orientation does not permit it to take the plate with the proper orientation; b)-c) the operator has to move his arm to perform the hand-over with the desired orientation

Figure 5.46 The operator moves his hand and the robot starts its motion; b)-c) the hand-over is achieved with the desired orientation of the plate, and the operator does not move the arm to adapt the plate pose

The frames of the test conducted with the pose of the hand are presented in Figure 5.46. In this case, the worker can give/take the plate with different poses of the hand while the plate is exchanged with the desired orientation.

5.3.1.3 Discussion

The experimental tests presented in the previous section have been conducted to evaluate the performances of the proposed path planner algorithm in terms of ability to follow the pose of moving targets and times to perform hand-over tasks. The “TCP pose” tests showed that controlling the robot in order to adapt the pose

of the TCP to the one of the hand permits the operator to choose the preferred arm posture to perform the hand-over. This is not possible using algorithms that consider only the position of the hand, as other studies [100], [106], [107]. In fact, considering only the hand position, the human worker must change the pose of the hand in order to perform the hand-over with the proper orientation of the object to exchange. Furthermore, the good fluidity with which the robot controlled by the proposed algorithm is able to follow the orientation of the hand can be seen on the video attached to this work.

The results of the "Prediction" tests showed clearly that the possibility to predict the human hand position permits to obtain faster hand-over tasks reducing the waiting time for the operator. The solution presented in this work achieves a mean value of the waiting time of 1 s. The control strategies proposed in [102], [105] that have the same characteristics of the algorithm proposed in this work (bidirectionality, reactivity, ability to follow the pose of the hand and possibility to consider the dimensions of the object to exchange) have mean values of the waiting time of $4\ s$ and $1.4\ s$ respectively. So the algorithm here proposed has shorter waiting times than other hand-over control strategies.

Chapter 6

Conclusions and future works

In this work, different tools and algorithms to deal with the most demanding collaborative robotics application have been presented. The desire of a safe, responsive and human-friendly robot behavior translated into technical challenges. They consisted in combining human tracking devices and real-time algorithms to let the robot accomplish the task with a view on human preferences.

The importance of human tracking has been discussed in a dedicated chapter. The study and the trade-off that led to the choice of the Kinect camera has been documented. The Microsoft sensor showed the best performances among depth cameras on the market. At low price, it offers built in skeleton tracking and compatibility with most of the existing libraries for computer vision.

To improve Kinect capabilities, the attention has moved to the multiple sensor layout and the fusion algorithms. The steps for skeleton optimization and point cloud processing have been presented. The optimized skeleton recognizes the human body parts approximating the human size with joints. The merged point cloud better detects the human shape but suffers large data processing. Regardless of the type of human data, the human tracking algorithm can be synthesized in a calculation block that gives human position in 3D coordinates. This information is exploited by the robot control algorithms depending on the application.

Two main topics have been discussed: collision avoidance and hand-over. Different algorithms have been developed to obtain a natural and seamless collaboration in these two scenarios. The algorithms use human and robot relative

position to calculate robot commands in terms of joint velocities. The innovation and the theory behind the algorithms have been discussed in detail in dedicated chapters.

The collision avoidance algorithm based on repulsive velocities does not present important changes with respect to the previous works. However, the method has been described to understand the control strategy that drive the collaborative assembly task in the application chapter.

The MAP is the major contribution of the dissertation. It presents important novelties with respect to the state of the art. The method is inspired by the previous studies on the artificial potential fields but moves the attention on the role of the attractors rather than the obstacles. By considering some examples of robotics applications, the importance of choosing preferred directions while avoiding collision emerges. Since the artificial potential field techniques turned out to be simple and effective in the previous works, the intuition has suggested to explore this branch. The state of the art revealed a lack of works dealing with multiple attractors. Thus, the idea of a new formula to handle attractors and repulsors coexistence has been developed.

The hand-over chapter presented an improved algorithm with human hand motion prediction. The method uses Kalman filter and assumes the hand motion as a Wiener process. The hand position at the time horizon of two samples is estimated by propagating the prediction stage of the Kalman filter. This scheme has been integrated with the results of the previous work to obtain an ergonomic robot motion.

The second part of the dissertation show the applications. Depending on the grade of novelty, the developed algorithms have been verified either by simulation or experimental test. Simulations run in Matlab by considering the kinematic model of two among the most popular collaborative robots, i.e. the LBR iiwa 14 by KUKA and the UR3 by Universal robots. Experimental tests are carried out with a real robot UR3 driven by the 3D vision system. The hardware set-up of the laboratory robotic cell has been described. Two computers are connected to the Microsoft Kinect v2 cameras to obtain the data acquired by each sensor. The third computer is connected to the controller of the UR3 robot and run the control algorithms.

The first application investigates the feasibility and the potential benefits of a collaborative assembly cell characterized by the highest level of collaboration. A representative assembly task which can take advantage from collaborative

robotics has been chosen. Robot and operator carry out their task in parallel, with the possibility to operate at the same time in the shared workspace. The robot is controlled with collision avoidance based on repulsive velocities. The algorithm has a low computational cost and can drive the robotic arm away from the human body represented as a skeleton. The discussion has shown the capabilities of the proposed algorithm by analyzing the robot alternative trajectory. The robot is able to avoid the human and return on the planned path to carry on the task. Moreover, the cycle time has been compared for the cases of responsive and sequential collaboration. The highest level of collaboration, which has been tested in this work, promises significant improvement in terms of task time, even if the actual possible benefit should be carefully evaluated for each specific application.

The second application investigates the use of the point cloud to represent the human size. Simulations have shown that using relatively cheap hardware and useable software, it is possible to run the whole algorithm for human detection and collision avoidance at a frequency above $30\ Hz$. Calculations are made properly combining custom functions and built-in Matlab tools; thus, the method is easily replicable and can be useful for the preliminary study of collision avoidance with point cloud. A comparison between the optimized point cloud and the resulting convex hull has been proposed, to verify the algorithm and to highlight different robot behaviors. An experimental test has been reported, but the presence of a physical robot resulted in a noisier point cloud. A stronger donoising was required and the processing time dropped to $20\ Hz$. A more efficient implementation with low level programming is currently under development.

The MAP application has been structured to show the algorithm acting in different scenarios. The simulation environment has been chosen to investigate the influence of the main parameters that regulate the MAP. The simplest case with single obstacle and local attractor revealed that the robot can avoid the obstacle choosing the local attractor side, even considering different approaching directions. The case with multiple local attractors and single obstacles has been considered to discuss the possibility to obtain smooth trajectories by regulating the attractive effect. The MAP has revealed also effective in the test with multiple obstacles and local attractors, with the possibility to obtain more complex collision avoidance paths. The tests with a dynamic obstacle showed how it is possible, in practice, to deal with moving objects. To verify that the real robot can follow the gradient lines of the MAP, the main simulation tests have been replicated with the robotic cell. The experimental results confirmed the robot

motion expected from simulation. Moreover, an application of the MAP to collaborative robotics has been analyzed. A local attractor, placed in front of the human hand, can effectively bend the robot path on the same side. The discussion outlines the strengths of the proposed method and analyses the limitations, with a positive verdict justified by the results. Future works will focus on the extension of the MAP so that not only the end-effector, but also the entire robot body is affected by obstacles and attractors.

The improved hand-over algorithm has been applied to a hand-over task where robot and worker can be either giver or receiver. Results of two kinds of tests are shown to verify the effectiveness of the control architecture. The first kind of tests demonstrates the reduction of the waiting time that can be achieved with the prediction of the virtual hand position. The second one permits to verify how the use of the pose of the hand instead of the only position improves the fluency of the hand-over tasks. An extended version of the algorithm that can predict also elbow and shoulder positions of the human arm is currently under development. In this way, not only the position of the hand but also the orientation of the forearm can be effectively anticipated by the robot.

Other important aspects related to hardware and software improvements will be addressed in future works. The implementation with a more efficient programming language will be extended to all the algorithms here presented, not only to point cloud fusion. In fact, MATLAB is not optimized for real-time application and would be convenient to write the algorithms using low-level programming languages, e.g., C++ or C#. This would also bring the possibility to test the proposed control scheme with a setup involving a reduced number of computers. Concerning the 3D vision system, since Kinect v2 is discontinued, the vision sensor will be updated to the last Azure Kinect DK. In this new scenario, it will be interesting to quantify the advantages of the new skeleton tracking software and to test an acquiring layout with more than two sensors.

References

[1] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Auton. Robots*, vol. 42, no. 5, pp. 957–975, 2018, doi: 10.1007/s10514-017-9677-2.

[2] Hospital Times, "NHS surgeons become pioneers in robotic surgery." https://www.hospitaltimes.co.uk/nhs-surgeons-become-pioneers-in-robotic-surgery (accessed Jun. 04, 2021).

[3] M. A. Peshkin, J. Edward Colgate, W. Wannasuphoprasit, C. A. Moore, R. Brent Gillespie, and P. Akella, "Cobot architecture," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 377–390, 2001, doi: 10.1109/70.954751.

[4] J. Krüger, T. K. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Ann. - Manuf. Technol.*, vol. 58, no. 2, pp. 628–646, 2009, doi: 10.1016/j.cirp.2009.09.009.

[5] F. Vicentini, "Collaborative Robotics: A Survey," *J. Mech. Des. Trans. ASME*, vol. 143, no. 4, pp. 1–20, 2021, doi: 10.1115/1.4046238.

[6] KUKA Roboter GmbH, "KUKA LBR iiwa product brochure-Sensitive Robotics_LBR iiwa," 2017. https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr iiwa (accessed May 19, 2021).

[7] ABB, "YuMi - IRB 14000 Collaborative Robot," 2015. https://new.abb.com/products/robotics/collaborative-robots/irb-14000-yumi (accessed Jun. 04, 2021).

[8] Universal Robots, "Robot UR5," 2021. https://www.universal-

robots.com/it/prodotti/robot-ur5/ (accessed Jun. 04, 2021).

[9] IFR, “Demystifying Collaborative Industrial Robots,” 2019.

[10] “ISO/TS 15066. Robots and Robotic Devices: Collaborative Robots,” *International Organization for Standardization*. Geneva, Switzerland, 2016.

[11] C. Chen, R. Jafari, and N. Kehtarnavaz, “A survey of depth and inertial sensor fusion for human action recognition,” *Multimed. Tools Appl.*, vol. 76, no. 3, pp. 4405–4425, 2017, doi: 10.1007/s11042-015-3177-1.

[12] L. S. Scimmi, “Development of a methodology for the human-robot interaction based on vision systems for collaborative robotics,” 2020.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics - Modelling, Planning and Control*. 2009.

[14] M. Ragaglia, A. M. Zanchettin, and P. Rocco, “Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements,” *Mechatronics*, vol. 55, no. December 2017, pp. 267–281, 2018, doi: 10.1016/j.mechatronics.2017.12.009.

[15] E. Rimon and D. E. Koditschek, “Exact Robot Navigation using Artificial Potential Functions,” *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, 1992, doi: 10.1109/70.163777.

[16] S. Giancola, M. Valenti, and R. Sala, *A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies*. 2018.

[17] structure-core, “structure-core.” https://structure.io/structure-core (accessed Sep. 07, 2021).

[18] imagingtechnology, “3D sensor technologies,” [Online]. Available: http://www.imagingtechnology.com/public_html/pdffiles/3DSensorTechnologies.pdf.

[19] D. Kim and S. Lee, “Advances in 3D camera: time-of-flight vs. active triangulation,” in *Intelligent Autonomous Systems 12*, 2012, vol. 1, pp. 301–310, doi: 10.1016/0921-8890(91)90033-H.

[20] P. Zanuttigh, C. D. Mutto, L. Minto, G. Marin, F. Dominio, and G. M. Cortelazzo, “Operating principles of structured light depth cameras,” in *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*, 2016, pp. 43–112.

[21] A. Grunnet-Jepsen and J. N. Sweetser, “Intel® RealSense™ Depth Cameras for Mobile Phones,” 2018.

[22] H. Sarbolandi, D. Lefloch, and A. Kolb, “Kinect range sensing: structured-light versus time-of-flight Kinect,” *Comput. Vis. Image Underst.*, vol. 139,

pp. 1–20, 2015, doi: 10.1016/j.cviu.2015.05.006.

[23] M. R. Andersen *et al.*, “Kinect depth sensor evaluation for computer vision applications,” 2012.

[24] P. Sturm, “Pinhole camera model,” in *Computer Vision*, Springer US, 2014, pp. 610–613.

[25] Microsoft, “Kinect Azure.” https://www.azure.com/kinect (accessed Sep. 07, 2021).

[26] E. Lachat, H. Macher, T. Landes, and P. Grussenmeyer, “Assessment and calibration of a RGB-D camera (Kinect v2 sensor) towards a potential use for close-range 3D modeling,” *Remote Sens.*, vol. 7, no. 10, pp. 13070–13097, 2015, doi: 10.3390/rs71013070.

[27] S. Hussmann, T. Ringbeck, and B. Hagebeuker, “A performance review of 3D TOF vision systems in comparison to stereo vision systems,” *Stereo Vis.*, no. November, pp. 103–120, 2008, doi: 10.5772/5898.

[28] B. Langmann, *Wide area 2D/3D imaging: development, analysis and applications*. Springer US, 2014.

[29] Nuitrack, “Skeletal Tracking Software.” https://nuitrack.com (accessed Sep. 07, 2021).

[30] J. Shotton *et al.*, “Real-Time human pose recognition in parts from single depth images,” *Commun. ACM*, vol. 56, no. 1, pp. 116–124, 2013, doi: 10.1145/2398356.2398381.

[31] Microsoft, “Kinect SDK 2.0.” https://developer.microsoft.com/it-it/windows/kinect (accessed Sep. 07, 2021).

[32] Occipital, “OpenNI SDK.” https://structure.io/openni (accessed Sep. 07, 2021).

[33] J. Rocha, “Skeltrack.” https://github.com/joaquimrocha/Skeltrack (accessed Sep. 07, 2021).

[34] G. Hidalgo *et al.*, “Openpose.” https://github.com/CMU-Perceptual-Computing-Lab/openpose (accessed Sep. 07, 2021).

[35] M. Munaro, A. Horn, R. Illum, J. Burke, and R. B. Rusu, “OpenPTrack.” https://github.com/OpenPTrack/open_ptrack_v2 (accessed Sep. 07, 2021).

[36] M. Melchiorre, L. S. Scimmi, S. P. Pastorelli, and S. Mauro, “Collison Avoidance using Point Cloud Data Fusion from Multiple Depth Sensors: A Practical Approach,” *2019 23rd Int. Conf. Mechatronics Technol. ICMT 2019*, 2019, doi: 10.1109/ICMECT.2019.8932143.

[37] J. M. D. Barros, F. Garcia, and D. Sidibé, “Real-time human pose estimation from body-scanned point clouds,” *VISAPP 2015 - 10th Int.*

Conf. Comput. Vis. Theory Appl. VISIGRAPP, Proc., vol. 1, pp. 553–560, 2015, doi: 10.5220/0005309005530560.

[38] K.-Y. Yeung, T.-H. Kwok, and C. C. L. Wang, "Improved skeleton tracking by duplex kinects : a practical approach for real-time applications," *J. Comput. Inf. Sci. Eng.*, pp. 1–10, 2013, doi: 10.1115/1.4025404.

[39] M. Melchiorre, L. S. Scimmi, S. Mauro, and S. Pastorelli, "Influence of human limb motion speed in a collaborative hand-over task," *ICINCO 2018 - Proc. 15th Int. Conf. Informatics Control. Autom. Robot.*, vol. 2, pp. 349–356, 2018, doi: 10.5220/0006864703490356.

[40] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik, "Evaluating and improving the depth accuracy of Kinect for Windows v2," *IEEE Sens. J.*, vol. 15, no. 8, pp. 4275–4285, 2015, doi: 10.1109/JSEN.2015.2416651.

[41] Microsoft, "Kinect SDK 2.0 - Body Tracking." https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799273(v=ieb.10) (accessed Jun. 23, 2021).

[42] L. Shuai, C. Li, X. Guo, B. Prabhakaran, and J. Chai, "Motion capture with ellipsoidal skeleton using multiple depth cameras," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 2, pp. 1085–1098, 2016, doi: 10.1109/TVCG.2016.2520926.

[43] J. R. Terven, "Kinect 2 Interface for Matlab," *GitHub*, 2021. .

[44] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa, "Effects of Robot Motion on Human-Robot Collaboration," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2015, pp. 51–58, doi: 10.1145/2696454.2696473.

[45] M. Koppenborg, P. Nickel, B. Naber, A. Lungfiel, and M. Huelke, "Effects of movement speed and predictability in human – robot collaboration," *Hum. Factors Ergon. Manuf. Serv. Ind.*, vol. 27, no. 4, pp. 197–209, 2017, doi: 10.1002/hfm.20703.

[46] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Mech. Mach. Sci.*, vol. 29, pp. 3–27, 2015, doi: 10.1007/978-3-319-14705-5_1.

[47] S. G. Tzafestas, "Mobile Robot Control and Navigation: A Global Overview," *J. Intell. Robot. Syst. Theory Appl.*, vol. 91, no. 1, pp. 35–58, 2018, doi: 10.1007/s10846-018-0805-9.

[48] T. S. Abhishek, D. Schilberg, and A. S. Arockia Doss, "Obstacle Avoidance Algorithms: A Review," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1012, p. 012052, 2021, doi: 10.1088/1757-899x/1012/1/012052.

[49] A. a Ata, "Optimal Trajectory Planning of Manipulators: A Review," *J.*

Eng. Sci. Technol., vol. 2, no. 1, pp. 32–54, 2007.

[50] E. K. Xidias, "Time-optimal trajectory planning for hyper-redundant manipulators in 3D workspaces," *Robot. Comput. Integr. Manuf.*, vol. 50, pp. 286–298, 2018, doi: 10.1016/j.rcim.2017.10.005.

[51] J. Mattmüller and D. Gisler, "Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path," *Int. J. Adv. Manuf. Technol.*, vol. 45, no. 9–10, pp. 1007–1016, 2009, doi: 10.1007/s00170-009-2032-9.

[52] R. R. Dos Santos, V. Steffen, and S. D. F. P. Saramago, "Robot path planning in a constrained workspace by using optimal control techniques," *Multibody Syst. Dyn.*, vol. 19, no. 1–2, pp. 159–177, 2008, doi: 10.1007/s11044-007-9059-1.

[53] F. Rubio, C. Llopis-Albert, F. Valero, and J. L. Suñer, "Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory," *Rob. Auton. Syst.*, vol. 86, pp. 106–112, 2016, doi: 10.1016/j.robot.2016.09.008.

[54] C. Llopis-Albert, F. Rubio, and F. Valero, "Optimization approaches for robot trajectory planning," *Multidiscip. J. Educ. Soc. Technol. Sci.*, vol. 5, no. 1, p. 1, 2018, doi: 10.4995/muse.2018.9867.

[55] P. Bosscher and D. Hedman, "Real-time collision avoidance algorithm for robotic manipulators," *Ind. Rob.*, vol. 38, no. 2, pp. 186–197, 2011, doi: 10.1108/01439911111106390.

[56] A. Kuntz, C. Bowen, and R. Alterovitz, "Interleaving Optimization with Sampling-Based Motion Planning (IOS-MP): Combining Local Optimization with Global Exploration," 2016, [Online]. Available: http://arxiv.org/abs/1607.06374.

[57] A. Sepehri and A. M. Moghaddam, "A Motion Planning Algorithm for Redundant Manipulators using Rapidly Exploring Randomized Trees and Artificial Potential Fields," *IEEE Access*, vol. 9, pp. 1–1, 2021, doi: 10.1109/access.2021.3056397.

[58] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[59] R. Volpe and P. Khosla, "Manipulator control with superquadratic artificial potential functions: theory and experiments," *IEEE Trans. Syst. Man Cybern.*, vol. 20, no. 6, pp. 1423–1436, 1990.

[60] I. Filippidis and K. J. Kyriakopoulos, "Adjustable navigation functions for unknown sphere worlds," in *Proceedings of the IEEE Conference on Decision and Control*, 2011, pp. 4276–4281, doi: 10.1109/CDC.2011.6161176.

[61] C. I. Connolly and R. A. Grupen, “The applications of harmonic functions to robotics,” *J. Robot. Syst.*, vol. 10, no. 7, pp. 931–946, 1993, doi: 10.1109/ISIC.1992.225141.

[62] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997, doi: 10.1109/100.580977.

[63] A. Chakravarthy and D. Ghose, “Obstacle avoidance in a dynamic environment: A collision cone approach,” *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans.*, vol. 28, no. 5, pp. 562–574, 1998, doi: 10.1109/3468.709600.

[64] S. Haddadin *et al.*, “Real-time reactive motion generation based on variable attractor dynamics and shaped velocities,” *IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc.*, pp. 3109–3116, 2010, doi: 10.1109/IROS.2010.5650246.

[65] J. Guldner and V. I. Utkin, “Sliding Mode Control for Gradient Tracking and Robot Navigation Using Artificial Potential Fields,” *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 247–254, 1995, doi: 10.1109/70.370505.

[66] C. De Medio and G. Oriolo, “Robot Obstacle Avoidance Using Vortex Fields,” *Adv. Robot Kinemat.*, pp. 227–235, 1991, doi: 10.1007/978-3-7091-4433-6_26.

[67] R. R. Murphy, *Introduction to AI robotics*. Cambridge, Massachusetts: The MIT Press, 2000.

[68] X. Zou and J. Zhu, “Virtual local target method for avoiding local minimum in potential field based robot navigation,” *J. Zhejiang Univ. Sci.*, vol. 4, no. 3, pp. 264–269, 2003.

[69] Z. Long, “Virtual target point-based obstacle-avoidance method for manipulator systems in a cluttered environment,” *Eng. Optim.*, vol. 52, no. 11, pp. 1957–1973, 2020, doi: 10.1080/0305215X.2019.1681986.

[70] M. Castelnovi, A. Sgorbissa, and R. Zaccaria, “Ghost-Goal algorithm for reactive safe navigation in outdoor environments,” in *Intelligent Autonomous Systems 9*, T. Arai et al. (Eds.), Ed. IOS Press, 2006, pp. 49–56.

[71] O. Arslan and D. E. Koditschek, “Sensor-based reactive navigation in unknown convex sphere worlds,” *Int. J. Rob. Res.*, vol. 38, no. 2–3, pp. 196–223, 2019, doi: 10.1177/0278364918796267.

[72] R. Beard and T. McClain, “Motion planning using potential fields,” *Brigham Young University, BYU ScholarsArchive, Faculty Publications 1313*. 2003, [Online]. Available: http://www.et.byu.edu/~beard/papers/preprints/BeardMcLain03-

potential.pdf.

[73] F. Flacco, T. Kroger, A. De Luca, and O. Khatib, “A Depth Space Approach to Human-Robot Collision Avoidance,” *IEEE Int. Conf. Robot. Autom.*, 2012.

[74] M. Parigi Polverini, A. M. Zanchettin, and P. Rocco, “A computationally efficient safety assessment for collaborative robotics applications,” *Robot. Comput. Integr. Manuf.*, vol. 46, no. November 2016, pp. 25–37, 2017, doi: 10.1016/j.rcim.2016.11.002.

[75] M. Safeea, P. Neto, and R. Bearee, “On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case,” *Rob. Auton. Syst.*, vol. 119, pp. 278–288, 2019, doi: 10.1016/j.robot.2019.07.013.

[76] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, “Optimal Collision-Free Robot Trajectory Generation Based on Time Series Prediction of Human Motion,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 226–233, 2018, doi: 10.1109/LRA.2017.2737486.

[77] K. Wei and B. Ren, “A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm,” *Sensors (Switzerland)*, vol. 18, no. 2, 2018, doi: 10.3390/s18020571.

[78] R.-J. Halme, M. Lanz, J. Kamarainen, R. Pieters, J. Latokartano, and A. Hietanen, “Review of vision-based safety systems for human-robot collaboration,” 2018, doi: 10.1016/j.procir.2018.03.043.

[79] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. Pastorelli, “Multiple collision avoidance between human limbs and robot links algorithm in collaborative tasks,” *ICINCO 2018 - Proc. 15th Int. Conf. Informatics Control. Autom. Robot.*, vol. 2, pp. 291–298, 2018, doi: 10.5220/0006852202910298.

[80] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. P. Pastorelli, “Implementing a Vision-Based Collision Avoidance Algorithm on a UR3 Robot,” *2019 23rd Int. Conf. Mechatronics Technol. ICMT 2019*, 2019, doi: 10.1109/ICMECT.2019.8932105.

[81] L. S. Scimmi, M. Melchiorre, M. Troise, S. Mauro, and S. Pastorelli, “A Practical and Effective Layout for a Safe Human-Robot Collaborative Assembly Task,” *Appl. Sci.*, vol. 11, no. 4, 2021.

[82] P. Corke, “Robotics Toolbox for MATLAB Realease 10,” *Robot. Toolbox*, p. 437, 2017, [Online]. Available: http://petercorke.com/Robotics_Toolbox.html.

[83] J. Ren, K. A. Mcisaac, R. V Patel, and T. M. Peters, “A potential field

model using generalized sigmoid functions," *Construction*, vol. 37, no. 2, pp. 477–484, 2007.

[84] M. Huber, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer, "Human-robot interaction in handing-over tasks," *Ro-Man*, pp. 107–112, 2008, doi: 10.1109/ROMAN.2008.4600651.

[85] S. Shibata, B. M. Sahbi, K. Tanaka, and A. Shimizu, "An analysis of the process of handing over an object and its application to robot motions," *1997 IEEE Int. Conf. Syst. Man, Cybern. Comput. Cybern. Simul.*, vol. 1, pp. 64–69, 1997, doi: 10.1109/ICSMC.1997.625724.

[86] K. W. Strabala *et al.*, "Towards Seamless Human-Robot Handovers," *J. Human-Robot Interact.*, vol. 2, no. 1, pp. 112–132, 2013, doi: 10.5898/jhri.2.1.strabala.

[87] F. Tamar and N. Hogans, "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.

[88] A. Agah and K. Tanie, "Human interaction with a service robot: mobile-manipulator handing over an object to a human," *Proc. Int. Conf. Robot. Autom.*, vol. 1, no. April, pp. 575–580, 1997, doi: 10.1109/ROBOT.1997.620098.

[89] M. Huber *et al.*, "Evaluation of a novel biologically inspired trajectory generator in human-robot interaction," *Proc. - IEEE Int. Work. Robot Hum. Interact. Commun.*, pp. 639–644, 2009, doi: 10.1109/ROMAN.2009.5326233.

[90] M. Cakmak, S. S. Srinivasa, M. Kyung Lee, S. Kiesler, and J. Forlizzi, "Using spatial and temporal contrast for fluent robot-human hand-overs," *HRI 2011 - Proc. 6th ACM/IEEE Int. Conf. Human-Robot Interact.*, no. June 2014, pp. 489–496, 2011, doi: 10.1145/1957656.1957823.

[91] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1986–1993, 2011, doi: 10.1109/IROS.2011.6048340.

[92] W. P. Chan, M. K. X. J. Pan, E. A. Croft, and M. Inaba, "Characterization of handover orientations used by humans for efficient robot to human handovers," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015-Decem, pp. 1–6, 2015, doi: 10.1109/IROS.2015.7353106.

[93] A. Bestick, R. Pandya, R. Bajcsy, and A. D. Dragan, "Learning human ergonomic preferences for handovers," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3257–3264, 2018, doi: 10.1109/ICRA.2018.8461216.

[94] A. G. Eguiluz, I. Rano, S. A. Coleman, and T. M. McGinnity, "Towards

robot-human reliable hand-over: Continuous detection of object perturbation force direction," *RO-MAN 2017 - 26th IEEE Int. Symp. Robot Hum. Interact. Commun.*, vol. 2017-Janua, pp. 510–515, 2017, doi: 10.1109/ROMAN.2017.8172350.

[95] M. K. X. J. Pan, E. A. Croft, and G. Niemeyer, "Exploration of geometry and forces occurring within human-to-robot handovers," *IEEE Haptics Symp. HAPTICS*, vol. 2018-March, pp. 327–333, 2018, doi: 10.1109/HAPTICS.2018.8357196.

[96] W. Wang, R. Li, Y. Diekel Z Max Chen, and Y. Jia, "Controlling Object Hand-Over in Human-Robot Collaboration via Natural Wearable Sensing," *IEEE Trans. Human-Machine Syst.*, vol. 49, no. 1, pp. 59–71, 2018, doi: 10.1109/THMS.2018.2883176.

[97] A. Sidiropoulos, E. Psomopoulou, and Z. Doulgeri, "A human inspired handover policy using Gaussian Mixture Models and haptic cues," *Auton. Robots*, vol. 43, no. 6, pp. 1327–1342, 2019, doi: 10.1007/s10514-018-9705-x.

[98] V. Micelli, K. Strabala, and S. S. Srinivasa, "Perception and Control Challenges for Effective Human-Robot Handoffs," *RSS 2011 RGB-D Work.*, 2011, [Online]. Available: http://www.cs.washington.edu/ai/Mobile_Robotics/rgbd-workshop-2011/camera_ready/micelli-rgbd11-hand-off.pdf.

[99] J. Aleotti, V. Micelli, and S. Caselli, "Comfortable robot to human object hand-over," *Proc. - IEEE Int. Work. Robot Hum. Interact. Commun.*, pp. 771–776, 2012, doi: 10.1109/ROMAN.2012.6343845.

[100] A. Koene, A. Remazeilles, M. Prada, A. Garzo, M. Puerto, and S. Endo, "Relative importance of spatial and temporal precision for user satisfaction in human-robot object handover interactions," *Artif. Intell. Simul. Behav.*, no. April, 2014.

[101] C.-M. Huang, M. Cakmak, and B. Mutlu, "Adaptive coordination strategies for human-robot handovers," *Robot. Sci. Syst. XI*, 2015, doi: 10.15607/RSS.2015.XI.031.

[102] D. Vogt, S. Stepputtis, B. Jung, and H. Ben Amor, "One-shot learning of human–robot handovers with triadic interaction meshes," *Auton. Robots*, vol. 42, no. 5, pp. 1053–1065, 2018, doi: 10.1007/s10514-018-9699-4.

[103] Y. Tamura, M. Sugi, J. Ota, and T. Arai, "Prediction of target object based on human hand movement for handing-over between human and self-moving trays," *Proc. - IEEE Int. Work. Robot Hum. Interact. Commun.*, pp. 189–194, 2006, doi: 10.1109/ROMAN.2006.314416.

[104] S. Parastegari, B. Abbasi, E. Noohi, and M. Zefran, "Modeling human

reaching phase in human-human object handover with application in robot-human handover," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017-Septe, pp. 3597–3602, 2017, doi: 10.1109/IROS.2017.8206205.

[105] H. Nemlekar, D. Dutia, and Z. Li, "Object transfer point estimation for fluent human-robot handovers," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 2627–2633, 2019, doi: 10.1109/ICRA.2019.8794008.

[106] W. Sakata, F. Kobayashi, and H. Nakamoto, "Robot-human handover based on motion prediction of human," *2017 6th Int. Conf. Informatics, Electron. Vis. 2017 7th Int. Symp. Comput. Med. Heal. Technol. ICIEV-ISCMHT 2017*, vol. 2018-Janua, pp. 1–4, 2018, doi: 10.1109/ICIEV.2017.8338592.

[107] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. Pastorelli, "Experimental Real-Time Setup for Vision Driven Hand-Over with a Collaborative Robot," *2019 Int. Conf. Control. Autom. Diagnosis, ICCAD 2019 - Proc.*, 2019, doi: 10.1109/ICCAD46983.2019.9037961.

[108] M. Melchiorre, L. S. Scimmi, S. Mauro, and S. P. Pastorelli, "Vision-based control architecture for human–robot hand-over applications," *Asian J. Control*, vol. 23, no. 1, pp. 105–117, 2021, doi: 10.1002/asjc.2480.

[109] M. Edwards and R. Green, "Low-Latency Filtering of Kinect Skeleton Data for Video Game Control," *Proc. 29th Int. Conf. Image Vis. Comput.*, pp. 190–195, 2014.

[110] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*, Wiley. 2001.

[111] K. Loumponias, N. Vretos, P. Daras, and G. Tsaklidis, "Using Kalman Filter and Tobit Kalman Filter in Order To Improve the Motion Recorded By Kinect Sensor II," *Proc. 29th Panhellenic Stat. Conf. 2016*, pp. 322–334, 2016, [Online]. Available: https://zenodo.org/record/1073287.

[112] S. Moon, Y. Park, D. W. Ko, and I. H. Suh, "Multiple Kinect Sensor Fusion for Human Skeleton Tracking Using Kalman Filtering," *Int. J. Adv. Robot. Syst.*, pp. 1–10, 2016, doi: 10.5772/62415.

[113] Universal Robots, "Robot UR3," 2021. https://www.universal-robots.com/it/prodotti/robot-ur3/ (accessed May 24, 2021).

[114] MathWorks, "knnsearch - Matlab." https://www.mathworks.com/help/stats/knnsearch.html (accessed Jun. 25, 2021).

[115] Robotiq, "2F-85 and 2F-140 Grippers," 2021. https://robotiq.com/products/2f85-140-adaptive-robot-gripper (accessed May 24, 2021).

[116] J. Berg, D. Gebauer, and G. Reinhart, "Method for the evaluation of layout

options for a human-robot collaboration," *Procedia CIRP*, vol. 83, pp. 139–145, 2019, doi: 10.1016/j.procir.2019.04.068.

[117] Universal Robots, "Remote control via TCP/IP," 2021. https://www.universal-robots.com/articles/ur/interface-communication/remote-control-via-tcpip/ (accessed May 24, 2021).

[118] Universal Robots, "The URScript Programming Language," 2021. https://www.universal-robots.com/download/manuals-e-series/script/script-manual-e-series-sw-510/ (accessed May 24, 2021).

[119] D. Frisch, "point2trimesh()-Distance between point and triangulated surface," 2021. https://it.mathworks.com/matlabcentral/fileexchange/52882-point2trimesh-distance-between-point-and-triangulated-surface (accessed May 26, 2021).

[120] E. Weisstein, "Cubic formula," *MathWorld-A Wolfram Web Resource*, 2021. https://mathworld.wolfram.com/CubicFormula.html (accessed Jun. 07, 2021).

MAP complements

Appendix A

From (3.12), the gradient of the repulsive potential field is:

$$\nabla U_o(\boldsymbol{p}, \beta_o, \gamma_o) = \frac{\partial}{\partial \boldsymbol{p}} U_o(\boldsymbol{p}, \beta_o, \gamma_o) = -\beta_o \gamma_o (\boldsymbol{p} - \boldsymbol{p}_o) e^{-\frac{\gamma_o}{2}\|\boldsymbol{p}-\boldsymbol{p}_o\|^2} \quad \text{(A.1)}$$

More precisely, it is written $\nabla U_o(\boldsymbol{p}, \beta_o, \gamma_o)$ to stress the dependency on β_o and γ_o, that are the design parameters of the potential field related to the obstacle. By considering $r_o = \|\boldsymbol{p} - \boldsymbol{p}_o\|$, the modulus of the gradient can be written as:

$$|\nabla U_o|(r_o, \beta_o, \gamma_o) = \beta_o \gamma_o r_o e^{-\frac{\gamma_o}{2} r_o^2} \quad \text{(A.2)}$$

The distance r_o where the magnitude of the gradient is maximum can be found by studying the derivative of (A.2):

$$\frac{\partial}{\partial r_o} |\nabla U_o|(r_o, \beta_o, \gamma_o) = \left(\frac{1}{\gamma_o} - r_o^2\right) \beta_o \gamma_o^2 e^{-\frac{\gamma_o}{2} r_o^2} = 0 \quad \text{(A.3)}$$

Equation (A.3) is verified if $r_o = 1/(\gamma_o)^{1/2}$. By substituting in (A.2):

$$|\nabla U_o|_{max}(\beta_o, \gamma_o) = \beta_o \gamma_o^{1/2} e^{-\frac{1}{2}} \quad \text{(A.4)}$$

If λ is the ratio between the modulus of the repulsive gradient at the generic distance r_o and its maximum value, it is:

$$\lambda(r_o, \gamma_o) = \frac{|\nabla U_o|(r_o, \beta_o, \gamma_o)}{|\nabla U_o|_{max}(\beta_o, \gamma_o)} = \gamma_o^{1/2} r_o e^{(\frac{1}{2} - \frac{\gamma_o}{2} r_o^2)} \tag{A.5}$$

Thus, λ does not depend on β_o. This suggests that (A.5) can be used to choose γ_o. At the contour of the obstacle, i.e. in $r_o = R_o$, the ratio is:

$$\lambda_o(R_o, \gamma_o) = \frac{|\nabla U_o|_{R_o}}{|\nabla U_o|_{max}} = \gamma_o^{1/2} R_o e^{(\frac{1}{2} - \frac{\gamma_o}{2} R_o^2)} \tag{A.6}$$

which is the explicit form of (3.16).

Appendix B

By taking the square, (A.6) becomes:

$$-\gamma_o R_o^2 e^{-\gamma_o R_o^2} = -\frac{\lambda_o^2}{e} \tag{B.1}$$

which can be written as:

$$\begin{gathered} we^w = u \\ w = -\gamma_o R_o^2 \ , \qquad u = -\frac{\lambda_o^2}{e} \end{gathered} \tag{B.2}$$

Equation (B.2) can be solved with the Lambert *W* function as long as $\lambda_o \leq 1$, which is always verified by definition (A.5). Since u is real and $-1/e \leq u < 0$, (B.2) has exactly two real solutions, corresponding to the two branches W_0 and W_{-1}. The meaningful solution, in this case, is:

$$w = W_{-1}(u) \tag{B.3}$$

that leads to (3.17). In fact, it can be verified that the γ_o obtained with the case W_{-1} allows to concentrate U_o around the obstacle (Figure B.1).

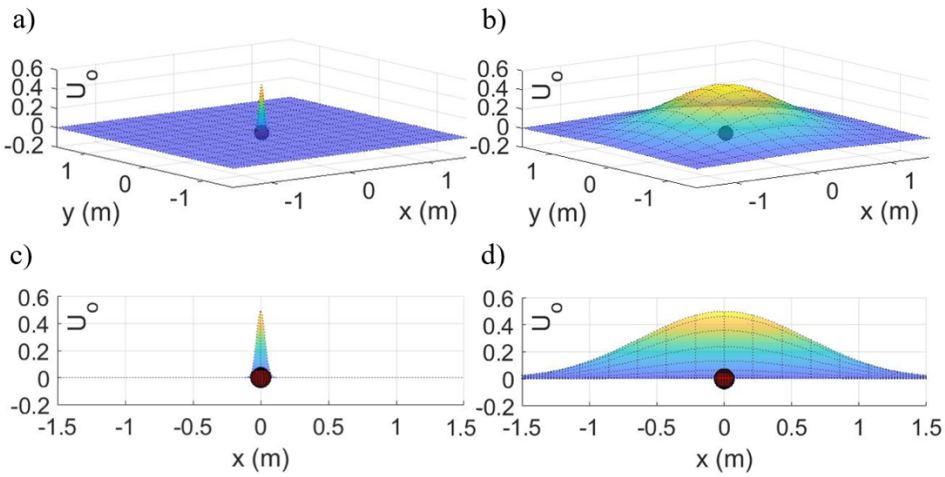


Figure B.1 Repulsive potential in a radial plane of a spherical obstacle with $R_o = 0.075\ m$. U_o is obtained by choosing $\lambda_o = 0.2$, $\beta_o = 0.5$ and by varying γ_o: in a) and c) is depicted the same function from different views, with $\gamma_o = 1069$, which corresponds to the solution W_{-1}; in b) and d) the case with $\gamma_o = 2.6554$, which relates to W_0, is shown

Appendix C

Outside of the active region the gradient goes to zero. To quantify this condition, for U_o it is assumed that the gradient magnitude is less than or equal to a small value s_ϵ. Thus, by placing $|\nabla U_o|(r_o, \beta_o, \gamma_o) = s_\epsilon$ in (A.2) and by taking the square, the following relation holds:

$$-\gamma_o r_o^2 e^{-\gamma_o r_o^2} = -\frac{s_\epsilon^2}{\beta_o^2 \gamma_o} \tag{C.1}$$

With the same fashion of Appendix B, (C.1) can be written as:

$$\begin{gathered} we^w = u_\epsilon \\ w = -\gamma_o r_o^2\ , \qquad u_\epsilon = -\frac{s_\epsilon^2}{\beta_o^2 \gamma_o} \end{gathered} \tag{C.2}$$

which can be solved with the Lambert W function as long as $s_\epsilon^2 \leq \beta_o^2 \gamma_o / e$. Since the latter condition is easily verified in practice as proved in the results section, (C.2) has two real solutions.

Without loss of generality, the meaningful solution corresponds to the lower branch W_{-1}:

$$w = W_{-1}(u_\epsilon) \tag{C.3}$$

Equation (C.3) solved for r_o gives (3.18).

Appendix D

From (3.14), the gradient of the attractive potential field is:

$$\nabla U_a(\boldsymbol{p}, \alpha_a, \gamma_a) = \frac{\partial}{\partial \boldsymbol{p}} U_a(\boldsymbol{p}, \alpha_a, \gamma_a) = \alpha_a \gamma_a (\boldsymbol{p} - \boldsymbol{p}_a) e^{-\frac{\gamma_a}{2}\|\boldsymbol{p}-\boldsymbol{p}_a\|^2} \tag{D.1}$$

By considering $r_a = \|\boldsymbol{p} - \boldsymbol{p}_a\|$, the modulus of the gradient can be written as:

$$|\nabla U_a|(r_a, \alpha_a, \gamma_a) = \alpha_a \gamma_a r_a e^{-\frac{\gamma_a}{2} r_a^2} \tag{D.2}$$

which is similar to (A.2). Thus, with the same procedure of Appendix A, by introducing μ as the ratio between the modulus of the attractive gradient at the generic distance r_a and its maximum value, it is:

$$\mu(r_a, \gamma_a) = \frac{|\nabla U_a|(r_a, \alpha_a, \gamma_a)}{|\nabla U_a|_{max}(\alpha_a, \gamma_a)} = \gamma_a^{1/2} r_a e^{(\frac{1}{2} - \frac{\gamma_a}{2} r_a^2)} \tag{D.3}$$

At the distance $r_a = R_a$, the ratio becomes:

$$\mu_a(R_a, \gamma_a) = \frac{|\nabla U_a|_{R_a}}{|\nabla U_a|_{max}} = \gamma_a^{1/2} R_a e^{(\frac{1}{2} - \frac{\gamma_a}{2} R_a^2)} \tag{D.4}$$

which is the explicit form of (3.21). Equation (3.22) is obtained as described in Appendix B, by replacing γ_o, R_o and λ_o with γ_a, R_a and μ_a respectively.

Appendix E

To quantify the active region of U_a, the condition on the gradient is replaced with a condition on μ. In particular, beyond the distance R_a^* from $\boldsymbol{p}_a$, must be $\mu < \mu_\epsilon$ where μ_ϵ is a small number. Thus, by placing $\mu = \mu_\epsilon$ in (D.3) and by taking the square:

$$-\gamma_a r_a^2 e^{-\gamma_a r_a^2} = -\frac{\mu_\epsilon^2}{e} \tag{E.1}$$

Equation (E.1) is similar to (B.1) and can be solved for $r_a = R_a^*$ with the Lambert W function as long as $\mu_\epsilon \leq 1$. Without loss of generality, the meaningful solution is given in the form (B.3), which leads to (3.23).

Appendix F

From (3.28):

$$\alpha_a \gamma_a e^{-\frac{\gamma_a}{2}(x'-x'_a)^2} = \frac{-\sigma x'}{(x'-x'_a)} \tag{F.1}$$

By recognizing this term in (3.30) and substituting, it results the cubic (3.31), which can be written as:

$$x'^3 - (2x'_a)x'^2 + (x'^2_a)x' - \frac{x'_a}{\gamma_a} = 0 \tag{F.2}$$

Equation (F.2) can be solved using the cubic formula ([120]). By considering $b_2 = 2x'_a$, $b_1 = x'^2_a$ and $b_0 = -x'_a/\gamma_a$, the cubic has three real solutions if the polynomial discriminant D is negative:

$$\begin{gathered} D = Q^3 + P^2 = x'^2_a \left(\frac{1}{4\gamma_a} - \frac{x'^2_a}{27}\right) < 0 \\ Q = \frac{3b_1 - b_2^2}{9} = -\frac{x'^2_a}{9} \quad , \quad P = \frac{9b_2 b_1 - 27b_0 - 2b_2^3}{54} = -\frac{x'^3_a}{27} + \frac{x'_a}{2\gamma_a} \end{gathered} \tag{F.3}$$

Since x'_a is positive, condition (F.3) translates into (3.32). In this case, the three solutions are:

$$\begin{gathered} x'_I = 2\sqrt{-Q}\cos\left(\frac{\theta}{3}\right) - \frac{b_2}{3} \\ x'_{II} = 2\sqrt{-Q}\cos\left(\frac{\theta + 2\pi}{3}\right) - \frac{b_2}{3} \\ x'_{III} = \tilde{x}' = 2\sqrt{-Q}\cos\left(\frac{\theta + 4\pi}{3}\right) - \frac{b_2}{3} \\ \theta = \cos^{-1}\left(\frac{P}{\sqrt{-Q^3}}\right) \end{gathered} \tag{F.4}$$

By substituting the three roots in (3.28), as many values for α_a can be found. Figure F.1 shows the meaning of the three solutions of a generic case, given σ, x'_a

and γ_a. The first root x'_I, represented by the dashed curve, gives a negative α_a. The dotted curve refers to the root x'_{II} and to a high positive value of α_a, which gives the tangency nearby the global minimum still producing a local minimum around x'_a. The only meaningful solution for the case study of this paper is then x'_{III}, which can be written in the form (3.33).

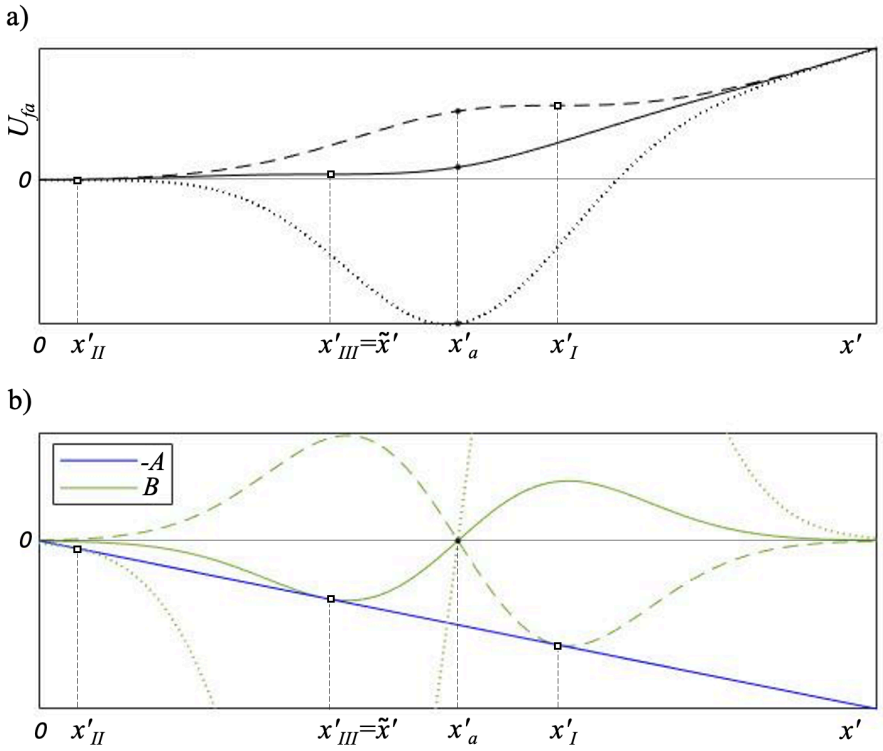


Figure F.1. Analysis of the parametric solution of equation (3.28). The different style of the lines refers to the 3 values of α_a α_a obtained with x'_I, x'_{II} and x'_{III}. a) Potential function U_{fa} along the x' axis. b) Graphical solution by means of the intermediate variables (3.29); the squares identify the points of tangency.

Appendix G

To satisfy condition (3.20), the limit distance between the local attractor and the desired position is:

$$\|\boldsymbol{p}_a - \boldsymbol{p}_f\| = \|\boldsymbol{p}'_a - \boldsymbol{p}'_f\| = x'_a = R^*_a \tag{G.1}$$

in which the distance between the local attractor and the saddle point is maximum. In this case:

$$\varepsilon = (x'_a - \tilde{x}')_{max} = R^*_a - \tilde{x}' \tag{G.2}$$

By considering (3.33), for $x'_a = R^*_a$ the saddle point coordinate $\tilde{x}'$ can be written as:

$$\begin{aligned} \tilde{x}'(R^*_a, \gamma_a) &= \frac{2}{3} R^*_a \left[\cos\left(\frac{\vartheta_\varepsilon + 4\pi}{3} \right) + 1 \right] \\ \vartheta(R^*_a, \gamma_a) &= \vartheta_\varepsilon = \cos^{-1}\left(\frac{27}{2\gamma_a R^{*2}_a} - 1 \right) \end{aligned} \tag{G.3}$$

Then, by substituting (G.3) in (G.2):

$$\varepsilon = \frac{1}{3} R^*_a \left[1 - 2\cos\left(\frac{\vartheta_\varepsilon + 4\pi}{3} \right) \right] \tag{G.4}$$

Since R^*_a is given by (3.23), equation (G.4) can be written in the form (3.36).

List of symbols

A	x'-axis component of the gradient of the potential field related to the desired position, evaluated in $y' = 0$
$\boldsymbol{\mathcal{A}}_K$	Transformation matrix from Kinect frame to world frame
$\boldsymbol{\mathcal{A}}_r$	Transformation matrix from robot frame to world frame
$\boldsymbol{\mathcal{A}}''$	Transformation matrix from world frame to obstacle frame
$\boldsymbol{\mathcal{A}}''_t$	Transformation matrix $\boldsymbol{\mathcal{A}}''$ at the generic time t
a	physical marker for spatial matching
a_{DH}	Denavit-Hartenberg parameter: length of the common normal
$\boldsymbol{a}_{FA}$	Forearm axis
a_{max}	Maximum magnitude of the end-effector linear acceleration
$\boldsymbol{a}^K$	marker a observed in the Kinect depth frame
$\boldsymbol{a}^r$	marker a observed in the robot frame
B	x'-axis component of the gradient of the potential field related to the local attractor, evaluated in $y' = 0$

b	physical marker for spatial matching
b_0, b_1, b_2	Coefficients of the cubic formula
$\boldsymbol{b}^K$	marker b observed in the Kinect depth frame
$\boldsymbol{b}^r$	marker b observed in the robot frame
c_x, c_y	Coordinates of the principal point in the image plane of Kinect depth sensor
D	Discriminant of the cubic formula
d_{DH}	Denavit Hartenberg parameter: offset along previous joint axis to the common normal
$\boldsymbol{d}_{e-o}$	Distance vector between the end-effector and the obstacle
d_{e-o}	Distance between the end-effector and the obstacle
$\boldsymbol{d}_{Li-human}$	Minimum distance vector between the set of points L_i and the obstacle, identified as the human
$\boldsymbol{d}_{Li-o}$	Minimum distance vector between the set of points L_i and the obstacle
$\boldsymbol{e}$	Error between the desired and the actual end-effector pose
$\boldsymbol{e}_o$	Error between the desired and the actual end-effector orientation
$\boldsymbol{e}_p$	Error between the desired and the actual end-effector position
$\boldsymbol{e}_{p,TCP}$	Error between the desired and the actual TCP position
$\boldsymbol{F}$	State transition matrix
$\boldsymbol{F}_k$	State transition matrix at time index k
$\mathcal{F}_W$	Contracted notation of the world reference frame
f_c	Robot control frequency
f_x, f_y	Focal lengths of Kinect depth sensor
$\boldsymbol{H}$	Measurement matrix

$\boldsymbol{H}_k$	Measurement matrix at time index k
h	$= 0, 1, ... N$. Continuous time index
$\boldsymbol{I}$	Identity matrix
i	$= 0, 1, ... n$. Index that distinguishes different n points or sets of points
J	Jacobian
J_{L_i}	Partial Jacobian related to the set of control points L_i
j	$= 0, 1, ... n$. Alternative notation to index i, used to refer to different generic points within the same formula
$\boldsymbol{\mathcal{K}}$	Matrix of intrinsic parameters of Kinect camera
$\boldsymbol{K}_J$	Gain matrix of the control law with Jacobian inverse
$\boldsymbol{K}_k$	Kalman gain at time step k
$\boldsymbol{K}_o$	Gain matrix related to the orientation error of the control law with Jacobian inverse
k	Discrete-time step index
k_1, k_2, k_3	Coefficients of radial distortion of Kinect depth sensor
L_i	Generic set of robot control points
$l_{i,j}$	Distance between two consecutive joints of the mean skeleton. Also identified as bone length
n_k	Prediction index
O	Origin of the world frame. In spatial matching, it is identified by a physical marker
$O - xyz$	World reference frame
$O' - x'y'z'$	Auxiliary reference frame
$O'' - x''y''z''$	Obstacle reference frame

$O_d - x_d y_d$	Image plane reference frame, considering distortion
$O_e - x_e y_e z_e$	End-effector reference frame
$O_K - x_K y_K z_K$	Kinect depth reference frame
$O_p - x_p y_p$	Image plane reference frame
$O_r - x_r y_r z_r$	Robot reference frame, placed at the robot base
$\boldsymbol{o}^K$	Origin of the world reference frame observed in the Kinect frame
$\boldsymbol{o}^r$	Origin of the world reference frame observed in the robot frame
P	Intermediate variable of the cubic formula
$\boldsymbol{P}_k^+$	Updated state covariance matrix at time index k
$\boldsymbol{P}_k^-$	A priori estimate of the state covariance matrix at time index k
$\boldsymbol{P}_{k-1}^+$	Updated state covariance matrix at time index $k-1$
p	$= 1,\ 2,\ \ldots m$. Index to distinguish different obstacles
$\boldsymbol{p}$	Generic point in the world frame
$\boldsymbol{p}'$	Generic point in the auxiliary frame
$\tilde{\boldsymbol{p}}'$	$= [\tilde{x}'\ 0\ 0]^T$. Saddle point related to the local attractor, observed in the auxiliary frame
$\boldsymbol{p}^K$	$= [p_x^K\ p_y^K\ p_z^K]^{\mathrm{T}}$ Generic point in the Kinect frame
$\hat{\boldsymbol{p}}^K$	Generic point in the Kinect frame, expressed with homogeneous coordinates
$\boldsymbol{p}_A, \boldsymbol{p}_B, \boldsymbol{p}_C$	Strategical nodes of the robot trajectory for the assembly task
$\boldsymbol{p}_a$	Position of the local attractor in the world frame
$\boldsymbol{p}'_a$	$= [x'_a\ 0\ 0]^T$. Local attractor observed in the auxiliary reference frame
$\hat{\boldsymbol{p}}_a$	Position vector of the local attractor in the world frame, expressed with homogeneous coordinates

$\hat{\boldsymbol{p}}_a''$	Position of the local attractor in the obstacle frame, expressed with homogeneous coordinates
$\boldsymbol{p}_{ai}$	Position i of the local attractor with fixed radius R_a in the world frame
$\boldsymbol{p}'_{ai}$	$= [x'_{ai}\ 0\ 0]^T$. Position i of the local attractor with fixed radius R_a, observed in the auxiliary frame
$\boldsymbol{p}_{a,i}$	Position of the local attractor i in the world frame
$\boldsymbol{p}_{a,j}$	Position of the local attractor j in the world frame
$\boldsymbol{p}_{a,t}$	Position of the local attractor at the generic time t
$\boldsymbol{p}_{a,t_h}$	Position of the local attractor at time t_h
$\dot{\boldsymbol{p}}_d$	End-effector linear velocity in the world frame
$\boldsymbol{p}_E^*$	Human elbow joint optimized by the skeleton fusion algorithm, observed in the world frame
$\boldsymbol{p}_e$	Position of the end-effector in the world frame
$\dot{\boldsymbol{p}}_e$	End-effector linear velocity
$\boldsymbol{p}_f$	Final position of the end-effector in the world frame, alternatively indicated as global attractor in the MAP theory
$\boldsymbol{p}'_f$	Final position observed in the auxiliary reference frame
$\boldsymbol{p}_H^*$	Human hand joint optimized by the skeleton fusion algorithm, observed in the world frame
$\boldsymbol{p}^e_{H,n_k}$	Predicted human hand position at time horizon n_kT, observed in the world frame
$\boldsymbol{p}^e_{H,2}$	Predicted human hand position at time horizon $2T$, observed in the world frame
$\boldsymbol{p}_i^A$	Generic joint i of the skeleton acquired by Kinect A, observed in the world frame
$\overline{\boldsymbol{p}}_i^A$	Mean position of the joint $\boldsymbol{p}_i^A$ over 30 frames

$\boldsymbol{p}_i^B$	Generic joint i of the skeleton acquired by Kinect B, observed in the world frame.
$\overline{\boldsymbol{p}}_i^B$	Mean position of the joint $\boldsymbol{p}_i^B$ over 30 frames
$\tilde{\boldsymbol{p}}_i'$	$= [\tilde{x}_i' \ 0 \ 0]^T$. Saddle point related to the local attractor at position *i*, observed in the auxiliary frame
$\boldsymbol{p}_i^*$	Joint i optimized by the skeleton fusion algorithm, observed in the world frame
$\boldsymbol{p}_o$	Position of the obstacle in the world frame.
$\boldsymbol{p}_{o,p}$	Position of the obstacle p in the world frame
$\boldsymbol{p}_{o,t}$	Position of the obstacle at the generic time t
$\boldsymbol{p}_r$	Position of the robot considered as a point in the world frame
$\boldsymbol{p}_s$	Initial position of the end-effector in the world frame
$\boldsymbol{p}_{TCP}$	Position of the TCP in the world frame
$\boldsymbol{p}_{VH}$	Virtual hand in the world frame
$\boldsymbol{p}_W^*$	Human wrist joint optimized by the skeleton fusion algorithm, observed in the world frame
Q	Intermediate variable of the cubic formula
$\boldsymbol{Q}$	$= E\{\boldsymbol{w}\boldsymbol{w}^T\}$ Covariance matrix of the process noise
$\boldsymbol{Q}_e$	Unit quaternion identifying the actual end-effector orientation with respect to the world frame
$\boldsymbol{Q}_f$	Unit quaternion identifying the final end-effector orientation with respect to the world frame
$\boldsymbol{Q}_k$	Covariance matrix of the process noise at time step k
$\boldsymbol{q}$	Manipulator joint positions
$\tilde{q}$	Process noise intensity

$\dot{\boldsymbol{q}}$	Manipulator joint velocities
$\dot{\boldsymbol{q}}_{fb}$	Manipulator joint velocity feedback
$\boldsymbol{q}_k$	Manipulator joint positions at discrete time step k
$\boldsymbol{q}_{k-1}$	Manipulator joint positions at discrete time step $k-1$
$\boldsymbol{q}_{Li}$	Manipulator joint positions related to the set of control points L_i
$\dot{\boldsymbol{q}}_{Li}$	Manipulator joint velocities related to the set of control points L_i
$\dot{\boldsymbol{q}}_{set}$	Manipulator joint velocity set
$\dot{\boldsymbol{q}}_{set,k}$	Manipulator joint velocity set at discrete time step k
$\boldsymbol{R}$	$= E\{\boldsymbol{v}\boldsymbol{v}^T\}$ Covariance matrix of the measurement noise
R_a	Radius of the local attractor
R_a^*	Radius of the active region of the potential field related to the local attractor
$R_{a,i}$	Radius of the local attractor i
$R_{a,i}^*$	Radius of the active region of the potential field related to the local attractor i
$R_{a,lim}$	Maximum admissible radius of the local attractor
R_{a,t_h}	Radius of the local attractor at time t_h
R_{a,t_h}^*	Radius of the active region of the local attractor at time t_h
$\boldsymbol{\mathcal{R}}_e$	Rotation matrix identifying the actual end-effector orientation with respect to the world frame
$\boldsymbol{\mathcal{R}}_f$	Rotation matrix identifying the final end-effector orientation with respect to the world frame
$\boldsymbol{\mathcal{R}}_K$	Rotation matrix identifying the orientation of Kinect frame with respect to the world frame.
$\boldsymbol{R}_k$	Covariance matrix of the measurement noise at time index k

$\boldsymbol{\mathcal{R}}_K$	Rotation matrix identifying the orientation of Kinect frame with respect to the world frame
R_o	Radius of the spherical obstacle
R_o^*	Radius of the active region of the potential field related to the obstacle
$R_{o,p}$	Radius of the spherical obstacle p
$R_{o,p}^*$	Radius of the active region of the potential field related to the obstacle p
$\tilde{r}$	Measurement noise intensity
r_a	Generic distance from the local attractor center
r_o	Generic distance from the obstacle center
$\boldsymbol{\mathcal{S}}(\circ)$	Skew-symmetric operator
S_A	Skeleton of Kinect A
S_B	Skeleton of Kinect B
$\boldsymbol{s}$	Human hand position in the world frame
$\dot{\boldsymbol{s}}$	Human hand velocity in the world frame
$\ddot{\boldsymbol{s}}$	Human hand acceleration in the world frame
s_ϵ	Zero threshold
T	Sampling period of the discrete-time state equation
t	Time
t_h	Time instant h
U_a	Potential field related to the local attractor
U_a^*	Active region of the potential field related to the local attractor
$U_{a,i}^*$	Active region of the potential field related to the local attractor i
$U_{a,t}$	Potential field of the local attractor at the generic time t

$U^*_{a,t}$	Active region of the potential field related to the local attractor at the generic time t
U_f	Potential field related to the final end-effector position
U_{fo}	Sum of the potential fields related to the final position and to the obstacle
U_o	Potential field related to the obstacle
U^*_o	Active region of the potential field related to the obstacle
$U^*_{o,p}$	Active region of the potential field related to the obstacle p
U_t	Sum of the potential fields related to the desired position, the obstacle and the local attractor
$U_{t,MAP}$	Potential field U_t whose parameters have been selected according to the MAP (Multiple Attractors Potential)
u	Argument of the Lambert *W* function for γ_o calculation
u_x	Coordinate x in the pixel frame
u_y	Coordinate y in the pixel frame
u_ϵ	Argument of the Lambert *W* function for R^*_o calculation
V_m	Meeting volume
$\boldsymbol{v}$	Measurement noise
$\boldsymbol{v}_{att}$	Attractive linear velocity
v_{att}	Magnitude of the attractive linear velocity
$v_{att,max}$	Maximum magnitude of the attractive linear velocity
v_e	Magnitude of the end-effector linear velocity
$\boldsymbol{v}_k$	Measurement noise at time index k
$\boldsymbol{v}_{Li}$	Repulsive linear velocity related to the set of control points L_i
v_{Li}	Magnitude of the repulsive linear velocity related to the set of control points

	L_i
v_{max}	Maximum magnitude of the end-effector linear velocity
$\boldsymbol{v}_{rep}$	Repulsive linear velocity
v_{rep}	Magnitude of the repulsive linear velocity
$v_{rep,max}$	Maximum magnitude of the repulsive linear velocity
$\boldsymbol{w}$	Process noise
w	Intermediate variable that represents the solution of the Lambert *W* function.
w_i^A	Optimization weighting coefficient related to the generic joint i of the skeleton acquired by Kinect A.
w_i^B	Optimization weighting coefficient related to the generic joint i of the skeleton acquired by Kinect B.
$\boldsymbol{w}_k$	Process noise at time index k
x'_I, x'_{II}, x'_{III}	Roots of the cubic formula
$\boldsymbol{x}$	$= [\boldsymbol{s}\ \dot{\boldsymbol{s}}\ \ddot{\boldsymbol{s}}]^T$ State vector
$\tilde{x}'$	x'-axis coordinate of the saddle point related to the local attractor
x'_a	$= \|\boldsymbol{p}'_a - \boldsymbol{p}'_f\| = \|\boldsymbol{p}_a - \boldsymbol{p}_f\|$. Distance between the local attractor and the final position
x'_{ai}	$= \|\boldsymbol{p}'_{ai} - \boldsymbol{p}'_f\| = \|\boldsymbol{p}_{ai} - \boldsymbol{p}_f\|$. Distance between the local attractor with fixed radius R_a at position i and the final position
$x'_{a,i}$	Distance between the local attractor i and the final position
$x'_{a,t}$	Distance x'_a at the generic time t
x'_{a,t_h}	Distance x'_a at time t_h
$\tilde{}$	axis coordinate of the saddle point related to the local attractor at position i
$\boldsymbol{x}_k$	State vector at time index k

$\overline{\boldsymbol{x}}_k^+$	Updated mean state vector at time index k
$\overline{\boldsymbol{x}}_k^-$	A priori estimate of the mean state vector at time index k
$\boldsymbol{x}_{k-1}$	State vector at time index $k-1$
$\overline{\boldsymbol{x}}_{k-1}^+$	Updated mean state vector at time index $k-1$
$\overline{\boldsymbol{x}}_{k+n_k}^-$	Predicted state at time index $k+n_k$
$\boldsymbol{z}_k$	Measurement vector at time index k
α	Shape factor of the magnitude of repulsive velocity
α_a	Intensity of the potential function related to the local attractor
$\tilde{\alpha}_a$	Value of α_a that generates the saddle point related to the local attractor
$\tilde{\alpha}_{ai}$	Value of α_a that generates the saddle point related to the local attractor with fixed radius R_a at position i
$\tilde{\alpha}_{a,i}$	Value of α_a which generate the saddle point related to the local attractor i
$\alpha_{a,t}$	Parameter α_a at the generic time t
$\tilde{\alpha}_{a,t}$	Parameter $\tilde{\alpha}_a$ at the generic time t
α_{a,t_h}	Parameter α_a at time t_h
$\tilde{\alpha}_{a,t_h}$	Parameter $\tilde{\alpha}_a$ at time t_h
$\alpha_{att,1}, \alpha_{att,2}$	Shape factors of the magnitude of attractive velocity
α_{DH}	Denavit Hartenberg parameter: angle about common normal
β_o	Intensity of the potential function related to the obstacle
$\beta_{o,p}$	Intensity of the potential function related to the obstacle p
δ	Rotation angle about the negative direction of the z-axis
γ_a	Exponential decay of the potential function related the local attractor

$\gamma_{a,i}$	Exponential decay of the potential function related the local attractor i
$\gamma_{a,lim}$	Maximum admissible value of γ_a
$\gamma_{a,t}$	Parameter γ_a at the generic time t
γ_{a,t_h}	Parameter γ_a at time t_h
γ_o	Exponential decay of the potential function related the obstacle
$\gamma_{o,p}$	Exponential decay of the potential function related the obstacle p
ε	Maximum distance between the saddle point and the local attractor
$\tilde{\varepsilon}$	Minimum admissible distance between the local attractor and the active region of the obstacle
$\tilde{\varepsilon}_i$	Minimum admissible distance between the local attractor i and the active region of the obstacle
ε_{lim}	Maximum admissible value of ε
η_e	Scalar part of the quaternion $\boldsymbol{\mathcal{Q}}_e$
η_f	Scalar part of the quaternion $\boldsymbol{\mathcal{Q}}_f$
ϑ	Angle of the cubic formula
ϑ_{DH}	Denavit Hartenberg parameter: angle about the previous joint axis
ϑ_ε	Value of the angle of the cubic formula evaluated at R_a^*
λ	$= \lvert\nabla U_o\rvert / \lvert\nabla U_o\rvert_{max}$. Gradient ratio of the obstacle
λ_o	$= \lvert\nabla U_o\rvert_{R_o} / \lvert\nabla U_o\rvert_{max}$. Design ratio of the obstacle
$\lambda_{o,p}$	Design ratio of the obstacle p
$''$	$- \lvert\nabla U_a\rvert / \lvert\nabla U_a\rvert_{max}$. Gradient ratio of the local attractor
μ_a	$= \lvert\nabla U_a\rvert_{R_a} / \lvert\nabla U_a\rvert_{max}$. Design ratio of the local attractor
$\mu_{a,i}$	Design ratio of the local attractor i

μ_ϵ	Zero-ratio threshold
ρ	Reference distance of the magnitude of repulsive velocity
$\rho_{att,1}$, $\rho_{att,2}$	Reference distances of the magnitude of attractive velocity
σ	Intensity of the quadratic potential function related to the final position
$\boldsymbol{\sigma}_e$	Vector part of the quaternion $\boldsymbol{\mathcal{Q}}_e$
$\boldsymbol{\sigma}_f$	Vector part of the quaternion $\boldsymbol{\mathcal{Q}}_f$
$\dot{\boldsymbol{\phi}}_d$	Rotational velocity of the end-effector frame with respect to the world frame
$\boldsymbol{\phi}_e$	Orientation of the end-effector frame with respect to the world frame
$\dot{\boldsymbol{\phi}}_e$	Rotational velocity of the end-effector frame with respect to the world frame
$\boldsymbol{\phi}_f$	Final orientation of the end-effector frame with respect to the world frame
$\boldsymbol{\phi}_s$	Initial orientation of the end-effector frame with respect to the world frame
$\boldsymbol{\chi}_d$	End-effector desired pose
$\dot{\boldsymbol{\chi}}_d$	End-effector desired velocity
$\boldsymbol{\chi}_e$	End-effector pose
$\dot{\boldsymbol{\chi}}_e$	End-effector velocity
$\boldsymbol{\chi}_f$	End-effector final pose
$\dot{\boldsymbol{\chi}}_{Li}$	Repulsive velocity related to the set of control points L_i
$\dot{\boldsymbol{\chi}}_{rep}$	End-effector repulsive velocity
$\boldsymbol{\chi}_s$	End-effector initial pose
$\dot{\boldsymbol{\chi}}_{TCP}$	TCP velocity
ψ_{max}	Maximum magnitude of the end-effector angular acceleration
$\boldsymbol{\omega}_e$	End-effector angular velocity

ω_e	Magnitude of the robot angular velocity
ω_{max}	Maximum magnitude of the end-effector angular velocity
$\boldsymbol{\omega}_{TCP}$	TCP angular velocity
$\nabla(\circ)$	Gradient operator
$\lvert\nabla U_a\rvert_{max}$	Maximum magnitude of the gradient of the potential field related to the local attractor
$\lvert\nabla U_a\rvert_{R_a}$	Modulus of the gradient of the potential field related to the local attractor, evaluated at the radius of the local attractor
$\lvert\nabla U_o\rvert_{max}$	Maximum magnitude of the gradient of the potential field related to the obstacle