Universidad
Carlos III de Madrid

Planning and Estimation Algorithms for Human-like Grasping

David Álvarez Sánchez

Ph.D. Thesis

Department of Systems Engineering and Automation
Leganés, Madrid, Spain, September 2016

Planning and Estimation Algorithms for Human-like Grasping

Candidate
David Álvarez Sánchez

Advisers
Luis Enrique Moreno Lorente
Santiago Garrido Bullón

Review Committee

President: _____

Secretary: _____

Vocal: _____

Grade:

Leganés, Madrid, Spain, November 2th, 2016

A mis padres, a mi hermana y a mis abuelos.
Fuente de amor e inspiración.
Espejo en el que verse reflejado.

Abstract

The use of robots in human-like environments requires them to be able to sense and model unstructured scenarios. Thus, their success will depend on their versatility for interacting with the surroundings. This interaction often includes manipulation of objects for accomplishing common daily tasks. Therefore, robots need to sense, understand, plan and perform; and this has to be a continuous loop.

This thesis presents a framework which covers most of the phases encountered in a common manipulation pipeline. First, it is shown how to use the Fast Marching Squared algorithm and a leader-followers strategy to control a formation of robots, simplifying a high dimensional path-planning problem. This approach is evaluated with simulations in complex environments in which the formation control technique is applied. Results are evaluated in terms of distance to obstacles (safety) and the needed deformation.

Then, a framework to perform the grasping action is presented. The necessary techniques for environment modelling and grasp synthesis and path planning and control are presented. For the motion planning part, the formation concept from the previous chapter is recycled. This technique is applied to the planning and control of the movement of a complex hand-arm system. Tests using robot Manfred show the possibilities of the framework when performing in real scenarios.

Finally, under the assumption that the grasping actions may not always result as it was previously planned, a Bayesian-based state-estimation process is introduced to estimate the final in-hand object pose after a grasping action is done, based on the measurements of proprioceptive and tactile sensors. This approach is evaluated in real experiments with Reflex Takktile hand. Results show good performance in general terms, while suggest the need of a vision system for a more precise outcome.

Resumen

La investigación en robótica avanza con la intención de evolucionar hacia el uso de los robots en entornos humanos. A día de hoy, su uso está prácticamente limitado a las fábricas, dónde trabajan en entornos controlados realizando tareas repetitivas. Sin embargo, estos robtos son incapaces de reaccionar antes los más mínimos cambios en el entorno o en la tarea a realizar.

En el grupo de investigación del Roboticslab se ha construido un manipulador móvil, llamado Manfred, en el transcurso de los últimos 15 años. Su objetivo es conseguir realizar tareas de navegación y manipulación en entornos diseñados para seres humanos. Para las tareas de manipulación y agarre, se ha adquirido recientemente una mano robótica diseñada en al universidad de Gifu, Japón. Sin embargo, al comienzo de esta tesis, no se había realzado ningún trabajo destinado a la manipulación o el agarre de objetos. Por lo tanto, existe una motivación clara para investigar en este campo y ampliar las capacidades del robot, aspectos tratados en esta tesis.

La primera parte de la tesis muestra la aplicación de un sistema de control de formaciones de robots en 3 dimensiones. El sistema explicado utiliza un esquema de tipo líder-seguidores, y se basa en la utilización del algoritmo *Fast Marching Square* para el cálculo de la trayectoria del líder. Después, mientras el líder recorre el camino, la formación se va adaptando al entorno para evitar la colisión de los robots con los obstáculos. El esquema de deformación presentado se basa en la información sobre el entorno previamente calculada con *Fast Marching Square*. El algoritmo es probado a través de distintas simulaciones en escenarios complejos. Los resultados son analizados estudiando principalmente dos características: cantidad de deformación necesaria y seguridad de los caminos de los robots. Aunque los resultados son satisfactorios en ambos aspectos, es deseable que en un futuro se realicen simulaciones más realistas y, finalmente, se implemente el sistema en robots reales.

El siguiente capítulo nace de la misma idea, el control de formaciones de robots. Este concepto es usado para modelar el sistema brazo-mano del robot Manfred. Al igual que en el caso de una formación de robots, el sistema al completo incluye un número muy elevado de grados de libertad que dificulta la planificación de trayectorias. Sin embargo, la adaptación del esquema de control de formaciones para el

brazo-mano robótico nos permite reducir la complejidad a la hora de hacer la planificación de trayectorias. Al igual que antes, el sistema se basa en el uso de *Fast Marching Square*. Además, se ha construido un esquema completo que permite modelar el entorno, calcular posibles posiciones para el agarre, y planificar la movimientos para realizarlo. Todo ello ha sido implementado en el robot Manfred, realizando pruebas de agarre con objetos reales. Los resultados muestran el potencial del uso de este esquema de control, dejando lugar para mejoras, fundamentalmente en el apartado de la modelización de objetos y en el cálculo y elección de los posibles agarres.

A continuación, se trata de cerrar el lazo de control en el agarre de objetos. Una vez un sistema robótico ha realizado los movimientos necesarios para obtener un agarre estable, la posición final del objeto dentro de la mano resulta, en la mayoría de las ocasiones, distinta de la que se había planificado. Este hecho es debido a la acumulación de fallos en los sistemas de percepción y modelado del entorno, y los de planificación y ejecución de movimientos. Por ello, se propone un sistema Bayesiano basado en un filtro de partículas que, teniendo en cuenta la posición de la palma y los dedos de la mano, los datos de sensores táctiles y la forma del objeto, estima la posición del objeto dentro de la mano. El sistema parte de una posición inicial conocida, y empieza a ejecutarse después del primer contacto entre los dedos y el objeto, de manera que sea capaz de detectar los movimientos que se producen al realizar la fuerza necesaria para estabilizar el agarre. Los resultados muestran la validez del método. Sin embargo, también queda claro que, usando únicamente la información táctil y de posición, hay grados de libertad que no se pueden determinar, por lo que, para el futuro, resultaría aconsejable la combinación de este sistema con otro basado en visión.

Finalmente se incluyen 2 anexos que profundizan en la implementación de la solución del algoritmo de *Fast Marching* y la presentación de los sistemas robóticos reales que se han usado en las distintas pruebas de la tesis.

Acknowledgments

First, I want to thank my advisors, Luis and Santiago for believing I could make it and guide me through the thesis. I do not want to forget Dolores, who should probably appear as co-advisor, it is difficult to come across anyone more helpful than her. In general, I find all the members of RoboticsLab make it a good place to work and learn, and that makes all easier. I feel lucky for having such a nice working environment. I want to specially thank Javi for all the talking, the travelling, the cycling and projects we had together, you are a good lab mate and friend to have.

A big part of this thesis has been developed abroad, making it a very special experience. I want to thank Kawasaki-sensei and Mouri-sensei for inviting me to Gifu University and introducing me to Japanese culture, I'll always remember my experience in Japan. I also want to thank Máximo for inviting me to work with his group at DLR, that felt like an intensive robotics course. You might not know, but I really learnt a lot working around you.

Outside robotics, there is also a lot of people to remember now. It's difficult to express the gratitude to my family for their support. All my close friends deserve a whole paragraph each of them; you are just awesome and I really enjoy my time with you. Thanks for being around, for making me a little bit how I am and for your encouragement all these years.

The last person to appear is probably the most important along these years, Naiara. Thanks for the unconditional support I always find in you and for being by my side during this journey. Also, for many other things, I'll just keep them between us.

It's been a very long way until I got here. However, it is not the destiny, but walking the path which makes it interesting. This path has allowed me to visit very beautiful places and to meet very nice people in the way. I just cannot name all of you here, but all of you helped in some way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Human expectancy for robots' performance and appearance is highly influenced by futuristic science fiction on literature and movies. When hearing the word *'robot'*, most usual images that are brought to anyone's mind include a nicely design, high skilled, and, very commonly, human-like machine which does not fail in whatever it is asked to do, ever. But those mechanisms are not real yet. In fact, the kind of robots that are actually most often used, industrial manipulators, are very poorly skilled and not looking as a human being. They, at least, usually have a nice design. From the point of view of the tasks they perform and how they do them, there are two essential properties. First, they are able to perform one, and only one, task for which they have been programmend previously. They do it very fast and accurately, but their ability to adapt to the slightest change in the task in almost null. And second, they rarely rely on sensors which measure the environment. This means that their ability to react and response in a different manner due to environmental changes is also almost null. In other words, their main lack is versatility.

Non-industrial robots are usually designed to work in human environments and perform common human daily tasks. These scenarios are usually unstructured, either due to the design of the place or to human intervention. This means that, just for moving around, robots cannot only rely on a perfectly modelled environment, but they actually need to be aware of unexpected obstacles before making any decision, or react to them if the decision was previously made, what we usually call 'a change of plans'. Besides, human environments tend to change over time. The most basic change is the one provoked by humans moving around. From the robot's point of view, they not only are obstacles but they also move all the time in a not very predictable way. Other challenging situations may be: using different locations for the same object every day or performing the same task at a different location; all of them make it a must for the robot to have good sensing capabilities. In order to sum up, in human-like environments robots need to: sense, understand, plan and perform; and this has to be a continuous loop.

As examples of the general pipeline for robots working in human environments, we can think on a couple of common daily tasks: opening doors and grasping a cup of water. Some problems we can find during these manipulation tasks are: the unexpected weight of the cup depending of the amount of water it has; the difficulty of detecting a transparent object if it is made of glass; the many different shapes of doorhandles which lead to a broad amount of manners to open a door. This list can be made as long as desired. From the manipulation point of view, human experience on performing the tasks starts with a very lengthy training which, as a result, gives us a high level of awareness of our capabilities and a high degree of understanding the outcome of the actions performed, both after failing or success, and therefore, makes us unaware of the tremendous complexity they require. However, there are many difficulties for a robot to perform them. If we want to understand them, we can

think of ourselves trying to perform them as if we were a child (before the training happened) or as if we were somehow physically or mentally impaired, or even both at the same time, like it usually happens with aging. When combining some of this issues together, the likeliness of success in the task decreases exponentially.

Grasping is a particular case of the tasks robots are designed for, commonly forming part of more complex manipulation functions. Mechanical hands are developed to give robots the ability to grasp objects of varying geometric and physical properties. The price of using a mechanical hand is the complexity of the overall system. Anthropomorphic designs usually involve from four to five fingers, like the examples in figure 1.1, and therefore the whole hand-arm system can sum up to more than twenty-five degrees of freedom (DOFs), which makes analytical studies much harder. On the other hand, this high number of DOFs increases the possible ways of performing a task to a point in which it is not obvious how to choose the best, or simply a good one, among all of them. Besides this, one of the biggest problems in the grasping task (and maybe in robotics in general) is uncertainties. In all the stages of a control loop of a versitle robotic system (as described above), several level of unaccuracies and errors are accumulated. The information from the environment provided by the sensors, the execution of a given plan by the actuators, or even the planning itself, all of them include a centain level of uncertainties.

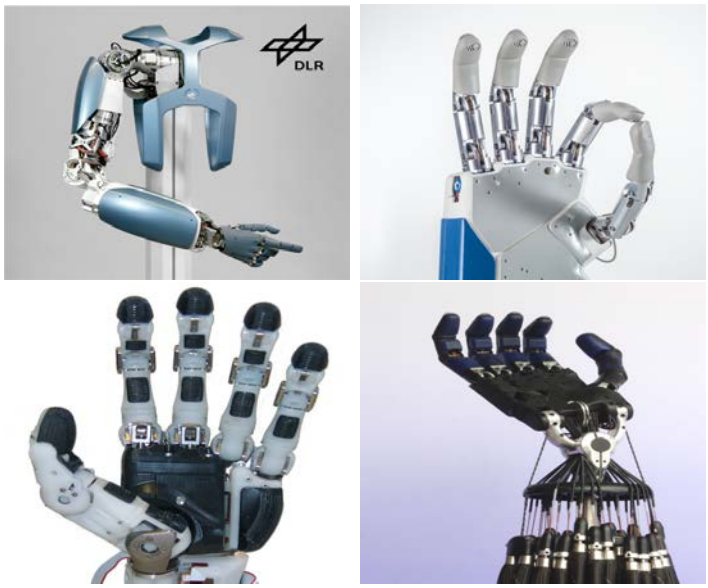


Figure 1.1: Examples of anthropomorphic robotic hands. Top-left: DLR hand-arm system. Top-right: Azzurra hand. Bottom-left: Elu2 hand by Elumotion. Bottom-right: Shadow hand.

In this thesis, two parts of the grasping problem are addressed: first, given a table

top scene, how to make and execute a plan in the high dimensional space of a hand-arm system in order to grasp an object; second, once this object is grasped by the hand and assuming that the grasping action may have not been perfectly performed, how to evaluate the resulting in-hand object to hand pose, so that we are able to plan further than that, is studied.

1.1 Context and motivation

Mobile manipulators are nowadays a common research platform in many laboratories around the world, either because they are now more affordable, or because research groups have accumulated enough knowledge to build one. Robotics Lab research group at Carlos III University of Madrid (UC3M) has been building and improving such a robotic platform for more than ten years. It is denominated ManfredV2 [1] and it can be seen in figure 1.2. It consists of a differential type mobile base with two DOFs, an anthropomorphic light-weight arm with six DOFs and a very small and simple gripper as an end-effector. The mobile base encloses a computer with all the electronic components needed to operate and the batteries. ManfredV2 is equipped with a vision sensorial system (both colour and depth based), two-dimensional (2D) and three-dimensional (3D) range lasers and encoders in each DOF. In 2012, the end-effector was updated with Gifu-Hand III [2], which is shown in figure 1.2. It is a human-like hand, with four fingers and one thumb. Each finger has four DOFs, three of which are actuated and one of them under-actuated. The thumb has four fully actuated DOFs. The mechanical design integrates the actuators (DC motors) and the sensors (incremental encoders), although an outer control and power system are needed.

Figure 1.2: On the left, a general view of ManfredV2 mobile manipulator. On the right, the robotic hand Gifu Hand III.

The final purpose of the team working with ManfredV2 is to develop techniques

to make it robust, reliable, accurate, and safe to work with when it is performing daily life tasks in human environments. ManfredV2 is already capable of performing localization [3, 4], path planning [5, 6], navigation and obstacle avoidance [7, 8] or object recognition and reconstruction [9, 10] among other tasks. These skills have been acquired throughout the years and most of them are still part of on-going research activities. However, at the start point of this thesis, manipulation skills were limited to path planning and following with the arm [11, 12]. One objective of this thesis is to explore new ways to enhance the grasping capabilities of ManfredV2.

As mentioned before, grasping tasks include solving several problems of different categories. First, it requires sensing the environment to detect the object to be grasped and other objects around it. From these objects, their pose and an estimated geometrical structure are needed. The techniques used in this part of the thesis are all of them based on previous projects in Robotics Lab [10], or other software available [13, 14]. A model of the environment is then built using this information in order proceed with the next stages. The second step consists on selecting a grasping pose for the end-effector which allows the robot to grasp the object. For this part, basic data-driven based algorithms have been developed for grasp selection. The generation of possible grasps is based on geometrical properties of the object to be grasped, while the grasp pose selection is based on the arm capabilities and the avoidance of collisions with obstacles in the scene.

Then, path planning techniques are needed to decide how to get to the selected grasp pose. Here, an important problem arises. The total hand-arm system consists on twenty-six DOFs, twenty-two of which are actuated. We are, obviously, in front of a high dimensional path planning problem. Several techniques have been used to solve this kind of problem before [11, 15]. However, a complete different approach is presented in this thesis. If the path planning and control problems applied to a formation of robots are considered, they actually represent a different example of the same problem, dealing with high-dimensional spaces due to the accumulation of robots. In this case, it is more straight forward to think that one could try to solve it for one of the robots, and then focus the effort in the coordination. This is called the leader-followers approach, in which the followers are organized to follow the leader while keeping a geometric formation. An example on how to reproduce such a thinking in a 2D formation based on the Fast Marching Method (FMM) is given in [16, 17]. This thesis presents the adaptation of this technique to the 3D world. Furthermore, a framework in which the robotic hand-arm system is treated as a robots formation, is also developed. Therefore, path planning is only done for the leader, the wrist, while the hand movements are controlled based on the leader's position and keeping a geometric formation highly constrained by the mechanical structure of the hand. This way, a reduction of the dimensionality of the problem is achieved in order to make it easier and faster to find a solution.

Finally, grasping the object is commonly part of a more general task. However, in order to carry on, it is important to know the pose of the object inside the hand, so that further decisions can be taken. One could assume that the result of the grasping action finishes exactly as it was planned or simulated. This is a very hard assumption though, since the accumulation of inaccuracies of actuators, sensor noise and simplifications in simulation or modelling, generally produces a different result than expected. For this reason, an in-hand pose estimation of the object is presented, in order to be able to have a more precise knowledge of the result of the grasping action.

1.2 Document structure

The document is divided into four chapters and two appendixes apart from this introduction. The first three chapters contain the specific methodologies and algorithms used to solve the problems above mentioned. Thus, a detailed formulation and state of the art is given in each one of them. They also contain their own results section.

Chapter 2 introduces the methodology for path planning and control of 3D robot formation based on Fast Marching Squared. Several simulations are analyzed to prove the usefulness of the method.

Chapter 3 presents the framework built on top of chapter 2 in order to treat a robotic hand-arm system as a robots formation in order to perform a grasping action.

Chapter 4 treats the in-hand object pose estimation issue. A methodology based on tactile sensing and particle filtering is presented.

Then, chapter 5 outlines the main conclusions extracted from this thesis and its results, pointing out the most promising ideas in this area.

Finally, appendices A and B are attached. The former contains a detailed description of the Fast Marching Method and its use in the path planning problem. The latter presents the robotic platforms used in the experimental part of chapters 3 and 4.

Chapter 2

3D Robot Formations

2.1 Introduction

In recent years, research on multi-robot systems (MRS) in 3D environments has increased exponentially due to the price drop of unmanned aerial vehicles (UAV) and the advent of micro-aerial vehicles (MAV) as a popular robotic test bed in the robotics community. Furthermore, in many tasks the use of multi-agent systems increases in overall mission performance, flexibility and robustness without augmenting the capacity of each UAV unit [18]. Based on these characteristics, typical applications for MRS are: exploration [19], search and rescue [20], surveillance [21, 22], and many others. In order to achieve a good performance in any of these applications several research topics need to be addressed, such as: modelling and control of such agents [23], collision avoidance [24], mapping and state estimation with such agents [25] or formation control and planning [26].

In formation control, a group of coordinated robots have to perform a specific task trying to keep a certain geometric configuration. The coordination of the robots is one of the essential topics in the field of MRS. When operating in close proximity, limited space or in a collaborative task, the movements of the robots have to be planned and coordinated efficiently. In real world applications, this synchronised navigation requires a computationally fast solution so that the velocity of the motion can be maintained.

There are many issues to be considered when designing a controller for mobile robot formation, such as: the stability of the formation, the controllability of the shape patterns or the safety of the movements. All these issues need to be addressed while the formation is moving along different scenarios. Therefore, the configuration of the formation should evolve over time in order to meet the different constraints. This evolution can be quantified using different metrics that characterize it. Only a few approaches have been found in the literature on how to do this measurement, mainly because of its difficulty to be generalised. However, some of the most used criteria for 2D formations can be easily extended to 3D formations. [27, 28] use two similar performance metrics for the evaluation of their experiments: average position error, which is the average displacement of each robot from their predefined positions in the formation, and percentage of time in formation, that reflects the amount of time in which the robots keep the geometry of the formation. [29, 30] use a formation evaluation criterion called uniform dispersion, which evaluates if the same distance is kept between all neighbour robots with a maximum tolerance of ε_d.

In this chapter, the average position error of each robot along the path is computed. Moreover, since the environments used in the simulations are quite challenging, an analysis based on the distance of the robots to obstacles is also considered so that the safety of the presented algorithm can be evaluated.

When looking into the literature describing how to control the evolution of the formation, many different strategies can be found. In [31] the multi-agent coordination problem is studied under the framework of control Lyapunov functions. The main idea is that every robot has a control Lyapunov function, and there exists a global Lyapunov function, for the whole formation, which is a weighted sum of individual Lyapunov function of each robot. The main drawback is the mathematical complexity needed to obtain satisfactory results. Other works use an approach based on potential fields which are combined in order to get the desired behaviour of the formation [32]. The major problem of these approaches is the existence of local minima. In other behaviour-based approaches [33] each robot has basic primitive actions that generate the desired behaviour in response to sensory input. Possible schemas include collision avoidance and goal seeking. Virtual structure introduced by [34] is defined as a collection of agents that maintain a desired geometric configuration. The algorithm has three main steps: the virtual structure is aligned with the position of the robots, then a trajectory for every agent is obtained, and finally each robot follows its own path. This approach is capable of maintaining a highly precise formation and has mainly been used for satellite formation control [35]. However, due to its high computational complexity is very difficult to apply it to multi-UAV control. In the case of leader-followers approach, a common scheme is the model predictive controller [36] which was recently introduced for holonomic robots [37]. The major focus of most methods is mainly to maintain the formation based on pre-planned paths and static environment assumption.

In this research a leader-followers approach has been used, extending the initial work presented in [38]. In this strategy, a robot (that could be virtual) is designated as the leader of the formation and follows a trajectory towards the goal point. At the same time, follower robots are positioned behind it according to a default geometry shape that can be deformed, within a given range, trying to accommodate to the environment conditions [39, 40, 41]. An advantage of this strategy lies on its simple implementation since no feedback loop from the followers to the leader is needed. This is due to the leader's behaviour, which can be considered as egoistic, since in the path planning phase the formation itself is not considered, and in the motion phase it also does not take into consideration the follower robots. This approach leads to paths that are optimal for the leader, while they might not be as good for the followers. Although a bigger robot describing the formation (instead of punctual ones) could be considered in planning phase, so that narrow passages are avoided, our approach considers that any path can be accomplished if the deformation scheme is good enough. Besides, the algorithm can find a solution even in the presence of very constrained situations. Therefore, our deformation scheme allows the robots to adapt their paths in the situations where their movements are constrained by the environment, which is one of the key points of the algorithm. Another important

characteristic is that it depends on the leader's motion, so it is very important to have very good path planning and tracking because once the leader loses its path, its error is fully propagated to all the followers and both the mission and coordination objectives could fail.

In the presented approach, the leader's path is calculated using the Fast Marching Square (FM^2) path planning method, which ensures obtaining very safe and smooth paths [5]. For the followers, a predefined geometry evolves dynamically, while the leader is covering the path, based on a velocities map calculated as a first step of FM^2. The manner the deformation is produced is based on the algorithm described in [42], which shows an easy way to deal with robot priorities when going through very narrow environments, with a very low mathematical complexity. Besides, a 3D modelling of dynamic obstacles based on a distance local map is introduced. Also, a slight modification of the algorithm is introduced so that all the robots are able to take into account the moving obstacles while covering the path. Finally, a quantitative analysis on the performance of the algorithm with respect to the amount of deformation needed to avoid the obstacles, and the safety of the solution in terms of distance to obstacles are presented. Furthermore, qualitative results are shown for different complex scenarios.

The next sections of this chapter are organised as follows. Section 2.2 introduces the Fast Marching Square path planning method. In section 2.3 the application of FM^2 to the robot formation problem in static and dynamic scenarios is explained. In section 2.4 simulation results for difference scenarios and a qualitative analysis of the performance of the algorithm are shown. Finally, in section 2.5 conclusions are addressed.

2.2 Fast Marching Square path planning method

The FM^2 is based on the Fast Marching Method (FMM), which was proposed by Sethian [43] in order to solve the Eikonal equation on Cartesian grids. This differential equation models an isotropic front propagation. Although a deep explanation of the FMM is made in appendix A, a general view is given next as an introduction of FM^2.

2.2.1 The Fast Marching Method

Light rays travelling through different materials follow the path which is less time-consuming, according to Fermat's principle [44]. This is an interesting concept in robotics since, if we are able to model how light waves travel in space, we can easily compute the fastest path between two points in space. This is exactly what we can do with FMM. If the goal point of a path is considered as the source of a light

wave, and the FMM is computed over all the free space of a map, the time of arrival of the wave at every cell in the map is obtained. Moreover, considering the set of all cells representing free space and the time as last coordinate, we can create a Lyapunov surface in which the level curves are isochronal, and the Fermat's paths are orthogonal to them. This means that it is impossible for the method to have local minima. Graphically, this can be seen in figure 2.1, which represents the funnel potential of the light wave propagation with a constant refraction index.

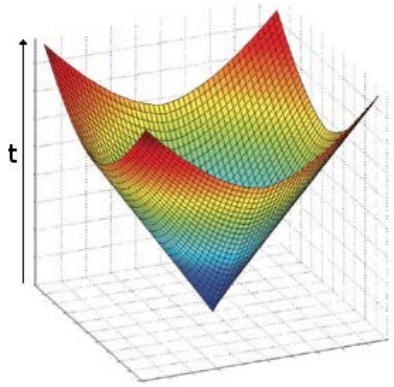


Figure 2.1: Lyapunov surface created when the propagation of the wave starts in one point and the refraction index is constant. Time is used as the vertical axis.

Mathematically, the propagation of the light is given by the Eikonal equation 2.1. In [43] a first order approximation of the solution for this equation was proposed and called the Fast Marching Method.

Let us assume a 2D map, where $\mathbf{x}_{\text{ij}} = (x_i, y_i)$ represents the point $\mathbf{x} = (x, y)$ in the space corresponding to a cell (i, j) of the grid (for the 2D case) in relation to a Cartesian referential. Besides, let $T(\mathbf{x})$ be the arrival time function of the front wave and $F(\mathbf{x})$ the velocity of the wave propagation. Moreover, we assume that a wave starts propagating at time $T(\mathbf{x}_o) = 0$, and the velocity $F(\mathbf{x})$ is always non-negative. The Eikonal equation defines the time of arrival of the propagating front wave, $T(\mathbf{x})$, at each point $\mathbf{x}$, in which the propagation speed depends on the point, $F(\mathbf{x})$, according to:

$$\left|\nabla T(\mathbf{x})\right| F(\mathbf{x}) = 1, \ \mathcal{X} \subset \mathbb{R}^N \tag{2.1}$$

Discretising the gradient $\nabla T(\mathbf{x})$ according to [45] it is possible to solve the Eikonal equation at each point $\mathbf{x}_{\text{ij}}$, as follows:

$$\begin{aligned} T &= T_{i,j} \\ T_{\text{x}} &= \min(T_{i-1,j}, T_{i+1,j}) \\ T_{\text{y}} &= \min(T_{i,j-1}, T_{i,j+1}) \end{aligned} \tag{2.2}$$

$$\max\left(\frac{T - T_{\text{x}}}{\Delta_{\text{x}}}, 0\right)^2 + \max\left(\frac{T - T_{\text{y}}}{\Delta_{\text{y}}}, 0\right)^2 = \frac{1}{F_{\text{ij}}^2} \tag{2.3}$$

The FMM consists on solving $T(\mathbf{x})$ for every point of the map starting at the source point of the wave where $T(\mathbf{x}_o) = 0$. The following iterations solve the value $T(i, j)$ for the neighbours of the points solved in the previous one. Using as an input a binary occupancy grid map, the output of the algorithm is similar to the distance transform, but in this case is continuous, not discrete. These distances have the meaning of the time of arrival of the expanding wave at every point in the map, since the velocity of the wave at every cell is known (and in this case constant). After applying the FMM, gradient descent can be used from any point of the map of distances to obtain a path towards the source of the wave, which works as a goal point. This is valid only if one wave has been employed to generate the map of distances. The main advantage of this method is that the path obtained is optimal in distance, like the example in figure 2.2.

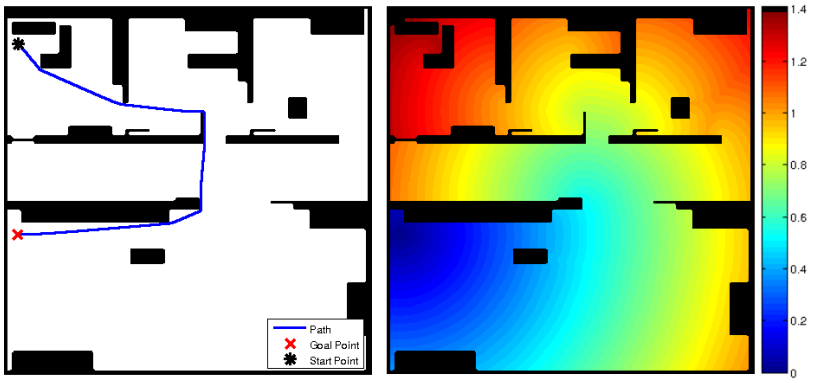


Figure 2.2: Example of a path obtained with the FMM. The left side shows the original map and the path calculated. In the right there is the map of distances computed with FMM, these distances are expressed in the same units as the cell resolution.

2.2.2 Fast Marching Square Method

As we can see in figure 2.2, although optimal in distance, it is obvious that the path produced by FMM is not safe in terms of distance to the obstacles, nor feasible in terms of the abruptness of the turns that the path requires. These problems lead us to consider using the Fast Marching Square method (FM^2) as path planner. The FM^2 [7] solves these two main disadvantages. It is based on applying the FMM twice.

Let us now consider a map in which obstacles are labeled as 1 and free space as 0. The Fast Marching Method can be applied to this map considering all the obstacles to be wave sources. When FMM was applied in section 2.2.1, there was just one wave source (at the target point). In this case, all the obstacles are considered a source of the wave, and hence, several waves are being expanded at the same time. If this technique is applied to the occupancy map in figure 2.2 (left), the resulting map can be seen in figure 2.3. This map can be interpreted in several different manners. It represents a potential field of the original map, as cells get further from obstacles, the computed T_{i} value is greater, as in the case of a distance field/map. It can also be interpreted as a *slowness/velocities map*: T_{i} value can be considered to be proportional to the maximum allowed speed of the robot at each point, which leads to allowing lower speeds when the cell is close to the obstacles, and greater when it is far away from them. In fact, a robot whose speed at each point is given by the T_{i} value will never collide, as $T_{\mathrm{i}} \to 0$ when approaching the obstacles.

Figure 2.3: Example of a *slowness map*, result of computing the FMM to an occupancy map and considering all the obstacles as sources of a wave.

The second time the FMM is applied, the resulting slowness map after the first step is applied, is considered as a map of velocities for the second wave expansion. This means that the second time the wave is propagated, the velocity at which it moves forward may be different at every point in the map. Furthermore, this velocity is proportional to the distance to the closest obstacle, meaning that the wave is faster when it is far from obstacles. This produces important differences in the path that

is computed, as it can be seen in figure 2.4. On the right, there is the time of arrival function result of applying the FM^2 algorithm over the occupancy map on the left of the figure. Besides, the computed path is drawn on the original map, it is easy to appreciate that the path avoids getting close to obstacles in a smooth manner.

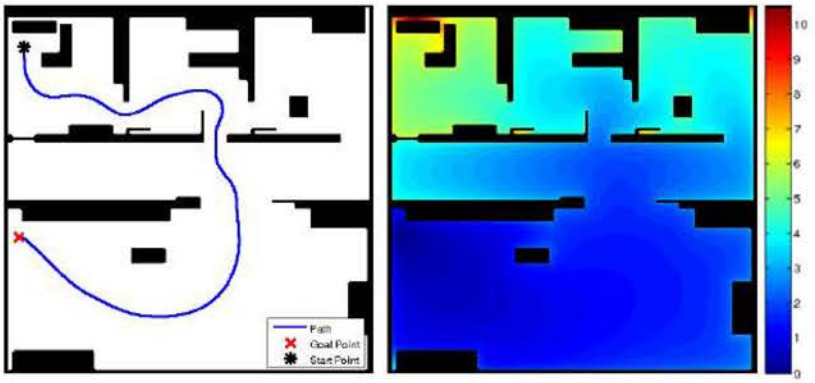
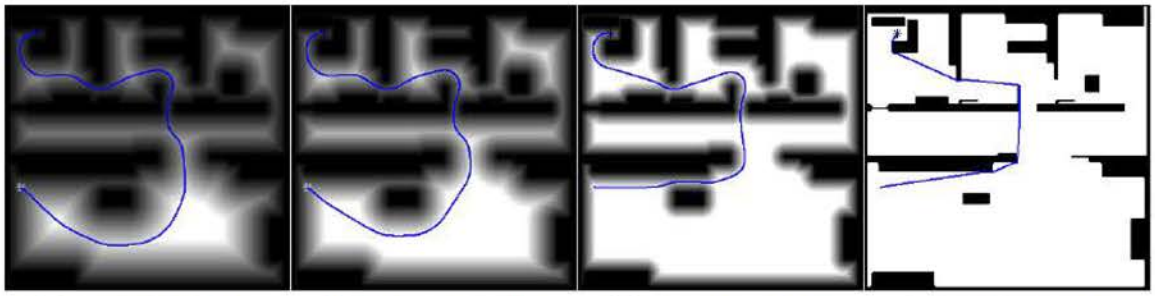


Figure 2.4: Example of a path obtained with the FM^2. The left side shows the resulting path of the algorithm in the initial occupancy map. In the right side there is the time of arrival map.

Furthermore, assuming that the *slowness map* contains values between 0 and 1, relative to the maximum allowed velocity, it is possible to trim (saturate) this map of velocities. With this small modification, the safety and smoothness of the computed paths is still ensured (except for saturation values close to 0), while obtaining trajectories closer to the optimal one in terms of distance. Examples of the variation of the path when the saturation value is modified are shown in figure 2.5. As the saturation value becomes lower, the clearance between the path and the obstacles is smaller



(a) Saturation: 0.75 (b) Saturation: 0.5 (c) Saturation: 0.25 (d) Saturation: 0

Figure 2.5: FM^2 saturated variation: modification of the path depending on the saturation value.

The proposed FM^2 algorithm has several properties which make it very good for

path planning purposes [17, 41]. The most important ones include:

- *No local minima*: as long as only one wave is employed to generate the time of arrival map, FM^2 ensures that there is a single global minimum at the source point of the wave (goal of the path).
- *Completeness*: the method finds a path if it exists and notifies in case of no feasible path.
- *Smooth trajectories*: the planner is able to provide a smooth motion plan which can be executed by the robot motion controller. In other words, the plan does not need to be refined.
- *Reliable trajectories*: it provides safe (in terms of distance to obstacles) and reliable trajectory (free from local traps). This avoids the coordination problem between the local collision avoidance controllers and the global planners, when local traps or blocked trajectories exist in the environment.
- *Fast response*: if the environment is static, the map of velocities is calculated only once. Since the FMM can be implemented with a complexity order of $O(n)$ [46], building the map of velocities is a fast process.

2.2.3 3-Dimensional Fast Marching Square

Since the FM^2 algorithm is based on the standard FMM, it is extensible to more than 2D, as it is done in this work on 3D robot formations planning. The algorithm works exactly in the same way as the 2D version, with the only difference that the front wave becomes a spatial curve. Also, the time response is a little slower since the size of the grid that models the environment is much bigger. Despite this, all the properties of the FM^2 remain in a n-dimensional environment. This is the main fact that leads us to use this algorithm as path planner. Figure 2.6 shows a lateral view of an example trajectory computed with FM^2 and performed by the arm of ManfredV2.

2.3 Robot Formation planning with FM^2

The algorithm described next is an extension of [16, 42], in which the FM^2 path planning method is used to control 2D formations in different scenarios. A general idea of the algorithm was presented in [38], however, still some improvements are necessary for adapting this method to dynamic and more complex environments.

This algorithm is based on a leader-followers scheme, which is used to control the robot formation. In this scheme, the reference pose for the follower robots is defined

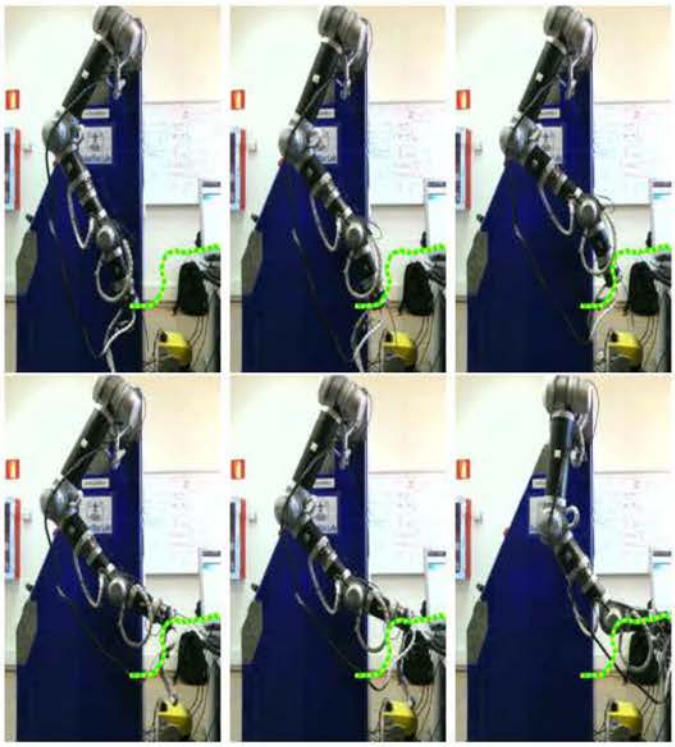


Figure 2.6: The robot ManfredV2 performing a 3D trajectory computed using FM^2.

by a geometric shape for the formation. This means that the goal poses along the path of each follower are a function of the leader's pose. How this shape is controlled is introduced in section 2.3.1. The leader of the formation can be a robot, a person or even a virtual leader. It is important to say that the path planning algorithm considers all the robots in the formation as punctual objects. Besides, the path for the leader is computed in an egoistic way, not taking into account the other robots in the formation. This might lead to situations where the followers may move too close to obstacles, or even collide with them. In order to avoid these situations, new partial goals are computed for the follower robots using the shape deformation scheme explained in section 2.3.2 while the leader covers its path. This approach takes the advantage of only computing the path for the leader of the formation, saving computation time in the path planning phase.

Both, the path planning and the evolution of the shape of the formation, are based on FM^2. FM^2 provides a two-level artificial potential field which repels the robots from obstacles and has no local minima. Integrating the potential given by FM^2 and the shape deformation scheme for the follower robots, each robot has at each moment one single potential attracting it into the objective, but repelling it from obstacles and other robots. The main requirement when integrating all the potentials is to

do it in a way that does not create local minima. Figure 2.7 shows the steps of the algorithm on a triangle-shaped robot formation. Although it is a 2D shape, it has been chosen because it is easier to understand the behaviour the followers to avoid colliding with obstacles and among themselves. The images are ordered starting from the top left one, which shows the different components of the robotic formation: the leader and its path, the followers, and their partial goals. Then, the top-right image presents a default triangle-shape formation (in green), note that it is expressed as a function of the tangential and perpendicular vectors ($\vec{v}$ and $\vec{u}$ respectively) of the leader's path and depends on distance $d1$ and $d2$, which are non-constant, so they may change along the path. Next, bottom left image shows the partial goals of the follower robots (blue circles) for a different step of the algorithm. The shape of the formation is kept when the leader turns. Finally, bottom right image shows that, when the new objectives (dashed blue line and circle) of the follower robots are close to an obstacle (grey shadow), these objectives are updated in order to move away from obstacles deforming the original triangle shape by changing the value $d2$ of the top follower into $d2'$ (blue line and circle).

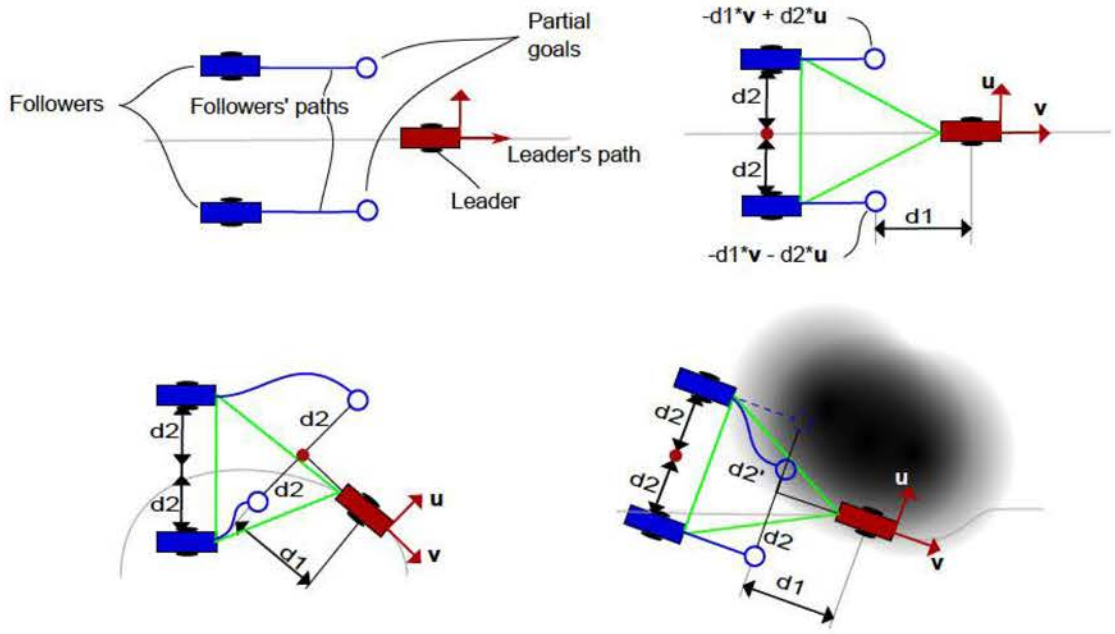


Figure 2.7: Top left - Main components of the robot formation algorithm. Top right - Reference geometric definition of a simple, triangle-shaped robot formation, note that the definition is based on vectors $\vec{u}$ and $\vec{v}$ (tangential and perpendicular to the path, respectively). Bottom left - Behaviour of the partial goals depending on the leader's pose. Bottom right - Behaviour of the partial goals depending on the obstacles in the environment.

2.3.1 Robot pose coordination

The aforementioned figure 2.7 shows the basic scheme for controlling a robots formation in a 2D scene. However, when the formation performs in a 3D environment, a problem arises when dealing with the orientation of the followers. While in 2D, perpendicular and tangent vectors can be used because they are unique, this cannot be directly applied in 3D, since there are an infinite number of perpendicular vectors to the tangent to the path. All these vectors are contained in a plane that is perpendicular to the tangent vector, as depicted in figure 2.8.

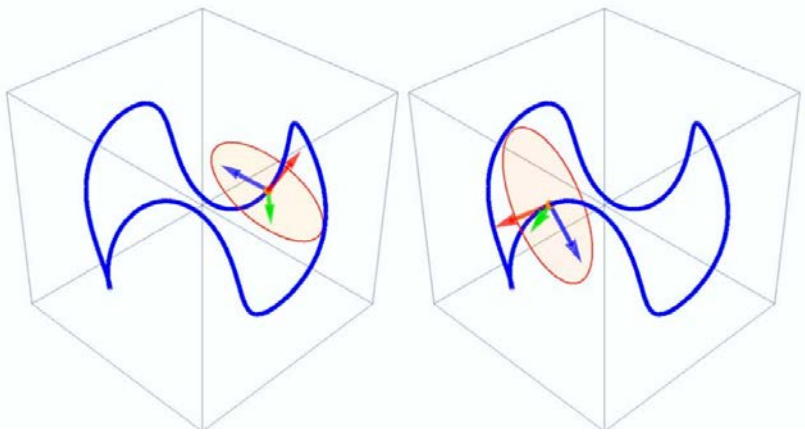


Figure 2.8: The red vector represents the tangent to the trajectory and the red circle is the perpendicular plane to this vector.

In order to be able to use the proposed robot formation planning method in 3D, a third reference has to be specified so that the reference shape of the formation can be determined in a unique form. In this case, a generalization using a relative reference based on the Frenet-Serret formulae [47, 48] is proposed. It consists on extracting the local characteristics of the path as a third reference, this way it allows the formation to be environment-independent when defining the reference geometry.

The Frenet-Serret formulae are used to describe the kinematic properties (velocity, curvature and torsion) of a particle which moves in a three-dimensional Euclidean space, $\mathbb{R}^3$. Let $\mathbf{r}(t)$ be a parameterization of a continuous, differentiable curve C in a Euclidean space $\mathbb{R}^3$. Then, its curvature, $\kappa(t)$, is defined as:

$$\kappa(t) = \frac{||\mathbf{r}'(t) \times \mathbf{r}''(t)||}{||\mathbf{r}'(t)||^3} \tag{2.4}$$

being $\kappa(t)$ a non-negative number. Then, for space curves with non-zero curvature, it is possible to define a *higher-level curvature* called torsion, $\tau(t)$, which is associated

with the third time derivative $r'''(t)$. It describes the *twisting* of the curve, and it is defined as:

$$\tau(t) = \frac{det[\mathbf{r}'(t)\mathbf{r}''(t)\mathbf{r}'''(t)]}{||\mathbf{r}'(t) \times \mathbf{r}''(t)||^2} \tag{2.5}$$

Curvature and torsion define the local characteristics of the curve $\mathbf{r}(t)$. Then, let us denote $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ as the unit tangent vector, the unit normal vector and the unit binormal vector respectively. The Frenet-Serret formulae defines them as:

$$\mathbf{T}'(t) = \kappa(t)|\mathbf{r}'(t)|\mathbf{N}(t) \tag{2.6}$$

$$\mathbf{N}'(t) = -\kappa(t)|\mathbf{r}'(t)|\mathbf{T}(t) + \tau(t)|\mathbf{r}'(t)|\mathbf{B}(t) \tag{2.7}$$

$$\mathbf{B}'(t) = -\tau(t)|\mathbf{r}'(t)|\mathbf{N}(t) \tag{2.8}$$

The Frenet-Serret frame along the curve is defined by the collection of the three vector functions $\mathbf{T}(t)$, $\mathbf{N}(t)$ and $\mathbf{B}(t)$ which satisfy the following fundamental relations:

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|} \quad \mathbf{N}(t) = \frac{\mathbf{T}'(t)}{|\mathbf{T}'(t)|} \quad \mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t) \tag{2.9}$$

The graphical representation of the Frenet trihedron at a certain point of a curve has already been shown in figure 2.8, where the red vector is $\mathbf{T}(t)$, the blue vector is $\mathbf{N}(t)$ and the green vector represents $\mathbf{B}(t)$. The main advantage of using the Frenet trihedron is that among the infinite possible vectors perpendicular to the tangent vector, the one chosen is in the direction of the curvature $\mathbf{N}$ (or normal acceleration). Furthermore, the direction of the vectors in the frame changes continuously, this is an important property when using it as a reference for the shape of the formation, in order to have a smooth evolution. So, from a geometrical point of view, by combining vectors $\mathbf{T}$, $\mathbf{N}$ and $\mathbf{B}$ any shape can be given to the formation.

Based on the Frenet trihedron of a 3D path, figure 2.9 shows an example of a robot formation shaped as a quadrangular base pyramid. The left side shows the different components of the formation: leader (red) and follower (blue) robots are represented by rectangles. The leader path is drawn as a grey line and attached to the leader, the Frenet trihedron is represented by a 3-axes frame in which the tangential vector is drawn in red, the normal vector in blue and the binormal vector in green. Also, blue and green circles (the difference in the colours is just for visualization purposes) represent partial goals of the follower robots. On the right, the partial goals (forming a quadrangular base pyramid) are defined as a function of the Frenet trihedron vectors and parameters $d1, d2$ and $d3$. $d1$ defines the height of the pyramid, while $d2$ and

$d3$ define the base. As in the 2D example explained before, these parameters may change while the leader covers its path, adapting the shape of the formation to the environment.

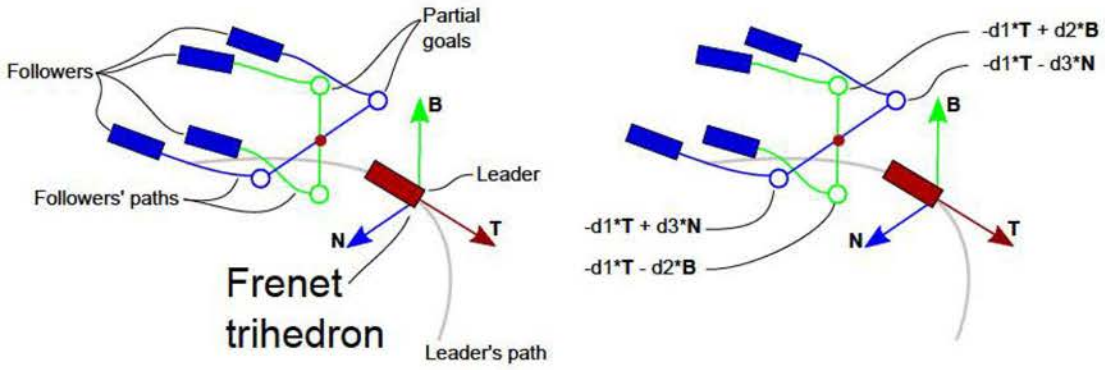


Figure 2.9: Definition of the formation based on the Frenet trihedron. Robots are represented by rectangles, red for the leader and blue for the followers. Attached to the leader, the Frenet trihedron is represented by 3-axes frame. The leader's path is represented by a grey line. Partial goals for the follower robots are computed with respect to the Frenet trihedron, as indicated in the right part of the figure.

2.3.2 Distance based shape deformation

As explained before, although the formation is predefined by geometric relationships, this shape cannot be exactly the same along all the path because this could lead to collisions. Therefore, when the formation gets close to obstacles, the positions of the robots in the formation should be modified in order to adapt the initial shape to the obstacles in the environment. The proposed method achieves this by modifying the distances which define the default geometry. The manner these distances are modified will determine the behaviour of the formation.

If the quadrangular base pyramid in figure 2.9 is taken as a default formation, then figure 2.10 shows simple deformation rules than can be used to adapt this shape to the environment. Note that distances in the vertical axis are named after their corresponding vector in the Frenet trihedron. Thus, dTi is the distance along the $Tangential$ (T) axis for every follower, which corresponds to $d1$ in figure 2.9. Besides, the value max represents the distances defining the desired default formation for the robots, the original pyramid. While min is the minimum value those distances can take when the geometric shape is deformed.

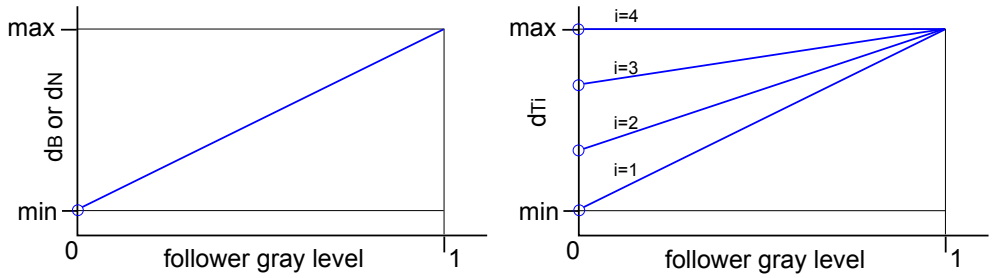


Figure 2.10: Followers' partial goals modification based on linear functions.

These two simple rules cause two kind of behaviours of the follower robots. For the distances in directions parallel to the *Binormal* (B) and *Normal* (N) axis, namely d_B and d_N, the behaviour is the same: since a lower less grey level means the robot is closer to obstacles, the distances in these directions have to decrease towards the minimum. Therefore, the follower robots get closer to the leader's path, which is collision free. In the formation presented in figure 2.9, this means that the base of the pyramid becomes smaller.

However, in very narrow environments, this deformation rule can make the robots suffer an excessive contraction towards the leader's path, forcing them to move too close to each other or even collide. In order to avoid this kind of situations, the concept of priority among the robots is introduced. This is done by modifying the distance in the direction parallel to the *Tangential* (T) axis, d_T. When a different variation of this distance is set for each robot, it makes the followers to be located at a different distance at the back of the leader. According to figure 2.10, follower 4 will not modify its distance, while follower 1 will be the one which gets closest to the leader. Therefore, highest priority is given by a higher slope in this function. This deformation rule allows the formation to contract as much as needed so that the obstacles are avoided since the maximum deformation would make the pyramid to change its shape into a line, in which the order of the robots is set by their priority rule.

In the previous paragraphs two simple rules have been introduced to control a pyramid-shaped formation in narrow environments. However, the behaviour of the followers can be modified by simply setting different functions to define the shape changes. For the experimental results in section 2.4, the values of the parameters in the rules presented have been chosen empirically. It is important to note that they mainly depend on the size of the robots, since they set how much close they can get among them, and on the abitlity of the robots, since more steep functions which govern the shape change require a more agile dynamic response of the robots.

2.3.3 Mobile obstacles modelling based on a distance map

Common environments for UAVs are not static in the real world scenarios. In order to take into account mobile obstacles, a local distance map model of the obstacles based on FMM has been developed. This technique deals, not only with the position of the obstacles, but also with their uncertainty which could exist due to, for example, sensor noise.

Let us consider a mobile obstacle O_i with position $P_i = [x_i, y_i, z_i]$ and uncertainty U_i, where i is the number of obstacles and U_i is $0 \leq U_i \leq 1$, being $U_i = 1$ the maximum uncertainty. This uncertainty level models the noise and inaccuracies of the sensors used for obstacle detection and modelling, since it is used to create areas around the objects in which we cannot assure whether there is an obstacle or not. Note that the aim is not to create an accurate error model for a given sensor, but to create a sensor-independent algorithm. This uncertainty could be understood as a *safety* parameter. Besides, the obstacles can be modelled differently, depending on the desired degree of accuracy. In the most simple case, a bounding box [49, 50] containing the obstacle is created. Other possibility can be modelling the objects using one or several superquadric functions [51, 52]. In any of these cases, a local 3D grid map around each mobile obstacle is created. In this map, we model the obstacle by filling with ones the voxels that correspond to the shape of the chosen model of the obstacle. The accuracy of this model is limited by the resolution of the grid.

As an example, we will consider a punctual obstacle as can be seen in figure 2.11, top-left. The rest of the voxels of the local map will have zero values, meaning that they are free locations. Now, if we compute the FMM over this map, we will get a distance transform function. Figure 2.11, top-right, shows the central slices of the 3D grid which contains an example of the distance transform in which the locations around the obstacle have a greater grey-scale value than those further from it. The drop of the grey level value as the voxels are further from the obstacle can be interpreted as the uncertainty, in which higher values mean a higher likelihood of the voxel to be occupied by the obstacle. Then, the distance map can be saturated with the value given by U_i to control the level of uncertainty. Then, the values in the map are scaled between 0 and 1. Figure 2.11, bottom, shows the final step of the local map computation for two different uncertainty values, left $U = 0.3$ and right $U = 0.6$. It can be seen how the uncertainty value U_i allows us to control the area of influence of the obstacle. It is important to note that the furthest positions in the grey scale images should be white, but they have been plotted in black to help the visualization.

Finally, the obstacle must be introduced in the environment so that it is taken into account by the leader and the followers in the formation. This is done by applying equation 2.10 among the voxels of the local map of the obstacle and the corresponding

voxels around its position P_i in the velocities map (first step of FM^2) of the environment. This results into areas around the obstacles that have a velocity value close to zero where the obstacle is located, which are then avoided with the use of the FM^2 path planning method.

$$W_j = \min(W_j, obstacle) \tag{2.10}$$

where j corresponds to the voxels where the function is applied.

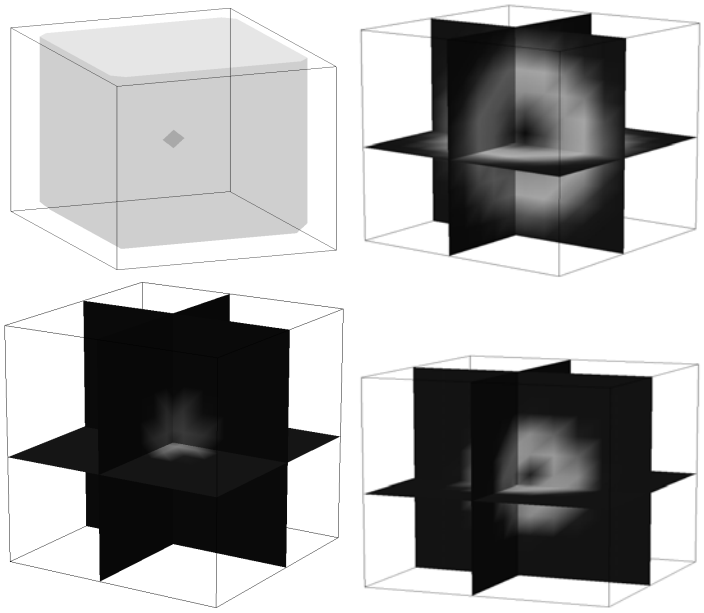


Figure 2.11: Computation of the distance based model of an obstacle with different uncertainty value. Top left - Local 3D map with a punctual obstacle. Top right - Central slices of the 3D grid representing the distance transform function of the obstacle. Bottom images: final result for different uncertainties, left $U = 0.3$ and right $U = 0.6$. It is important to note that the furthest positions in the grey scale images should be white, but they have been plotted in black to help the visualization.

2.3.4 Formation planning algorithm

In previous sections the low-level modules of the proposed framework have been detailed. Next, these steps are put together into a high-level algorithm, depicted in figure 2.12. This algorithm assumes that the following information is available at the beginning: a voxelized binary model of the environment; the initial and goal positions

of the leader; the default shape of the formation; the size, uncertainty value, initial position and velocity along the simulation (or the full path) of the mobile obstacles in the environment. It is important to note that a static goal is being considered. For dynamic goals, a simple replanning approach would not be enough since goal modifications could require the formation to carry out complex maneuvers. Once all the data is provided, it is processed following the steps shown in figure 2.12, which are explained next:

(a) The environment map $\mathbf{W_0}$ is read as a binary map, in which 1 means that the position is occupied by an obstacle and 0 means it is free space. This map is common for all the robots in the formation (both the leader and followers).

(b) The first potential $\mathbf{W}$ (distance or slowness map) is calculated applying the FMM to the binary map $\mathbf{W_0}$, according to the FMM 1st step of the FM^2. This potential is also shared by all the robots.

(c) The second potential $\mathbf{T}$ is calculated applying the FMM on the potential $\mathbf{W}$ from the goal point to the leader's initial position.

(d) A path for the leader is calculated applying gradient descent on the potential $\mathbf{T}$, according to the FM^2 method. In the case of static environments, this path never changes.

(e) Once the leader of the formation has its path to the goal position, a control loop for the formation is executed while the leader covers this path. In this loop, each cycle represents a step of the robots' movement. For every cycle, new partial goals are computed for each follower. It consists of the next steps:

   I Since we may have mobile obstacles, for each cycle t the position of every obstacle is updated both in the binary map $\mathbf{W_0}$, for visualization purposes, and in the first potential $\mathbf{W}$, so that they are taken into account by the robots in the formation, as explained in section 2.3.3.

   II The second potential $\mathbf{T}$ is updated, so that the changes in $\mathbf{W}$ are taken into account.

   III A new path for the leader is calculated applying gradient descent on the new potential $\mathbf{T}$, according to the FM^2 method. Since the environment is dynamic, the previous path that was computed for the leader of the formation may not be safe any more, and so it needs to be recomputed every time any of the obstacles move. Since the full path is only computed for one robot, it can be done online.

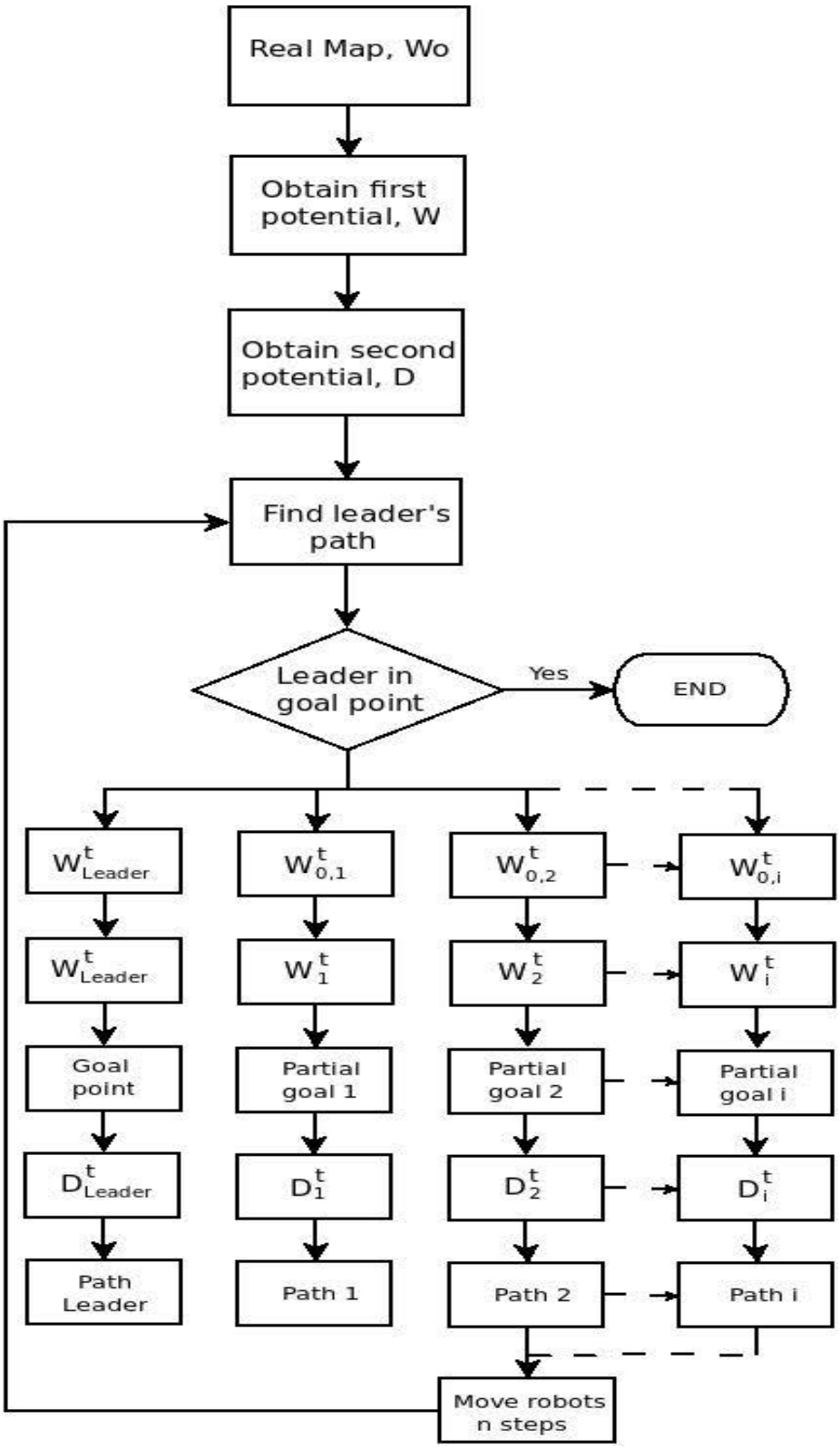


Figure 2.12: Robot formation path planning algorithm using FM^2.

IV According to the new pose of the leader and the default formation geometry, the partial goal (x_{gf}, y_{gf}, z_{gf}) is calculated for each follower f (where f represents each follower in the formation). Initial partial goals are computed with the predefined geometry referenced to the Frenet's trihedron, as explained in section 2.3.2.

V The level in the distance map, $\mathbf{W}$, of the partial goal position of each follower f is used in order to recompute its goal, as detailed in section 2.3.1. Therefore, the shape of the formation is deformed so that the robots move further from obstacles.

VI The second potential $\mathbf{T}_i^t$ is locally calculated for each follower f, applying the FMM to velocities map $\mathbf{W}$. The goal point each follower uses is their partial goal computed on the previous step. The low computational cost of FM^2 allows us to do this without compromising the refresh rate.

VII The path is calculated for each follower f. This path is the one with the minimum distance with the metrics $\mathbf{W}_i^t$ and it is obtained applying gradient descent on the potential $\mathbf{T}_i^t$.

VIII All the robots move forward following their paths until a new iteration is completed.

This algorithm, together with the formation coordination and the obstacle modelling approaches, allow us to significantly reduce the complexity of the path planning problem of a multi-robot formation. As remarks, it is important to keep in mind that the full path planning is only done for the leader, without taking into account the follower robots, reducing the time consumption of the process. Then, while the path is being covered, the formation is able to adapt to the local characteristics of the environment.

2.4 Results

According to the previously introduced algorithm, different simulations have been carried out in order to prove the validity of the proposed method. All of them are based on Matlab® implementations of the algorithm running on a Linux-based operating system in a 2.2 GHz dual-core PC (only one core was used). Simulations perform only kinematic simulation, meaning that we do not consider any specific dynamic model of the UAVs, and that a perfect path following algorithm is assumed for all the UAVs. In all the cases, the map of the environment is known in advance. Besides, both the initial and the final points of the trajectory are given, and the paths are calculated with the FM^2 algorithm. To calculate the partial goals of the followers, a default pyramid shape is previously set, as shown before. This shape evolves by

modifying height of the pyramid and the size of the base. In very narrow situations, in which making the size of the base really small could lead to dangerous situations, the followers will evolve into a convoy-like formation increasing the safety. These changes are produced as a function of the distance to obstacles value of the position of the followers, as explained in section 2.3.2. In the case of the moving obstacles, both the initial positions and their velocity are also known, although some uncertainty is introduced as explained in section 2.3.3.

While in previous works, only visual results of the simulation were given, in this research a quantitative analysis of the performance of the formation planning algorithm is shown. Two different criteria have been proposed: first, the distance of each robot to the closest obstacle along all the path is evaluated. This is a common criteria used in mobile robotics planning and gives us an idea of how safe is a path with respect to the distance to the obstacles in the environment. In the simulations, it is important to study the safety of the results since the algorithms are tested in very narrow environments and with dynamic obstacles. The second metric measures how much the shape of the formation changes with respect to the initial geometry. It consists on evaluating the average displacement of the robots in the formation, from their default relative position to their final pose, determined by the shape deformation algorithm introduced in this chapter. It is computed using equation 2.11:

$$E = \frac{1}{N} \sum_{i=1}^{N} dist(p_i, d\_p_i) \tag{2.11}$$

where N is the number of follower robots, $dist$ corresponds to the Euclidean distance in 3D, p_i is the computed position for follower i and d_p_i is the predefined position of the same robot.

The first simulation environment is shown in figure 2.13. It consists of two rooms connected by means of a very narrow passage in which the formation does not fit without some amount of shrinking. In this example, we can see how the concept of priorities is inserted in the algorithm since the formation evolves towards a convoy-like geometry in which the followers are assigned with a different priority value. This is done by modifying the height of the pyramid (distance in the tangential direction) with a different value for each robot, so that it is possible to get extra space between followers. At the same time, the size of the base changes in a way that makes the followers to almost converge into a straight line.

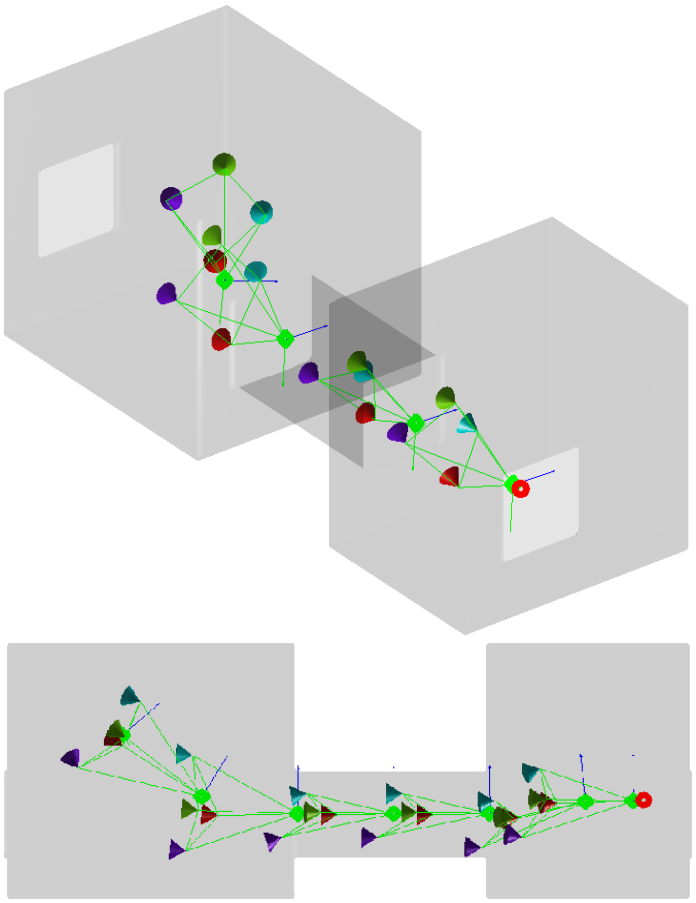


Figure 2.13: Example of a motion sequence in a narrow corridor, leveraging the priorities introduced in the algorithm. Top: 3D view. Bottom: top-view which allows to see how the robots change their relative position in the direction of the motion.

Besides, quantitative analysis according to the metrics aforementioned is shown in figure 2.14. The top image shows the distance to the closest obstacle for each robot. Although there is a legend in the image, the colour of each line is the same as the colour assigned for the robot in figure 2.13, and the leader's information is plotted in blue. Horizontal axis corresponds to the steps of the path, being 0 and 64 the start and final pose respectively. Vertical axis shows the distance to the closest obstacle. This value is normalised, meaning that a 0 value corresponds to a collision with an obstacle, and a 1 value means the maximum clearance a robot can have in the whole environment. We can observe that, at the beginning, the robots move to a more open space (clearance grows) which corresponds to the movement towards the centre of the first room. Then, the distance gets smaller while the robots approach to

the passage, and reaches a minimum value while moving through it. Finally, it grows again while entering in the second room, and becomes smaller when approaching to the goal, which is close to the wall. In terms of deformation (figure 2.14 bottom), we can see how its evolution is inversely proportional to the distance to the obstacles. This means that when the space in narrower, the shape needs to be deformed in order to fit in the space, which is the expected behaviour.

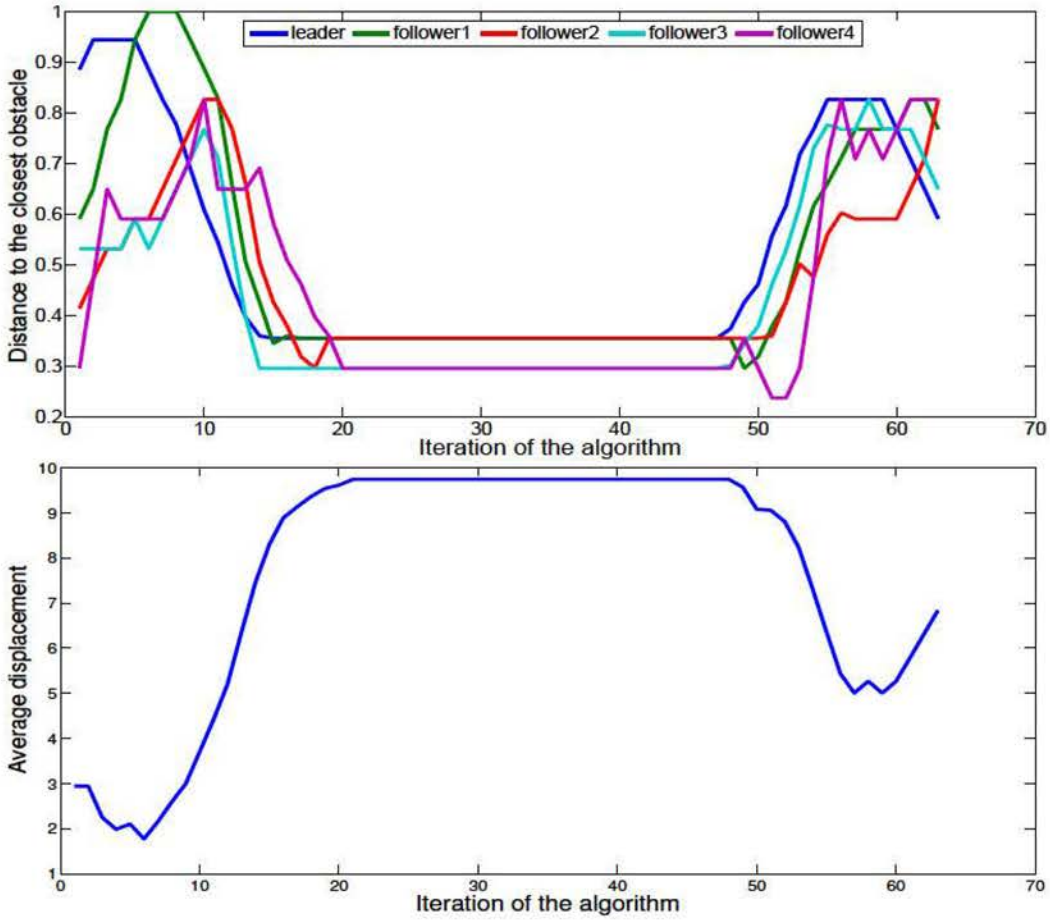


Figure 2.14: Quantitative analysis of scenario 1. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

Figure 2.15 depicts the results of the same algorithm run in a map of one of the laboratories of our university, in order to show that the proposed algorithm performs well in realistic environments. In this case, the formation is composed by two followers placed in parallel along the normal vector to the leader. Therefore, the formation is actually the 2 robots navigating with the leader placed in the straight line that

connects the followers. In this case, the leader is considered virtual in order to show the flexibility of the approach. The only deformation allowed is along the normal vector.

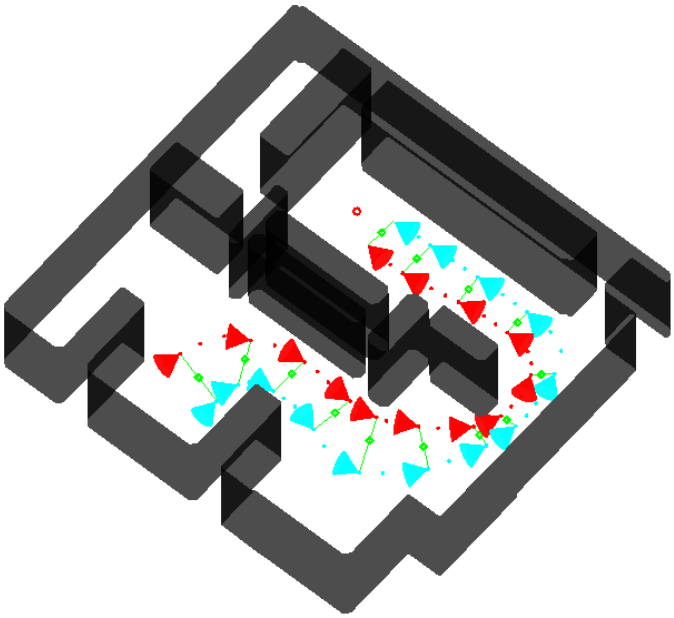


Figure 2.15: Example of a motion sequence in a map of one of our laboratories. Two robots (blue and red) move in parallel to a virtual leader (green).

The corresponding quantitative results are shown in figure 2.16. The distances between robots and obstacles are safe and there is not any collision. This performance is remarkable given how cluttered the scenario is.

The next scenario is shown in figure 2.17. In this case a sequence of positions along the path can be seen. The environment is composed by two big empty rooms connected by a narrow window. Moreover, in the second room there are two mobile obstacles, represented by black balls. These obstacles are located in the same point of the XY plane (floor plane), but at different heights. They move parallel to the floor towards the wall between the rooms. As we can see in the sequence, as explained in section 2.3.3, the path is different for each step since a new one is calculated at the beginning of it. One can appreciate how during the firsts steps, the path surrounds the obstacles using the space on the left side (from the robots point of view), however, after the robots have passed through the window, at which the formation shrinks in the same way as in the passage in the previous environment, the obstacles move forward and come closer to the wall. This implies that there is more free space on the right side of the obstacles, so the path of the leader changes the side the robots

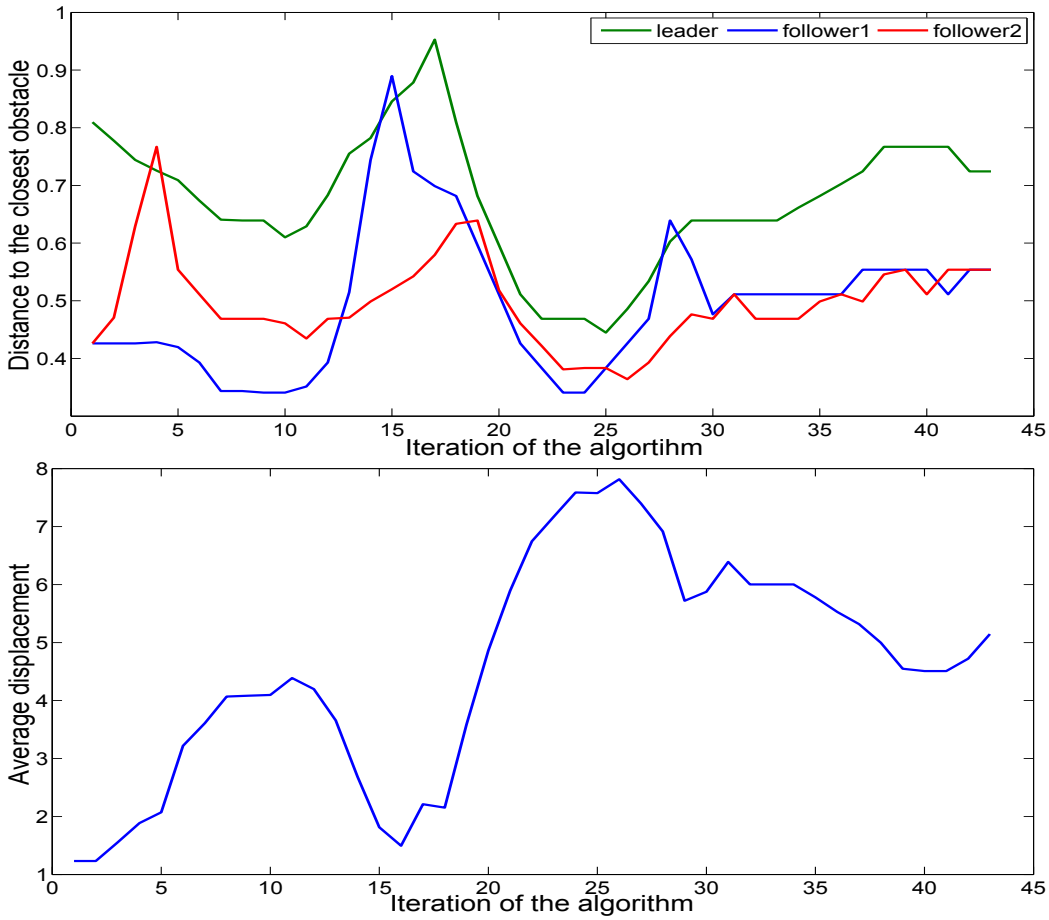


Figure 2.16: Quantitative analysis of scenario 2. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

use to avoid the obstacles. Nevertheless, even though the leader eludes the obstacles because of this change, the followers still have to skip the collision. As one can see from the 5^{th} to the 8^{th} images in figure 2.17, the follower plotted in light blue can avoid the moving obstacles just by following the same shape deformation algorithm used for static environments. This happens because the obstacles are included in the velocities map as explained in section 2.3.3, so that they cause the same behaviour as static obstacles. It is important to note, that in the 7^{th} image, although the light blue follower seems to collide with the obstacle, it is just a matter of the perspective used in the visualization.

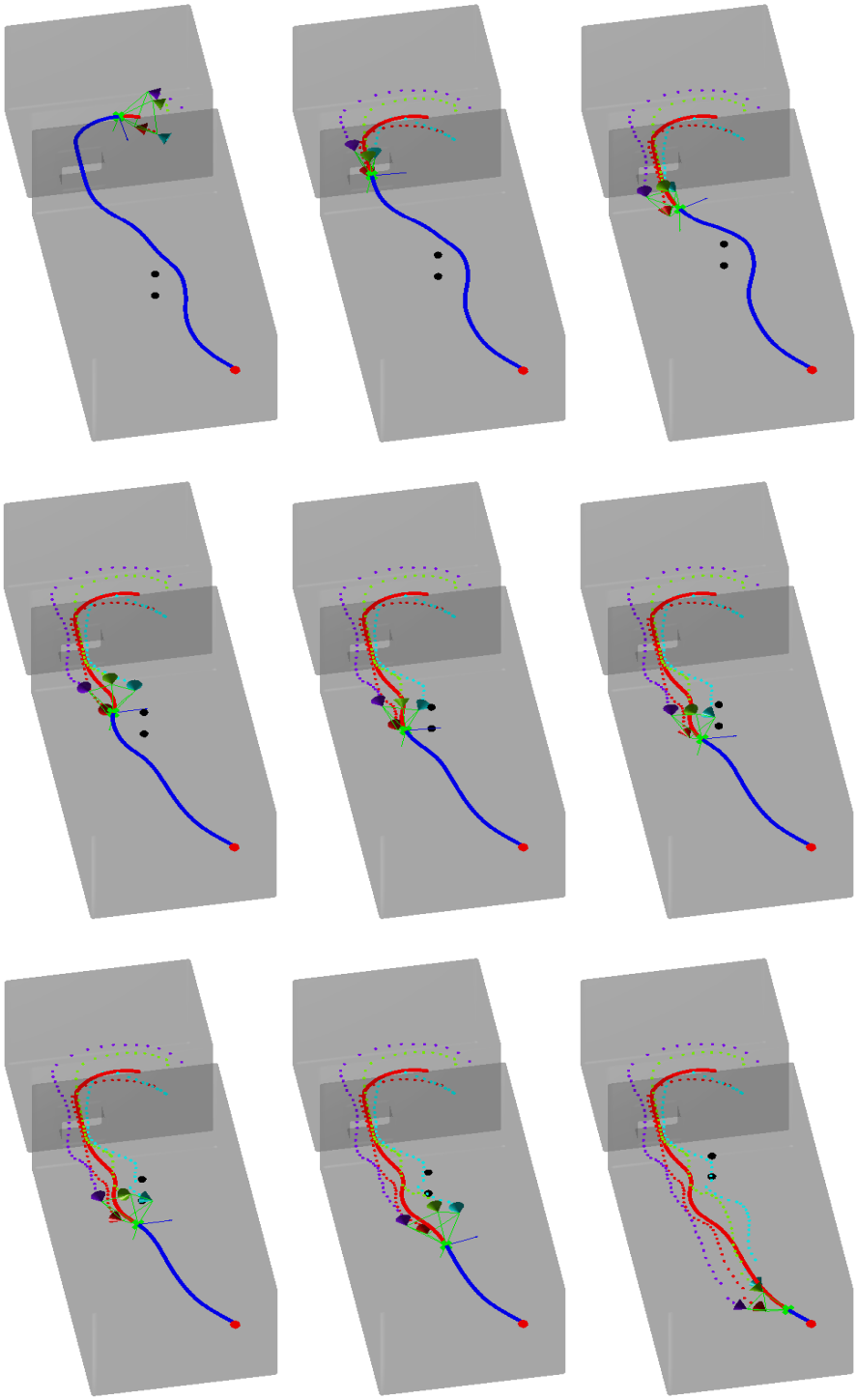


Figure 2.17: Environment with mobile obstacles. The temporal evolution starts in the top-left image and ends in the bottom-right image.

Moreover, the same quantitative analysis has been applied to this scenario and it is shown in figure 2.18. As before, the colours in the figure correspond to the same colour of the robots in the simulations. In the case of the distance to obstacles, it can be seen that the value of all the robots decreases towards 0 when they go through the window, and increases afterwards. The obstacles do not affect significantly to this values because of the change in the path and the shape deformation movement. In the case of the shape deformation result, it can be seen that there is a very big change in the shape when the robots go through the window and a slight one when they need to avoid the obstacles and arrive to the goal, which is close to the walls of the environment.

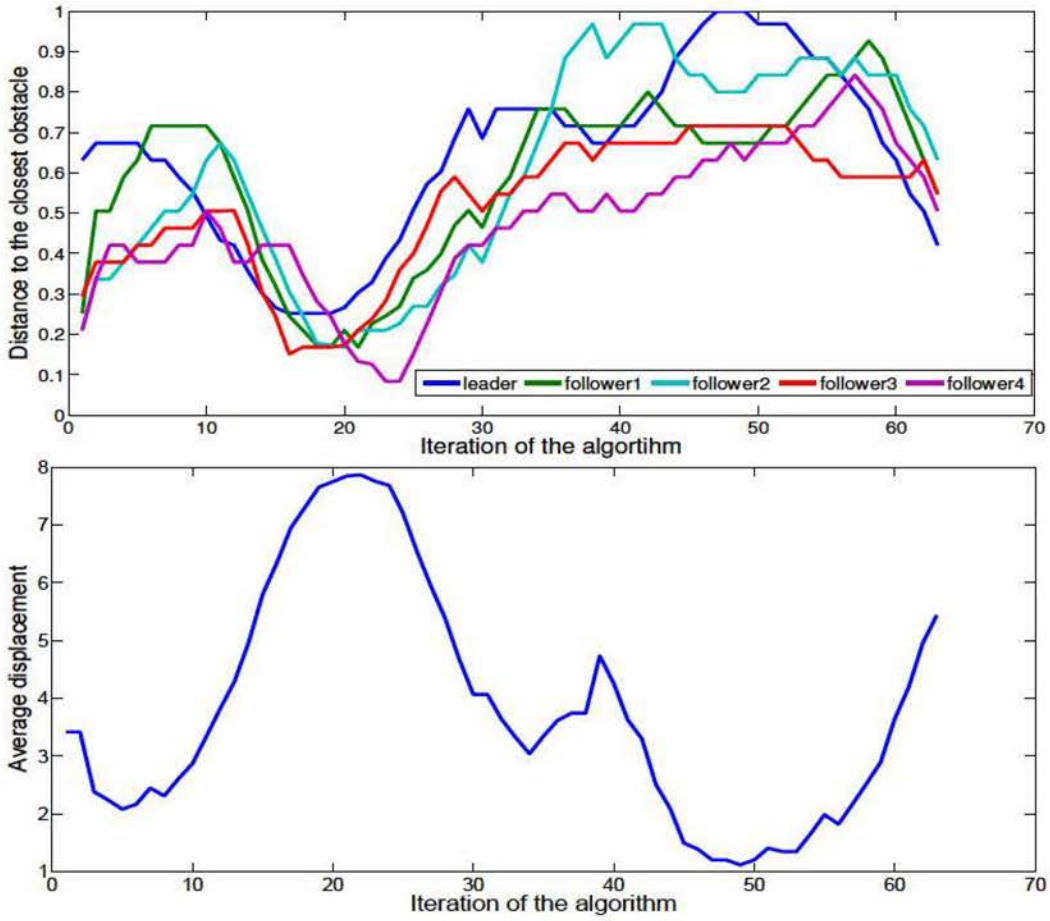


Figure 2.18: Quantitative analysis of scenario 3. Top: distance to the closest obstacle. Bottom: average displacement of the formation with respect to the default shape of the formation.

Finally, table 2.1 shows the mean computation times μ and their standard deviation σ for the three previous experiments. For all the experiments, the average and standard deviation have been computed over ten different runs of the algorithm. The first column shows the time elapsed in leader's path computations. These times depend on the environment chosen, specially on the amount of obstacles in them. They are the most time consuming part of the algorithm. The second column includes the formation algorithm times per iteration, which are really fast and could be run in real time with frequencies higher than 1000Hz. Third column contains the times elapsed for all the followers to compute the path to its partial objectives per iteration. It depends on the number of followers, but one can see that for 4 followers the computation time is still around 500ms, so it can be computed online. Note that this can be easily parallelised since every follower computes its own path. Be aware that the first and third columns are expressed in seconds, while the second is in milliseconds. Also note that the algorithm is completely deterministic, which means that the computations times suffer low variation among runs. Another important remark is that iteration times for experiments 1 and 2 are practically the same, despite the fact that experiment 1 includes 4 followers and experiment 2 only 2. However, experiment 3 needs more time per iteration since the mobile obstacles have to be taken into account as explained in section 2.3.3. Again, this process could be easily parallelised for every follower reducing considerably the required time.

Table 2.1: Computation times for the leader's paths, formation algorithm iterations and followers' paths.

	Leader's times (s)		Iteration times (ms)		Followers' times (s)	
	μ	σ	μ	σ	μ	σ
Exp. 1	0.47	0.01	0.22	0.09	0.49	0.03
Exp. 2	0.99	0.1	0.25	0.09	0.44	0.04
Exp. 3	0.65	0.04	10	2	0.52	0.02

2.5 Conclusions

This research broadens previous works presented by adapting the formation shape control scheme to 3D environments, and includes the necessary changes to apply the algorithm to non-static environments. Besides, it introduces a novel approach to model mobile obstacles in the environment in a way that they are, later on, easily treated by the obstacle avoidance algorithm explained. Finally, quantitative analysis of this approach has been carried out. The results explain how the formation evolves avoiding obstacles while covering the planned path. The tests show that the proposed

shape deformation method, in combination with the FM^2 path planner, is robust enough to manage autonomous movements through an indoor static and dynamic 3D environment.

It is important to note that the algorithm is both conceptually and mathematically very simple since it relies on basic natural behaviours such as light movement. Furthermore, the same method is applied in the planning phase for both the leader and the followers. Besides, the deformation schema for the geometry of the robot formation is based on basic 3D translations and rotations which can be computed very fast using standard algebraic tools.

Results show that the proposed algorithm is able to manage difficult environments, modifying the formation when it is necessary. In addition, this approach allows us to include any number of robots in the formation, by only setting the desired position with respect to the leader or the other robots. The introduction of function-based geometry deformation is very powerful since it permits setting complex behaviours to the followers by simply modifying these functions. As an example of this, the use of priorities in the formation is showed. These functions can be modified dynamically, an important property that is worthy to explore in the future.

Dealing with 3D robot formations, future work using FM^2 is also related to testing this method with other type of formations in which, for example, the leader is not always in front of the team. Also, future simulation should include dynamics in order to prove that the computed paths are smooth enough to be applied to real robots. Besides, the possibility of a goal position change should be included in the high-level formation algorithm, so that the algorithm allows to have dynamic goals and the replanning is able to compute the necessary maneuvers to achieve them.

In general, it is shown that it is possible to decouple a high dimensional path planning problem into two simpler ones: 3D path planning for one robot and formation control and coordination for the others. This idea is the ground for the construction of next chapter, which seeks to find an answer to the following question: Is it possible to do the same for an hand-arm system when performing the grasping action?

Chapter 3

From Robot Formations to Grasping Control

3.1 Introduction

A common use of robotic assistants, whether at home, factories or in space, will not be realized without the ability to grasp typical objects in human environments. The human hand, the most versatile end-effector known, is capable of performing a wide range of tasks with an impressive dexterity. In an attempt to match its abilities, a wide variety of anthropomorphic robotic hand designs with at least four fingers have been proposed over the last years. Examples include: the Utah/MIT Hand [53], the Robonaut hand [54], the Gifu Hand [55], the Shadow Hand [56], the DLR Hand [57] or the MA-I Hand [58]. Several discussions on the designs have already been presented [59, 60].

These models often include human-like kinematics and simplified motor or tendon based actuation. However, the increase in potential capabilities has come at the cost of similar increase in the complexity of usage. As the number of degrees of freedom of a robotic hand approaches the human hand, effective autonomous algorithms that can handle high-dimensional configuration spaces are required in order to take advantage of the new designs [61]. Therefore, despite the advanced features of these mechanical hands, one of the remaining problems in order to obtain a good outcome from them is the autonomous determination of their movements.

3.1.1 Grasp Synthesis

When a robot is about to grasp a given object, the first decision to be taken is where to position the fingers on the object so that the object is grasped safely. This problem is called grasp synthesis. The variety of approaches used to solve it often differ on how to compute the grasp positions, or on how to measure the quality of those positions. Recent reviews [62, 63] establish two different methodologies to solve this problem: analytic and empirical. Analytic methods are those which attempt to construct force-closure grasps with a multi-fingered robotic hand, usually formulated as a constrained optimization problem over criteria that measure equilibrium, stability and the possibility to exhibit a certain dynamic behaviour [64]. A grasp is then defined by the grasp map that transforms the forces exerted at a set of contact points to object wrenches [65]. The criteria used to compute this grasp map can be based on geometric, kinematic or dynamic formulations. A review on analytic techniques towards grasp synthesis can be found in [66]. Empirical approaches rely on sampling grasp candidates for an object and ranking them according to a specific metric. The sampling process is usually based on some existing grasp experience generated by a human operator, in simulation or on a real robot. For this reason these methods are also called knowledge-based approaches.

It has been recognised that classical metrics based on analytic formulations, such as the widely used ϵ-metric proposed in Ferrari and Canny [67], do not cope well with real scenarios. This metric is very widely used because is implemented in simulators such as GraspIt! [68] and OpenRave [69]. However, different studies have shown that: the grasps synthesized using this metrics are commonly fragile [70], under-perform significantly when compared to grasps planned by humans and transferred to a robot [71], or, under object pose error, perform specially poorly when grasping large objects [72]. Although grasp closure is often wrongly equated with stability, actually, these results are not unexpected, since force closure states the existence of equilibrium, which is a necessary but not sufficient condition. Therefore, these results suggest that there is a large gap between reality and the computational models for grasping that are currently available.

This gap is one of the reasons for the attention knowledge-based grasp synthesis has received during the last decade. In these schemes, a grasp is usually defined by [73, 74]:

- a point on the object with which the tool centre point is aligned,
- the wrist orientation of the robotic hand at this point,
- an initial finger configuration,
- and sometimes also an approach vector to perform the last movement of the hand.

Empirical approaches differ in how the set of grasp candidates is sampled and how the grasp quality is estimated. Besides, they commonly stress the efforts on the object representation and the perceptual processing, e.g., feature extraction, similarity metrics, object recognition or classification and pose estimation. This information is then used to sample and evaluate the grasps using some previous knowledge. Since the parameterization of the grasp is less specific, they accommodate better for uncertainties in perception and execution. However, these methods cannot provide guarantees regarding properties such us dexterity, stability or dynamic behaviour [64], since they can only be verified empirically.

3.1.2 Motion Planning for Hand-Arm Systems

Once a grasping position has been computed, it is necessary to determine how the robotic hand-arm system can perform the given grasp. This problem can be formulated as a well-known motion planning problem, but in a very large dimensional space. Planning collision-free motions for robots with a high number of DOFs is known to be a P-space hard problem [75]. Thus, some new approaches are still necessary in

order to find solutions in a faster way so that they can be implemented and used in practice. Besides, it would be desirable, yet not necessary, to perform the movements in a natural manner.

A classical approach to reduce this complexity is to decouple the grasping problem by planning separately for the end-effector and the manipulator. Therefore, it is common to use offline calculated grasping poses for which the inverse kinematics solutions are searched during the planning process [76, 77, 78, 79]. When looking for these solutions, if complete algorithms are used, it is possible to suffer from low performance, mainly caused by the complex task of computing the part of the configuration space (C-space) whose configurations do not lead to collisions in the workspace. Probabilistic algorithms may be used to avoid the time-consuming computation of the boundary of the collision free space. RRT-based approaches are widely used in the context of planning reaching and grasping motions for humanoid robots. The general theory for planning collision-free motions with RRT methods can be found in [80, 81].

This approach can result in computational bottlenecks, unreliable grasps and suboptimal manipulator motions in realistic application settings [82, 83]. For example, in cluttered environments, the grasp database needs to be densely populated with many diverse end-effector postures. On the other hand, when this database is very big, many of the pre-planned grasps are infeasible at runtime: either due to collisions between the robot and the environment, or because they are kinematically invalid. Therefore, selecting a good grasp out of the database can be time consuming. In order to minimize this problem, some application-specific heuristics can be formulated [76]. Other common problem is that both, objects and the end-effector, usually have symmetries, which leads to equally good grasps over a large region of the task space, making it more difficult to make a decision.

Another common approach to solve these problems addresses the reduction of the dimension of the search space. This technique is based on the knowledge of the way the hand is controlled by the brain. Actually, a large part of the human cortex is dedicated to grasping and manipulation, and it would seem reasonable to assume that all of this cognitive machinery is dedicated to finely controlling individual joints and generating highly flexible hand postures. However, results in both, robotics and neuroscience research, point to the contrary, suggesting that a majority of the human hand control during common grasping tasks lacks of individuation in finger movements [84, 85]. A typical example of this situation is the last two joints of each finger. In general, they cannot be moved independently, so it is common to fix this correlation either by the mechanical structure of the hand, or by the controlling software.

From the software point of view, several initiatives have been presented in order to reduce the search space. In [61], the use of a low-dimensional subspace of the hand configuration space for finding hand postures for a given task is proposed. The subspace, called *eigengrasp*, is derived from user studies on human grasping and

mapped empirically to robot hand kinematics. The eigengrasps are then used for finding a small number of optimized pre-grasps in the low-dimensional eigengrasp subspace, then, the final grasping positions are calculated and processed to compute the quality of grasps. In a related work [15], the search space reduction is done in a similar manner, by looking for a representative subspace of the hand configuration space, although this subspace is named as Principal Motion Directions (PMDs). This subspace is built by capturing a number of samples of human hand postures using a sensorised glove and then mapping them to the mechanical hand configuration space. These samples are analyzed using principal component analysis (PCA) to find the direction with largest dispersion. Then, by selecting the first n vectors of the new space base and choosing a bounding area around these vectors, a good bounded approximation of the hand workspace is found. Finally, a sampling based algorithm is used to search around the search space to connect the initial and final configuration of the grasping action. One of the problems using this technique is that many valid solutions in the original space might not be included in the final subspace. A proper selection of this subspace is still an open problem. In a posterior work [86], the search is even more reduced by considering virtual motion geometric constraints. These constraints satisfy certain needs in terms of orientation of the hand with respect to the object to be grasped (e.g. the normal of palm of the hand should be pointing towards the object). Then, a probabilistic road map (PRM) is used to plan natural motions for a hand-arm system. In [87], a similar approach also based on PRM is presented. The planner relies on a topological property that characterizes the existence of solutions in a specific manifold of the configuration space. This property leads to reduce the problem by structuring the search-space directly capturing in a probabilistic roadmap the connectivity of sub-dimensional manifolds of the composite configuration space.

In this context, one of the most repeated concepts in the literature is *synergy*. The notion of motor synergies, or high-level control knobs that have distributed action over sets of low-level actuators, arose in the context of motor coordination [88] and has remained a central topic in discussions of motor control theory, but also in neuroscience and mechanical design of artificial hands. One of the problems around the concept of synergy, is that it has evolved to get a different meaning to many people [89], although dimensionality reduction is generally accepted as the key of synergistic control. Therefore, it is not clear the distinction between this concept and the previously presented *eigengrasps* and *PMDs*. In the robotics field, the first relevant work in this line is that of Santello et al. [84], that determined a two-dimensional grasp subspace from a set of hand configurations obtained when several subjects where asked to grasp several objects. Following studies have demonstrated that the set of experimentally observed postures, or muscle activation patterns, spans only a small subspace of the available multi-dimensional space both during the reach-to-grasp and

the grasping phase [90, 91, 92]. Two important conclusions of their research are: synergies are clearly task dependant, and also that it is not clear how to generalize them for different persons. Vinjamuri et al. [93] analysed temporal postural synergies which they defined as profiles of postural synergies varying over time during rapid grasping movements.

The application of the concept of *synergies* in the literature is quite heterogeneous. In [94, 95], the synergies are considered perfectly stiff and the actual joint variables are a linear combination of synergies. In [96] a synergy-based impedance controller has been derived and implemented on the DLR hand. The problem of computing internal force distribution for controlling object movements by acting on synergies has been studied in [97]. Mapping synergies from the human hand to the robot hand has been addressed in [98]. Their work is based on the use of a virtual sphere for mapping the synergies in the task space. This approach has the advantages to be independent from the robotic hand in use. In [99] a dimensionality reduction for manipulation tasks based on the Unsupervised Kernel Regression (UKR) method is applied to the problem of turning a bottle cap. Zarubin et. al [100] introduced the concept of topological synergies, a low dimensional coordinate representation used to address the grasp transfer problem between kinematically different hands and for the motion planning in configuration space leading a final grasp. In [101], a spatio-temporal model of the lower dimensional manifold of human hand motions during object grasping spatial is presented. In general, the dimensionality reduction shown in these works is based on non-linear techniques. A comparison among several non-linear approaches is discussed in [102], showing that varying the reduction method can potentially reveal different manifold structures of the same data.

3.1.3 Grasping Framework

There are several solutions in the literature covering the topics presented before, however, not many of them address them together. Some grasp planners, like those proposed by [74, 76, 103, 104] rely on very different concepts and therefore is hard to compare them in any other feature but time, with solutions often varying from 0.5 to 10 seconds. This variability often comes from the disparity in the assumptions made through out the phases of a complete grasp planning. Examples of this assumptions include: availability of grasp database, guarantee of reachability of a grasp pose, quasi-static scenarios or the intervention of a human operator.

In the next sections, a framework to face the challenge of object grasping is presented. Both topics introduced before, grasp synthesis and high-dimensional motion planning for hand-arm systems are addressed in this framework. On-the-table scenarios are treated assuming some a priori knowledge: the pose of the objects on the table and an approximation of their 3D model. Then, geometric characteristics of the

object to be grasped and the arm are used to select a pose for the wrist of the hand from which the grasping is done, using a simple empirical approach. Finally, motion planning is performed based on FM^2. The proposed solution presents a combination between hand-arm decoupling plus a reduction of the dimensionality of the problem as the system is treated as a robot formation.

3.2 Grasping Approach based on FM^2

The work presented in this chapter defines a continuous grasping approach covering from perception to grasp execution. The main steps followed through this framework can be seen in figure 3.1. As stated in chapter 1, the visual perception process used has not been developed during this thesis, but it is rather based on previous work by [10]. Therefore, next sections cover the different methodologies used in the other parts, which can be summarize in: environment modelling, grasp pose selection and motion planning for grasping.

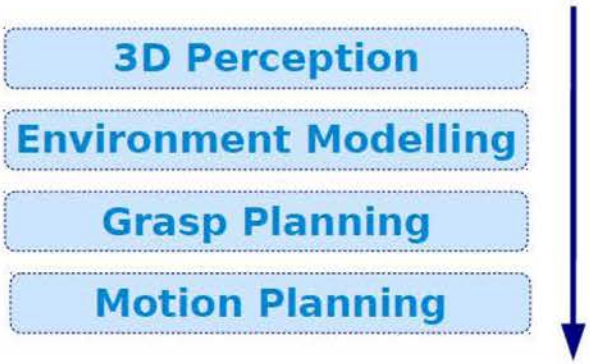


Figure 3.1: General computation pipeline for grasping objects.

3.2.1 Environment Modelling

The necessary model of the environment is highly influenced by the motion planning algorithm explained in section 2.2, since this step is based on FM^2, which is being applied in scenarios modelled as a 3-Dimensional occupancy grid. This grid establishes a binary representation of the world, in which occupied cells are represented by 1 and free space is represented by 0. For the grasping application addressed here, on-the-table scenarios are going to be used for real tests. In this context, the visual perception process provides the following information:

1. The 3D pose transformation from the camera frame to the arm frame represented as a transformation matrix, ${}^{c}T_{a}$.
2. The table pose estimation, which is defined by its height, h_t, and its limits in the XY plane, $X_{max}, X_{min}, Y_{max}, Y_{min}$.
3. The pose estimation of objects on the table, T_i, being i the ieth object.
4. A model of the objects on the table. We distinguish between two different kind of object models, which can be seen in figure 3.2.
    - When the visual perception is able to recognise the object, a 3D CAD model from a data base is provided.
    - When a positive recognition is not achieved, the bounding box (BB) of the object, aligned with the world axis, is used as an approximate representation of the object.

The difference on the information provided by the vision system can be seen in figure 3.2. Obviously, the complete 3D models are a more accurate representation of the information in the scene.

The information provided by the vision system is sufficient to create a 3D occupancy grid, W_o. For the generation of the grid, two main variables must be defined: the size in each dimension and the resolution of the grid. The resolution is user defined. Obviously, the higher resolution (voxel size is smaller) the more precision is achieved on grasp pose selection and motion planning. However, the more number of voxels usually leads to more computation time in motion planning, so a good compromise is needed.

The size is mainly defined by the workspace of the hand-arm system, which is attached to a mobile robot platform, as shown in figure 3.3. Although the workspace of the manipulator with rotational articulation can theoretically be a complete sphere, this is never true because the joints have limits in the rotation they can achieve. Anyway, if a sphere, centred in the attachment between the arm and the base, and with radius equal to the arm length, is considered, only a frontal-lower eighth (from the point of view of the base of the robot) of this sphere is selected to be the actual workspace. In the left picture of figure 3.3 the theoretical sphere we are considering is shown. The reduction of this workspace is based on the following reasons:

- The upper half is eliminated because it is very unlikely to need to perform grasping actions at this height in table top scenes.
- The posterior part of the sphere is also rejected since the sensor system is positioned so that there is more information on the frontal side. The central picture of figure 3.3 highlights (in green) the frontal part of the workspace.

Figure 3.2: At the top, a scene with several objects lying on a surface. Below, the cloud of the non-recognised objects extracted from the scene after Euclidean segmentation is applied. In the third row, besides the cloud of the object, several views of the corresponding 3D model of a recognised object can be seen.

- Finally, from the frontal part, the half situated in the opposite side of the attachment of the arm is not considered since it is very likely to provoke a collision between the arm and the mobile base. The right picture of figure 3.3 shows the lower part of the workspace in the side of the robot where collisions are less likely to happen.

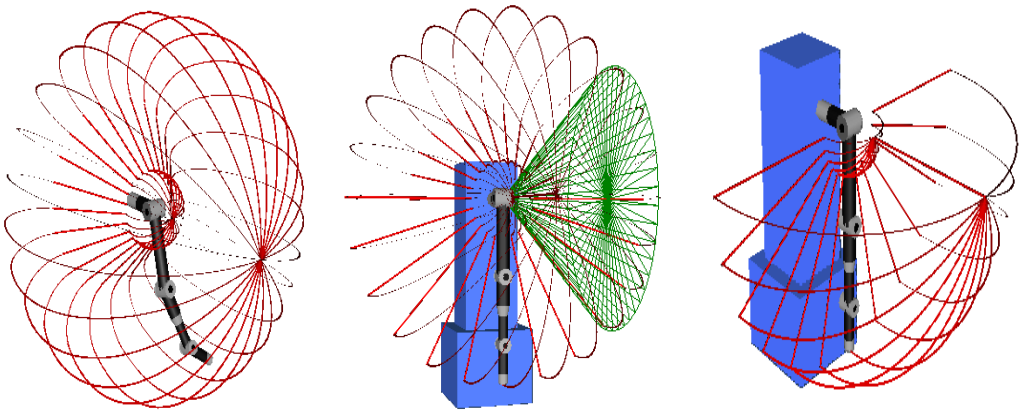


Figure 3.3: From left to right: the total workspace of the arm forms a sphere around the initial point. In the middle, the arm is attached to the robot base with which it collides. The green cone shows frontal area of the workspace. On the right, the fourth of the workspace which does not collide with the mobile base, only the frontal part of it is actually considered for manipulation.

Then, in figure 3.4, a posterior view of the considered workspace and its 3D grid representation can be seen. Besides, the arm start reference frame (in the up-right corner) and the hand starting position (blue palm and green phalanges) are also shown. Finally, a box-like grid representing the used workspace is created. This occupancy grid includes voxels which are not reachable for the arm. In order to avoid the planner to consider voxels out of the real arm workspace, those cells are considered as occupied. In figure 3.4. Reachable cells are plotted in grey, white areas are nor reachable by the arm. In this environment, the table plane is added by assuming as occupied all the voxels under the provided height and between its limits in the XY plane, which can be also appreciated in figure 3.4, just on top of it, there is the model of a jug.

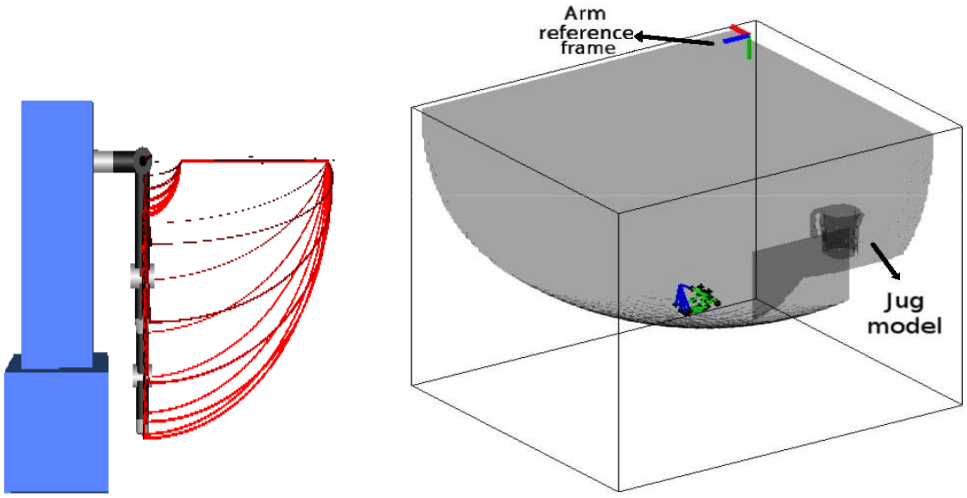


Figure 3.4: On the left, the rear view of the considered workspace of the arm. On the right, the workspace in modelled in an occupation matrix, the arm coordinate frame is in the top right corner of the image and the start position of the hand can also be seen. The workspace of the arm can be seen in grey, it can be appreciated that the space under the object (a jug) is also marked as occupied as that is the table plane.

Object Voxelization

Being the chosen spatial representation an occupancy grid, information of the objects on the table retrieved by the camera has to be adapted to such a representation to be included in the environment. Depending on whether the visual system is able to recognise the objects or not, two different situations are faced: bounding boxes or 3D meshes.

In the case of receiving a bounding box as a representation of the object, since its axis are aligned with the world axis, it is as simple as dividing its volume in voxels of the same resolution as the environment and then include it in the 3D grid.

In the case of using a 3D model, a voxelization process is also needed. The proposed approach uses a ray-tracing intersection method which is similar to the one described in [105]. There is an important requirement with the mesh: it has to be watertight. This implies that it cannot be used with typical incomplete object models retrieved by 2.5D sensors with only one view of an object, but instead a full 3D model is needed. The output of the algorithm is a binary matrix in 3 dimensions whose voxels are cubic in shape and aligned with the Cartesian coordinate system.

The different steps of the process used for the voxelization of the 3D mesh can be seen in figure 3.5, represented in 2D for clearness. Essentially, it converts a solid

model represented by a set of contiguous triangular facets modelling the object's surface. The algorithm uses OBJ 3D model format as an input, although any other format which includes vertices and facets information could be used instead. It starts by reading the coordinates for each facet, resulting in a $N \times 3 \times 3$ array, where N is the number of facets of the model, each of which has 3 vertices whose positions has 3 dimensions.

With the coordinates of each facet, the bounding box of the model is determined by looking at the maximum and minimum value of the vertices of each facet for every axis. Then, voxel size is computed by dividing the maximum dimension of the bounding box by the desired resolution. Therefore, the total number of voxels needed to represent a 3D model is given by:

$$\begin{aligned} size\_X &= max\_X - min\_X \\ X\_axis &= \lceil (\frac{size\_X}{resolution}) \rceil \\ total\_voxels &= X\_axis * Y\_axis * Z\_axis \end{aligned} \tag{3.1}$$

where $size\_X$ is the size of the X axis, X_axis is the number of voxels in the X axis, and $total\_voxels$ is the number of total voxels which will be checked. Note that only the expressions for the X axis have been included, buy they are the same for all of them. The resolution term is introduced by the user, however, it should be the same resolution as the one used to create the occupancy grid of the environment.

The voxelization process consists on ray-tracing in every axis direction and finding intersections with the facets. The user can specify if the ray-tracing is done along all the axes or only in some of them. Besides, since watertight models are being used, the number of intersections for any ray with the object has to be even if the 3D model is free of errors and the ray completely traverses the 3D model.

The ray-tracing starts at point given by:

$$\begin{aligned} X\_start &= X\_min + \frac{voxel\_size}{2} \\ Y\_start &= Y\_min + \frac{voxel\_size}{2} \\ Z\_start &= Z\_min + \frac{voxel\_size}{2} \end{aligned} \tag{3.2}$$

which corresponds to the centre of the voxel with smallest Cartesian coordinates. Subsequent points are computed by moving in the X direction at a distance equal to the voxel size until the final voxel in this direction is found, these points are the red dots in figure 3.5. Then, the loop iterates over the other two coordinates. For

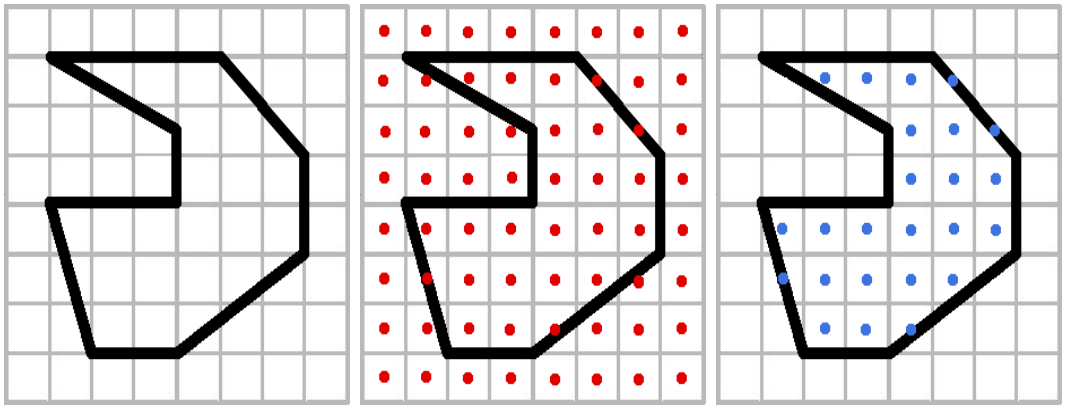


Figure 3.5: Steps of the voxelization of a 3D mesh. From left to right: first, a grid that covers the object size and using the desired resolution is made; then, the ray-tracing starts creating samples of the grid (red dots). Finally, the samples which are considered to be part of the object (blue dots), those pixels (voxels in 3D) are considered as occupied in the world representation.

each new the point of the ray-tracing, it is checked whether the ray crosses a facet or not. In order to improve computation time, the facets of the mesh which might be passed through by this ray iteration are filtered. Note that the start position, moving direction and voxel size are enough information to filter these facets. In order to check whether the ray crosses the facet or not, a general solution is:

1. For every facet, take one edge at a time.
2. For the edge, find the position of the opposing vertex relative to that edge.
3. Find the position of the ray relative to that edge.
4. Check if the ray is on the same side of the edge as the opposing vertex.

If the last verification holds true for the three edges, then the ray definitely passes through the facet. When doing this, two special cases may occur: the ray crosses exactly on an edge or on a vertex. In both cases, it is counted as crossing the facet. When the final voxel is reached, voxels lying between an odd and the next even found intersection are considered as occupied (the ray could enter into and out of the object several times in one ray-tracing), these points are represented with blue dots in figure 3.5. This process is repeated step by step from the minimum to the maximum coordinates in all axis of the model.

Possible errors in the voxelization can be caused by: (a) missing thin sections, (b) odd number of intersections caused by the presence of sections thinner than voxel

size, and (c) discretisation inherent to voxel models. Some of these errors can be identified and treated. For example, every voxel which is considered as occupied when ray-tracing along one axis, should be assign as occupied when ray-tracing along the other axes. If this is not accomplish, the voxel is identified as inconsistent and may be further analyzed with a lower resolution. Anyway, the easiest manner to overcome the discretisation errors is by increasing the voxel resolution. Note that the maximum possible error will be half of the voxel size.

Figure 3.6 shows the voxelization of the same mesh with a resolution of 1.25 mm when the ray-tracing is done in only one dimension. Although the difference in the resulting occupancy grid is not easily perceived (specially in a 2D image), the number of occupied voxels in the grid decreases by an average of 5.64%, the computation time decreases to only 0.951 s. Besides, figure 3.7 shows the resulting voxelization of a 3D mesh at different resolutions. From left to right and top to bottom: 10 mm, 5 mm, 2.5 mm, 1.25 mm and 0.625 mm of resolution. The last image corresponds with the original mesh, it has 208353 vertices and 69451 facets. The size in each axis is: 15.12, 9.2 and 15.5 mm respectively. In all cases, the ray-tracing was done in the three dimensions. Table 3.1 shows the average computation time for the voxelization of the same mesh at different resolutions over 20 trials.

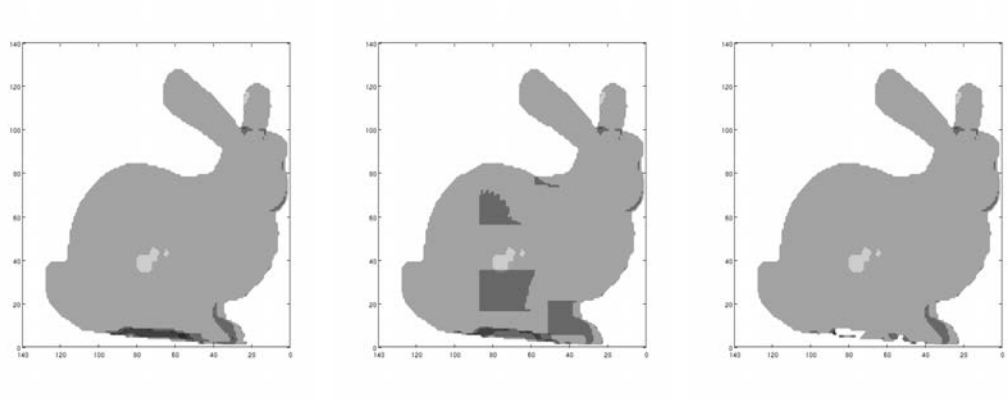


Figure 3.6: A frontal view of the *bunny* 3D mesh voxelized at a resolution of 1.25mm. In this case, the voxelization has only been done in on dimension, X, Y and Z respectively.

Table 3.1: Computation time for the voxelization of the same mesh at different resolutions over 20 trials.

Resolution (mm)	10	5	2.5	1.25	0.625	0.3125
Time (s)	0.092	0.284	0.752	2.102	5.84	14.54

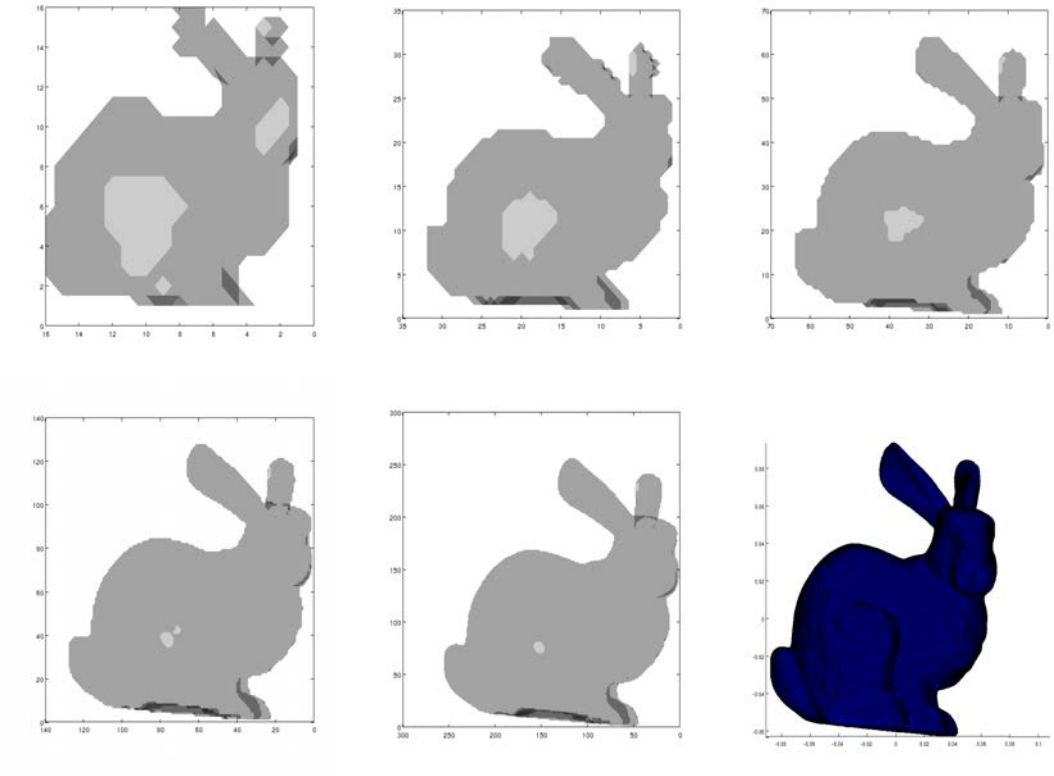


Figure 3.7: A frontal view of 3D mesh of a *bunny* voxelized at different resolutions.

3.2.2 Grasp Pose Selection

Although grasp synthesis itself is not a key point of research of the thesis, it is obvious that, along the grasping process, it is needed to compute a pose for the hand from which the object can be grasped. Therefore, a simple empirical algorithm has been developed for grasp selection, which is inspired by human general habits when grasping objects. In this case, computing the final positions of fingertips on the object is not a goal, but instead, the focus is on where to position the hand with respect to the object in order to perform the grasp.

An important aspect when generating samples of possible grasp poses is how to relate the hand with respect the object. In this work, this relation is given by the grasp centre frame (GCF). GCF is a virtual frame, f_G, which represents a reference for the grasps to be planned. For simplicity, this frame has the same orientation as the origin of the hand, f_H, and therefore it is defined as a translation from it. Figure 3.8 shows the reference frame of this point over the workspace of the fingers of Gifu Hand III. The location of the GCF has been set at the centre of mass of the

workspaces of all the fingers.

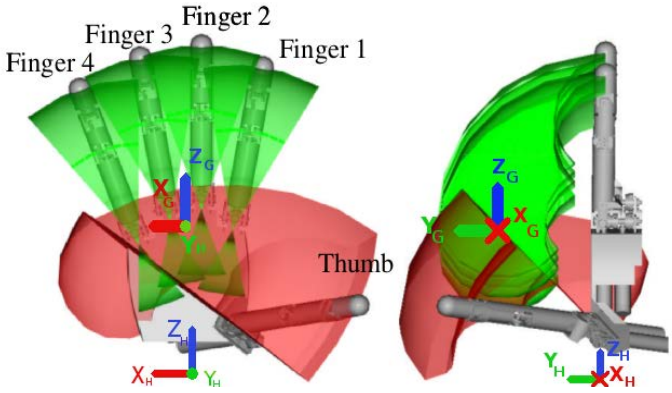


Figure 3.8: Gifu Hand III with the workspaces of its fingers (in green) and the workspace of the thumb (in red). Besides, the reference frame of the origin of the hand (H) and the reference frame of the GCF (G) are shown.

Then, if we take a look at the literature, neuroscientists have found to simple ideas commonly followed by humans when grasping objects:

- Human grasp locations are influenced by centre of mass (COM) location on the object and tend to draw the grasp position towards COM location [106].
- Humans tend to grasp objects by aligning their hand with the principal components of the target object [71].

Obviously, these two rules represent only a small quota among the information humans use to choose a grasp, however, these two simple ideas are a sufficient basis to build a grasp synthesis algorithm.

Therefore, objects to be grasped need to be studied to obtain the geometric properties just mentioned. It is important to note that when applying these rules to the objects, no difference is made in the treatment of the object's modelled by a bounding box or by a 3D mesh. Therefore, the variation in modelling only leads to a difference in the accuracy of the object's properties.

- **Centre of Mass (COM)**. In both cases, BB or 3D mesh, it is computed as the geometrical centre of the object, assuming that its mass is uniform along its volume. In the first case, it is determined by the half point in each axis. In the second, it is calculated as the arithmetic mean, following:

$$axis_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.3}$$

being x each of the three Cartesian coordinates axis and n the number of points in the mesh. Figure 3.9 shows a point cloud and a full 3D model of the same object. The centre of mass of the BB of the point cloud has a displacement of: $X = 2.15, Y = 0.46, Z = 5.8$ millimetres. The higher value in the Z axis is due to the fact that lower points of the point cloud of the object are taken out when the plane is segmented.

- **Principal axis**. In the case of the bounding box, axis are aligned to the world frame and then named after them, also, the principal corresponds to the longest axis. When treating a 3D mesh, it is aimed to compute a local, object-centric coordinate system. Principal Component Analysis (PCA) is used to compute the axes of minimum and maximum variance in the horizontal plane, assigning them to the local x- and y-axis respectively. Besides, since we assume that objects are resting on a surface, the vertical direction (relative to gravity) can directly be assigned to the local z-axis. Then the biggest one is taken as the principal one. Note that z-axis is then always parallel to gravity. Objects in figure 3.9 have an difference in the angular displacement of 2.6 degrees around the Y axis.

- Besides, the ratio between the two main axes is measured. Objects are considered to be *long* when this ratio is larger than 1.5. This property affects on how the samples are created.

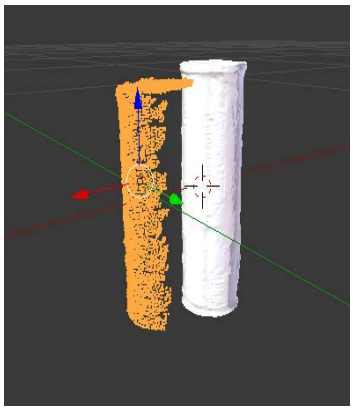


Figure 3.9: A full 3D model of an object, and the point cloud captured by a depth sensor in a frontal view. In this case, the centre of mass of the point cloud has a displacement of: $X = 2.15, Y = 0.46, Z = 1.8$ millimetres with the one of the 3D model. Besides, the main axis of the point cloud has an angular displacement of 2.6 degrees around the Y axis.

The developed approach is mainly based on [68], and the generation of possible grasps is based on the imitation oh human grasping attending to the geometrical properties of the object to be grasped. There are two types of possible grasps samples: top grasps, in which the palm of the hand faces the table plane, and side grasps, In which the palm surface is orthogonal to the table surface. Furthermore, two different type of grasps are considered: fingertip and power grasps. The former is commonly named as precision grasps, and only implies contacts on the fingertips. However, precision is not used here because no computation on where these contacts occur is made. The latter, intends to use the whole hand surface to make contact on the object. In practical terms, the difference between them is just the distance, from the object to the palm surface, from which the fingers are closed. Attending to all these characteristics, the grasp candidates are obtained by:

- The hand is placed at a distance so that the GCF matches the COM of the object.
- The local Y-axis of the GCF of the hand (normal to the palm surface) is placed in the perpendicular direction with respect to the principal axis of the object. Then, if this direction is parallel to world Z-axis (gravity), it is considered a top grasp, otherwise a it is a side grasp.
  - **Side Grasp:** The hand's X-axis is placed pointing against the tables' plane normal in order to keep the thumb pointing up, as humans do. Multiple grasps are then generated by rotating around the world's Z-axis creating a circle. The number of generated grasps depends on the divisions made along this circle. An example the side grasps sampling is shown in figure 3.10, in which samples are generated every 90º.

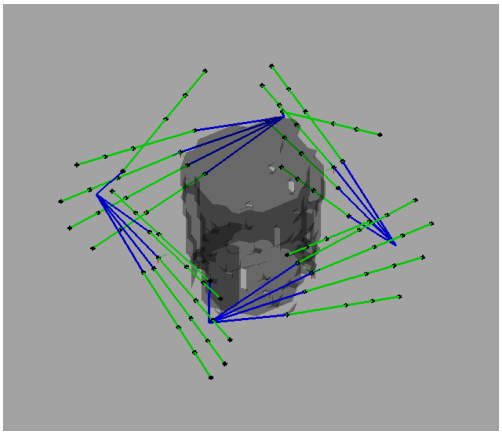


Figure 3.10: The hand positioned at different samples of possible a side grasps.

– **Top Grasp:** When the object is considered to be long, grasps are sampled as *side grasps*, considering both possible directions of the hand's X-axis. Otherwise, grasps are generated by rotating around the world's Z-axis. An example the top grasps sampling, for an object which is not considered long, is shown in figure 3.11.

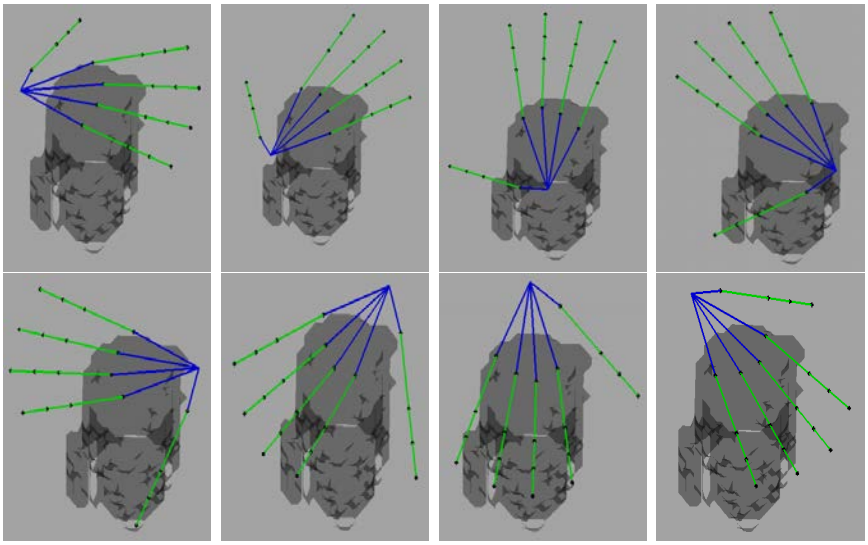


Figure 3.11: Examples of top grasped sampled by rotating around the world's Z-axis, there is one sample every 45º.

- Finally, both fingertip and power grasps are considered. When power grasps are used, the starting position is the one of the fingertip grasp, then the hand approaches the object moving in a straight line towards the COM of the object until a collision between the palm and the object is detected. Then, the grasp can be executed.

At this point, several grasp possibilities have been computed. In order to choose only one of them, we check whether they are physically feasible or not. This is done by looking at arm capabilities and environment restrictions. The first sample which fulfils these conditions, will be chosen to be executed. Two different conditions have to be met:

- Collision checking. The voxels which correspond to the volume of the hand are set as *occupied* and checked for a collision with the environment. For this purpose, the object to be grasped is considered part of the environment, meaning that the hand located in a grasp location is not allowed to collide with the

object. This is performed for every grasp candidate location, discarding the ones in which a collision is detected. Two examples of the application of the collision checking are shown in figure 3.12. In the left one, the hand is positioned in a side grasp and the fingertip of the little finger is in collision with the table surface, so this candidate is discarded. The hand model on the right picture is located for performing a top grasp and is not colliding with any object, so it is a valid sample.

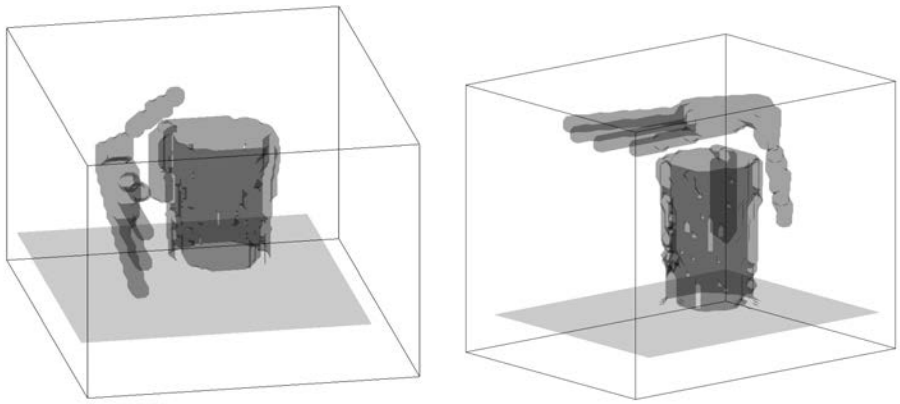


Figure 3.12: Collision checking examples in a top and side grasp pose candidates.

- Reachability. After collision checking is performed, several possible grasp poses may still be available. The last step before performing the grasp consists on checking whether the grasp poses are reachable by the arm. Therefore, the inverse kinematics (IK) problem has to be solved, which consists on finding the values of the joints of the arm which lead to the desired Cartesian pose. The first sample for which IK is solved, is the one to be executed. Since there is not an analytical solution for the IK problem of the arm attached to ManfredV2, an adaptative evolutionary strategy, detailed in [11] and therefore previously available in the laboratory, is used. The optimization is done using the Differential Evolution algorithm [107]. It mainly consists on a local-search optimization of pre-taught joint positions for finding the desired IK solution. Therefore, this method requires an offline creation of a database which contains joint positions of the arm, which lead to Cartesian poses from which on-the-table grasping is possible to be performed. The bigger the database and the more different the arm positions saved, the more likelihood to find the IK solution for a given reachable pose.

3.2.3 Motion Planning and Control based on FM^2

Once the final grasp pose has been selected, a motion plan for the hand-arm system has to be computed. In this case, the robotic system to be used is the robot ManfredV2, which is described in appendix B. Its hand-arm system has 22 degrees of freedom. As mentioned in section 3.1.2, such a complex system requires a special approach to find feasible solutions in a reasonable amount of time. The approach explained in this section is based on a dimensionality reduction of the problem. The solution is built on the next idea: treating the hand-arm system as a leader-follower robot formation. As explained in chapter 2, this idea allows as to convert a complex multi-robot path-planning problem into a much simpler coordination control. In the case of the hand-arm system, the grasp centre point is considered as the leader of the formation, for which a 3D Cartesian path is computed with FM^2, and the followers are the fingertips of the hand. While covering the path, both the orientation and the hand opening/closing are controlled to accomplish the grasp. Therefore, throughout the process, the hand acts as a functional unit [85]. Besides, since the path planning is only done in 3D, it is a much easier process and computation time is saved.

Geometry of the hand as a robot formation

In order to treat the hand as a robot formation and perform the necessary computations, the Gifu Hand III kinematic chain characteristics are modelled as shown in figure 3.13. This is the hand attached to the arm in ManfredV2, as detailed in appendix B. In this structure, the lengths of the links between different joints (palm, finger separation and proximal/inter/distal phalanges) are constant, while the angles between these links (α, β and γ) are variable. These angles are limited by software following the specifications of the Gifu Hand III [108]. It is important to notice that, as in a human hand, the distal joint of each finger engages linearly with the proximal joint [109]. The same idea is applied for the thumb, although in the real robotic hand these joints are independent. In order to consider the hand as a robot formation, mobile robots are virtually located at the position of the fingertips of the hand (followers) and the grasp centre point (leader). The hand is then considered as a leader-followers formation in which the followers change their location while performing the grasping process depending on their position in the environment. The main difference with the robot formation described in chapter 2 is that in this case, the potential position of the follower robots (fingertips) is highly limited by the mechanical design on the hand.

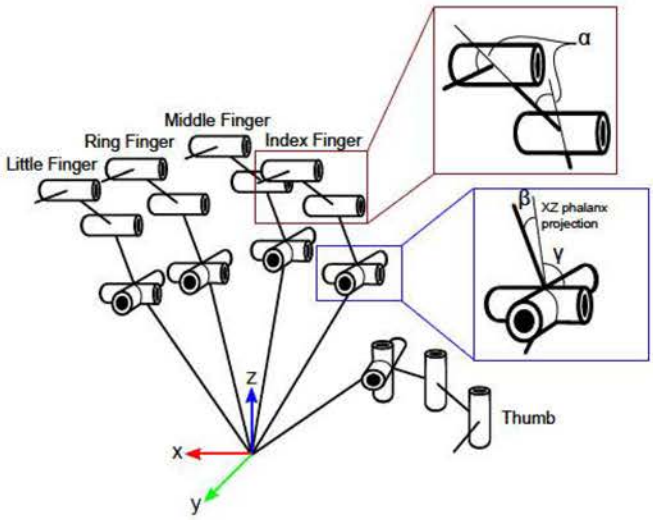


Figure 3.13: Kinematic model of the Gifu Hand III.

Path planning for the hand as a robot formation

In order to perform a given grasp, the grasping action can be divided into two phases: reach (often called pre-grasp) and grasp [110, 111]. In the reach phase the hand moves towards the object and stops at the grasping point from which the grasp is performed. For this movement to be done, a path from the start position towards the object is calculated using FM^2 path planning algorithm. While covering the path, the configuration of the hand evolves imitating human movements. The main objective of this evolution is to open the hand to ensure that the grasping can be accomplished, later a flexion of the hand (finger closure) may occur in order to anticipate the object contact [84, 112, 113]. Although it is clear that the preparation of the shape of the hand begins well in advance in the reach phase [85], this process occurs differently depending on the experiment setup [114]. For example, in [85], subjects were asked to grasp objects several times. In each trial, the time used for visual recognition of the scene was different. The experiment showed that subjects with more time for scene recognition, perform a more advanced preshape preparation. In the grasp phase, the grasp has to be executed. If exact grasp positions on the object were available, a path from each fingertip to these positions could be easily computed using FM^2. These paths would then be covered by moving the fingers of the hand while the palm remains in the same pose. If no grasp positions are available, then the fingers must converge towards the object's surface in order to finish the grasp. Next sections explain how to handle both phases considering the hand as a robot formation.

Reach phase: approaching the object to be grasped

As stated before, the first phase of the algorithm consists of an approaching movement towards the object to be grasped. In order to perform this approximation, the concept of robot formation planning based on FM^2 [17, 40] is used to select the different configurations adopted by the structure of the robotic hand.

At this point, the voxel-based occupancy map of the environment has already been built, and the object is included. Besides, a grasp pose has been selected. The hand is placed at the starting pose with a certain configuration, and then the robots of the formation are located at their positions using forward kinematics. Then, a path is computed from the leader's start point to the grasping point. As in chapter 2, the resulting path is a 3D vector of goal positions which has the main characteristics of FM^2: safeness and smoothness. This path is then covered in an iterative loop. For each iteration, the value of the joints of the kinematic model of the hand is updated, therefore the position of the followers in the formation changes. This approach allows us to control the 3D pose of the fifteen DOFs of the robotic hand calculating only one 3D path.

Figure 3.14, left, shows an example of a simple scenario in which the object to be grasped is a cube that is floating in space, as well as the hand, and they are both located in free space. The path towards the object is shown in blue. On the right side, a 3D view of the velocities map of the scenario is shown. The different colours indicate the distance to the closest obstacle, being the dark blue the closest one while dark red areas are the farthest ones. Note the blue circle created around the area where the cube is located. Then, at the bottom, a 2D projection of the path on the centre slice of the map of velocities, parallel to the XZ plane, is shown. Let's keep in mind that the velocities map can also be interpreted as a distance field, so it can be used to know when the hand is approaching to an obstacle or to free space. In order to have a good preshape of the hand when the grasping point is reached, the evolution of the hand geometry is divided into two different phases. These two phases, are highlighted in different colours in the bottom picture of figure 3.14. Blue and green areas around the path are the two phases of the reaching evolution. The blue area corresponds to the part of the path where the leader is moving towards free space, which is characterized by a positive gradient of the velocities map. Since this indicates that the hand is getting far from obstacles, the purpose in this phase is to evolve to an open configuration of the hand which ensures that the grasping points can be reached afterwards, which occurs following equations (3.4) and (3.5).

$$\alpha_{i,j} = max(\alpha_{i-1,j}, distance\_leader \times \alpha_{j,max}) \tag{3.4}$$

$$\beta_{i,j} = max(\beta_{i-1,j}, distance\_leader \times \beta_{j,max}) \tag{3.5}$$

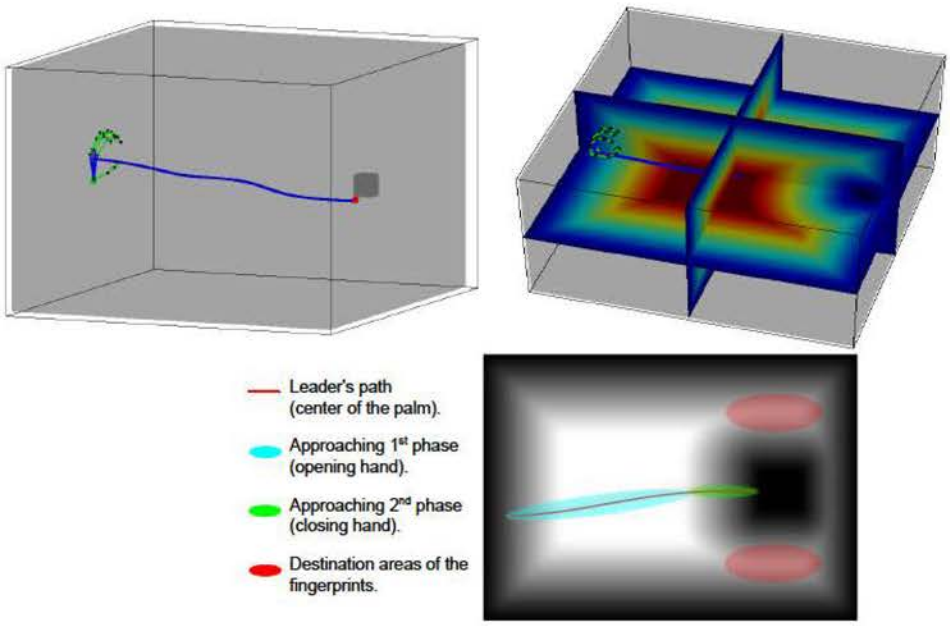


Figure 3.14: Simple scenario in which the object to be grasped is a cube floating in space, located on the right. The hand, on the left, is not attached to any arm, and they are both located in free space.

where j indicates the finger, i corresponds to the iteration and *distance_level* is the value of the map of velocities in the position where the leader is located in the environment. Note that maximum opening of the hand is obtained for values: $\alpha_{j,max} = 0°$, $\beta_{j,max} = 90°$ and $\gamma_{j,max} = \pm 10°$, all of them are expressed in degrees. Figure 3.15 shows the evolution of these angles for the simple scenario presented before. The hand's palm is drawn in blue, while the phalanges of the fingers are plotted in green. The small black dots are the joints of the hand and the green cross in front of the palm indicates the positions of the virtual leader of the formation, note that this time it differs from the GCF. The path of the leader is divided into two areas. The red one corresponds to the part that the leader has already covered, while the blue one is the part that it has not been covered yet. While the hand approaches the object, the dark grey cube on the right, the hand evolves to an open configuration.

It is also important to note the green area in the bottom part of figure 3.14, which corresponds to the moment in which hand is getting close to the object, which makes the value of the map of velocities become smaller. Therefore, the sign of the gradient for consecutive positions of the leader becomes negative, which allows for an easy

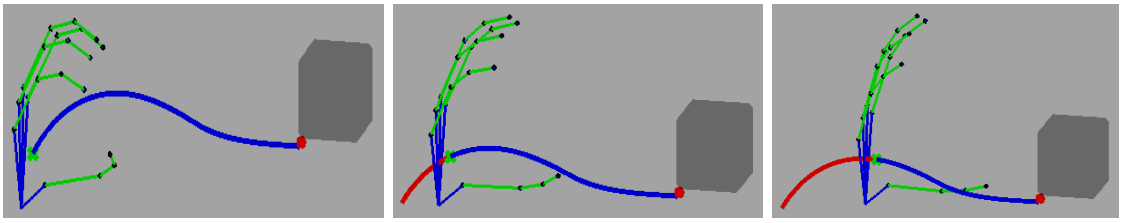


Figure 3.15: Evolution of the hand towards an open configuration.

detection of the change of the phase. In this stage we try to close the fingers so that the position of each fingertip gets as close as possible to its local maximum in the map of velocities, which are indicated by the top and bottom red ellipses in the right picture of figure 3.14. This way, we ensure that the position of every fingertip is as safe as possible, and at the same time make the grasping phase shorter since part of the movement is already done. To detect when the local maximum has been reached, the gradient value of the map of velocities for the fingertips is checked. Figure 3.16 shows the evolution of the geometry in this stage. The update of the angles of the joints follows equations 3.6 and 3.7.

$$\alpha_{i,j} = \alpha_{i-1,j} + K_1 \times grad\_distance\_level_{i,j} \tag{3.6}$$

$$\beta_{i,j} = \beta_{i-1,j} + K_2 \times grad\_distance\_level_{i,j} \tag{3.7}$$

where K_1 and K_2 are constants that define the speed of the closing movement, and $grad\_distance\_level$ is the difference in the distance value between two consecutive positions. K_1 and K_2 have to be set experimentally, specially because this movement depends on the change in the distance values, which depends on the resolution chosen.

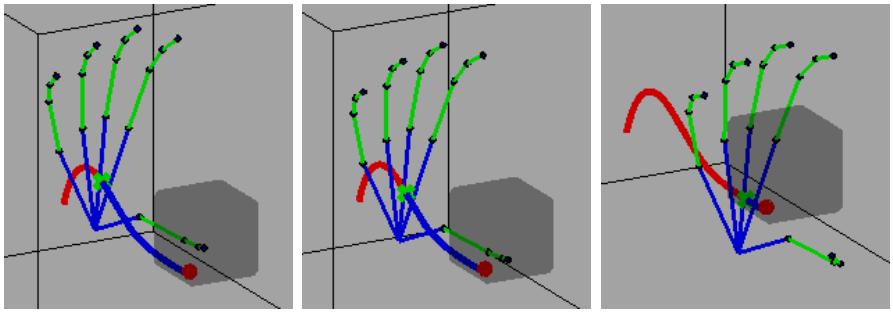


Figure 3.16: Second phase of the approaching step, while getting close to the object, the hand starts closing.

However, the proposed scenario in figure 3.14 is not realistic since objects are floating. Besides, the hand orientation does not change through the path, which also makes it easier to control. Normally, the start pose of the hand is assumed to be as shown in figure 3.17, which is the one in which the arm is in a relaxed configuration with no power in the motors. Note in the figure the configuration adopted by the fingers, the hand is completely closed, otherwise it would collide against the base. This makes the opening phase to be even more critical.

Figure 3.17: Starting configuration of the hand to avoid a collision with the base.

Furthermore, it is obvious that the orientation of the hand needs to change in order to be able to perform the grasp. The change in orientation is also controlled while the hand is moving along the path, although in contrast with robot formations, this cannot be done using the Frenet thriedron, since this reference depends on the path but does not take into account the pose of the object in the environment. In this case, the orientation is controlled using the spherical linear interpolation, commonly known as *slerp* algorithm [115]. This algorithm computes a constant-speed motion along a unit-radius great circle arc, given the two ends of the arc and an interpolation parameter between 0 and 1. The geometrical equation for *slerp* is as follows:

$$Slerp(q_0; q_1; t) = \frac{sin[(1-t)\Omega]}{sin(\Omega)} \times q_0 + \frac{sin[t \times \Omega]}{sin(\Omega)} \times q_1 \tag{3.8}$$

where q_0 and q_1 are the start and end rotation, and t is the interpolation parameter. The geometrical interpretation can be seen in figure 3.18.

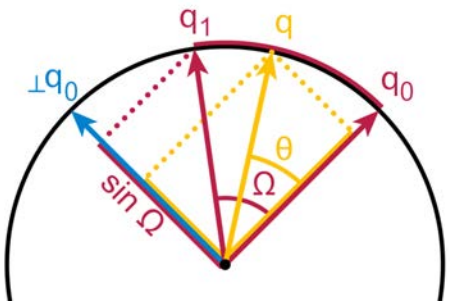


Figure 3.18: Geometrical representation of the *slerp* formulation.

When *slerp* is applied to unit quaternions, the resulting path is a 3D rotation with uniform angular velocity around a fixed rotation axis. *Slerp* is specially interesting because it gives a straightest and shortest path between its quaternion start and end points, and maps to a rotation through an angle of 2Ω, being Ω the angle between q_0 and q_1. However, because the mathematical definition of a quaternion shows that q and $-q$ map to the same rotation, the given rotation path may turn either the "short way" (less than 180°) or the "long way" (more than 180°). In order to prevent computing long paths, one of the orientations can be negated when the dot product, $cos(\Omega)$, is negative, thus ensuring that $-90° \leq \Omega \leq 90°$.

Figure 3.19 shows the environment modelled with the techniques previously indicated. Arm and hand start poses are indicated. The workspace of the arm is presented as a grey eighth of sphere. The table is also modelled and on top of it there is the model of a jug. On the right, the distances/velocities map of this scene is shown sliced around the object position. The voxels which are occupied, or very close to an obstacle, are shown in red. They correspond to the areas out of the workspace of the arm (or close to its virtual wall) and the voxels around the object. Then, while voxels are located further from any obstacle, their colour changes to yellow, green, blue and purple progressively.

Using the environment shown in figure 3.19, an example of a side and a top grasps are computed. In the case of the side one, the resulting path and evolution of the movement of the hand are shown in figure 3.20. The path can be seen as a red line. Although the virtual leader is located at the GCF, the resulting path is shown referenced to the wrist frame, since otherwise the visualization is much worse because the information would be overlayed. While covering the path, the hand evolves towards the grasp configuration by opening and the fingers and changing its orientation.

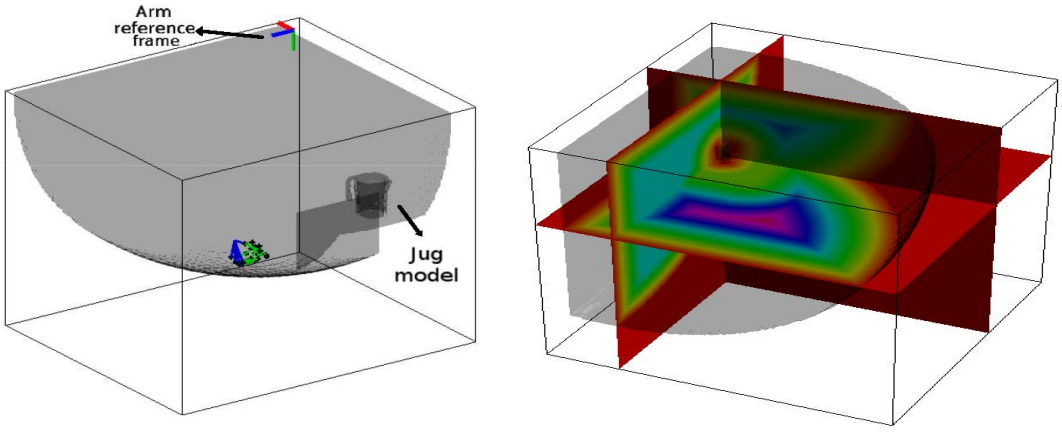


Figure 3.19: On the left, an object in an on-the-table scene modelled with the techniques explained through this chapter. On the right, the velocities map of the scene sliced around the object position drawn over the scene.

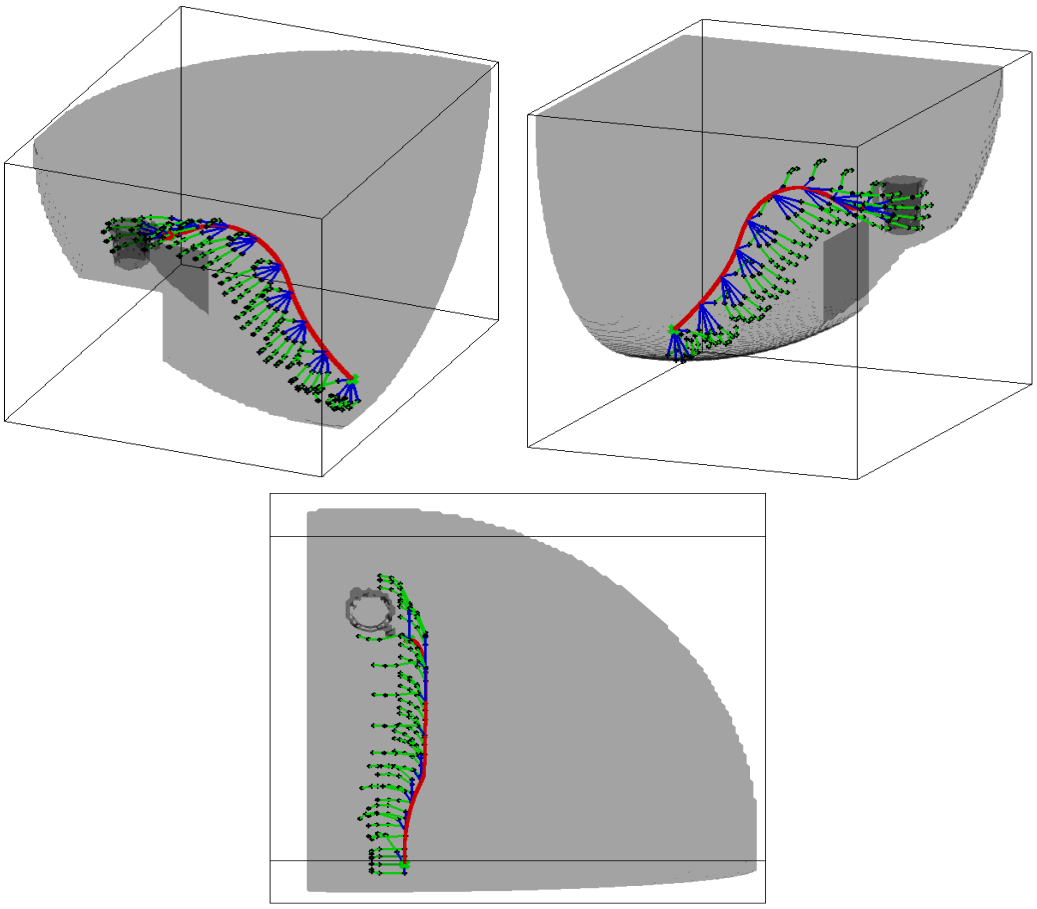


Figure 3.20: Different views of the evolution of the hand while performing the reaching phase of a *side* grasp. The hand starts completely closed and opens while approaching the object. The orientation of the hand changes linearly from initial to final pose.

Then, figure 3.21 shows the reaching phase behaviour over the path computed for the top grasp example. It can be appreciated that both the opening and the orientation evolve while the hand is covering the path. In the lower picture of the figure, which shows a view from the top, it is interesting to see how the path first moves towards the centre of the workspace of the arm and later turns towards the object. This is a natural behaviour in FM^2, since the central area of the workspace permits higher velocities.

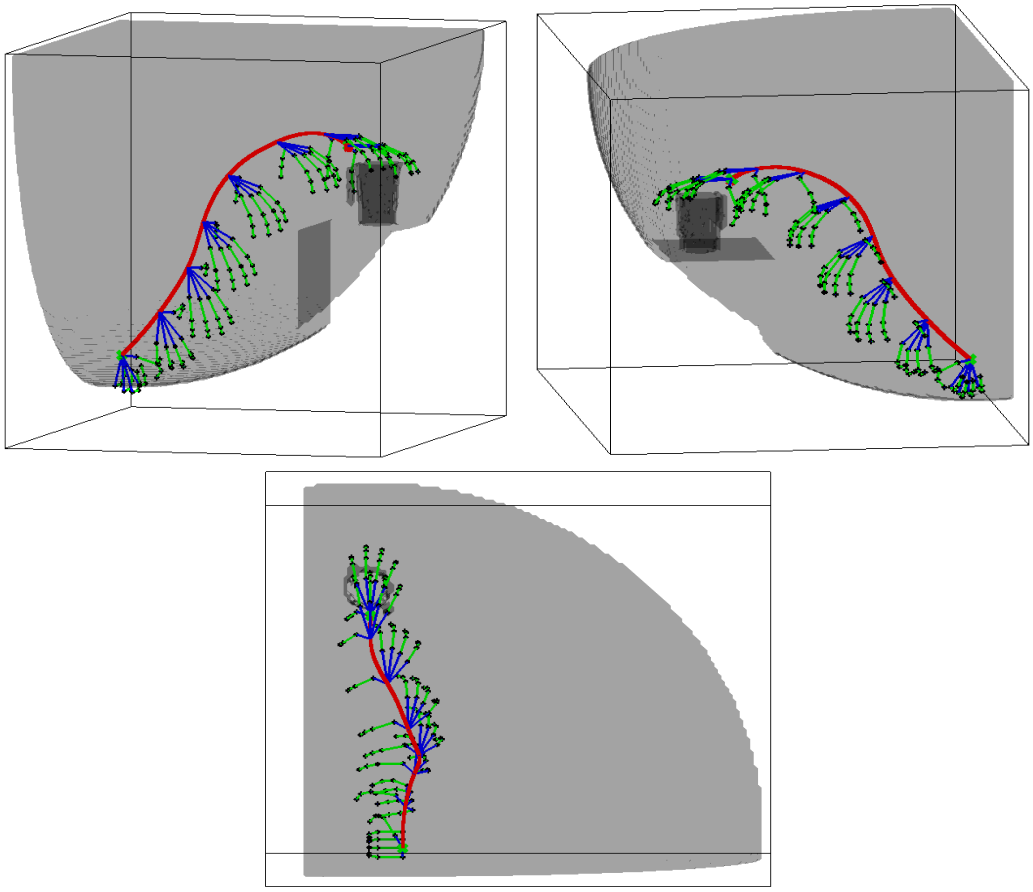


Figure 3.21: Different views of the hand while moving towards a *top* grasp pose. The hand starts completely closed, opens while approaching the object and, at the same time, the orientation of the hand changes linearly along the movement.

Grasp phase: fingers close on the object

The reach phase ends when the leader of the formation reaches the grasping position, which means that the computed grasping pose has been reached. Next, in the grasp

phase, the fingers have to move towards the object and grasp it.

If the grasp synthesis process computes grasping points on the objects, they would need to be reached by the fingertips of the hand. If we go back to the simple example of the floating cube, we could very easily compute some grasping points on the cube just by imitating a human grasp, as shown in figure 3.22 on the left side. Then, paths from each fingertip to their respective goal could be computed by FM^2 and then executed using inverse kinematics of the chain formed by each finger. The centre picture figure 3.22 shows an example the case of the floating cube, while the right side shows the final configuration of the hand when the grasping points are reached.

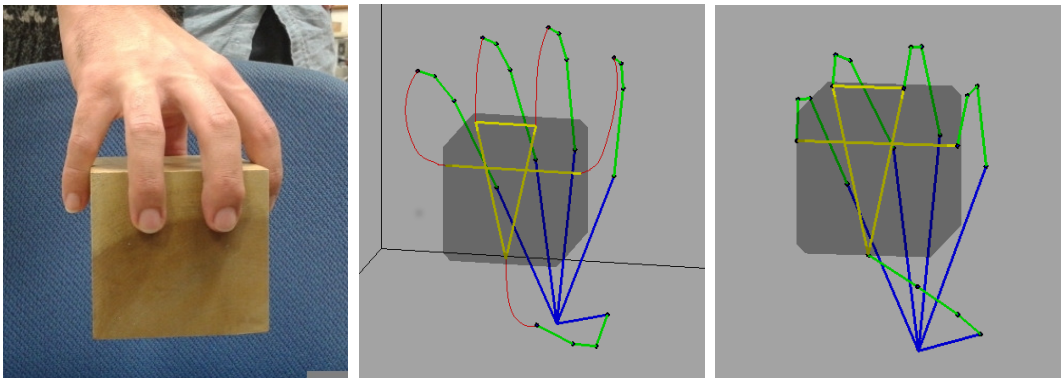


Figure 3.22: Left: a human hand holding a cube. Centre: in red, the paths computed with FM^2 from the initial position of the fingertips to their goal locations on the objects. Right: Final configuration of the hand grasping the object.

However, the grasp synthesis process presented in this chapter does not compute exact grasp locations on the object, but just the position from which the grasping action has to be executed. In order to close the fingers and grasp the object, again the velocities map (first potential of FM^2) is going to be used. Figure 3.23 shows a close look of the object on the table and the slices of X, Y and Z planes of the velocities map around the object. Occupied cells are drawn in dark blue and the colour changes from blue to orange as the voxel is further from any obstacle. Then, in figure 3.24 the same slices are drawn separately, and the gradient vectors are included. Note that the gradient vectors point towards the object's external surface in a perpendicular way in all cases. Besides, since the value of each voxel has also a *distance* meaning, it becomes smaller when approaching an obstacle.

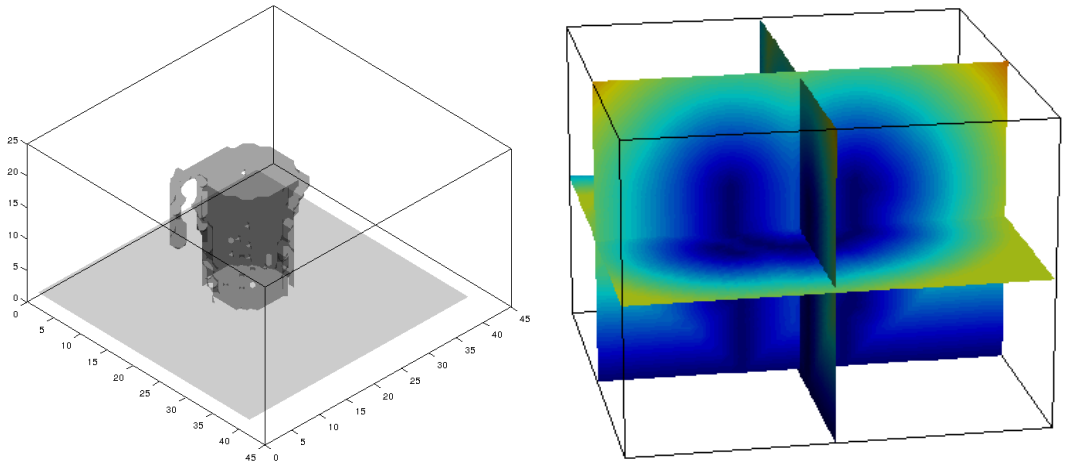


Figure 3.23: A close look of the object on the table and the slices of X, Y and Z planes of the velocities map around the object.

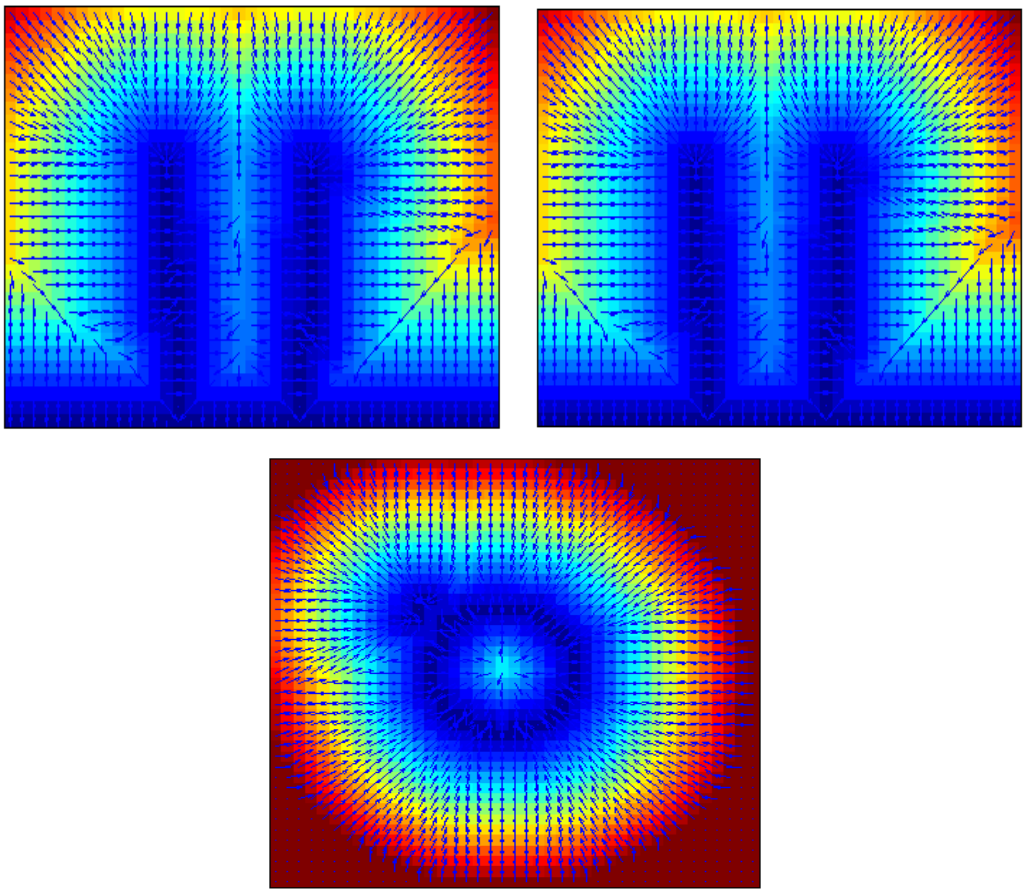


Figure 3.24: X, Y and Z slices of the velocities map respectively. The vectors indicate the direction of the gradient of these velocities.

This information is used to generate desired velocities for the fingertips, which generates an error as indicated by equation 3.9:

$$\dot{e} = \dot{x}_d - \dot{x} = \dot{x}_d - \dot{x}_f \tag{3.9}$$

where $\dot{e}$ is the difference between the desired and actual velocities of the fingertips. Cartesian velocity of the fingertip is computed as:

$$\dot{x}_f = J_A(q)\dot{q} \tag{3.10}$$

However, the control of movement of the fingers requires joint velocity references. Let's assume the analytic jacobian of the finger is squared and non-singular, then equation 3.11 is a equivalent lineal system [116].

$$\dot{q} = J_A^{-1}(q)(\dot{x}_d + Ke) \tag{3.11}$$

which fulfils

$$\dot{e} + Ke = 0 \tag{3.12}$$

It is known that when K is a positive matrix (normally diagonal) the system is asymptotically stable and the error tends to disappear as the trajectory is performed. Therefore, the trajectories generated for the fingertips lead them towards the surface of the object. Since it is assumed that the modelled environment might contain errors, if no contact is detected when the last desired velocity has been commanded, this velocity is kept until a contact is detected. When the contact between a fingertip and the object is produced, a constant velocity is kept so that the object can be hold. This velocity mainly depends on the object's weight and the friction between the hand and the object. The computation of this velocity is out of the scope of this thesis, therefore, for the experimental results it has been set empirically.

3.3 Evaluation with ManfredV2

The pipeline for grasp selection and the framework for planning and execution of the selected grasp explained throughout this chapter is evaluated in a real scenario. The robotic system in use is ManfredV2, described in detail in appendix B. The scene in which the tests are performed can be seen in figure 3.25. It is a table top scene in which the objects are placed on the table by an operator, at any pose, and then commands the robot to start the grasping pipeline. An important assumption is done in these tests: the objects can be grasped by the hand, meaning that it is not too big or small for being grasped by the hand.

Figure 3.25: Scene for grasp testing. Objects are positioned on the table for the robot to grasp them.

The selected objects can be seen in figure 3.26, they are presented in a 'standing' position (left) and 'lying down' on a surface (right). Both the cuboid and cylinder are symmetrical objects which, in a 'standing' position can be grasped from a top or side directions. Besides, a mug is also used. This object presents a handle, which makes it more difficult to success since it is not considered as a special case by the explained framework. Furthermore, the mug is small in comparison with the hand size, which forces to use a top approach direction for grasping to avoid collision with the table. The same happens when the objects are lying on the table and a side grasp is tried. After, collision and reachablility checking, the grasping possibilities of the objects are summarised in table 3.2. The boxes are filled with the grasps for which at least one sample is feasible. Then, F/P stands for 'Fingertip' and 'Power' grasps respectively and an X means that none of the possibilities were feasible.

Table 3.2: Grasping possibilities of the objects used.

	Standing		Lying	
	Side	Top	Side	Top
Cylinder	F/P	F/P	X	F/P
Box	F/P	F/P	X	F/P
Mug	X	F/P	X	F/P

Then, for every object, the feasible solution found is tried out with the robot. This

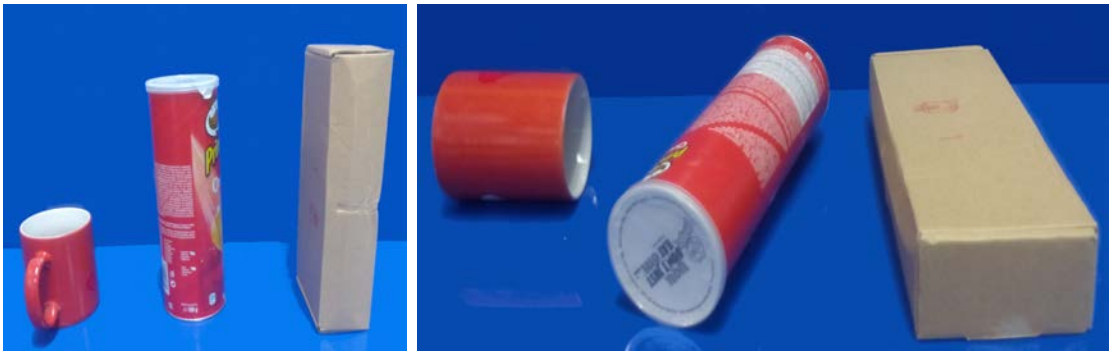


Figure 3.26: Objects used to test the grasping framework in standing (left) and lying poses (right).

is repeated 10 times for every object in which the operator chooses different poses for the object on the table. Although the parts of the pipeline developed for this thesis (environment modelling, grasp pose selection and motion planning for grasping) are deterministic, meaning that given the same input (object, table and robot poses) the same output is computed (grasp and motion planning), both the perception and execution parts are not. Besides, out of the tries, in half of them the perception algorithm is forced to recognise the object, therefore a full 3D model is used in the framework, while the other half will be performed using the point cloud of the object as an input. Finally, the parameters used among the different components of the pipeline are:

- Resolution of the world model and the voxelization of objects is $1cm$.
- Grasp candidates are computed every 30^{o}.

The results of this set of experiments are summarized in Table 3.3. For each of the objects we report the success rate Besides, PC stands for point cloud and model means that the object is recognised and its 3D model is used. A grasp is set as successful if, after the arm lifts the object, it is still holded by the hand.

These results allow to extract the next conclusions. First, it is clear that power grasps perform better than fingertip grasps, specially under non-recognition of the object, when the point cloud is used in the computation of the grasp. Also, it is important to note that, when the objects are lying on the table and a power grasp is tried, although the grasp pose does not lead to a collision, the fingers collide with the table while closing. This was tried once for every object and happened for all of them, so it was not tried any more, but it is the reason for not having any 100% in any case.

Table 3.3: Success grasping rate in the different objects under the situations tested.

	Power		Fingertip		
	PC	Model	PC	Model	
Cylinder	80%	80%	40%	60%	Side
	60%	60%	40%	60%	Top
Box	60%	80%	60%	80%	Side
	60%	80%	40%	60%	Top
Mug	40%	60%	40%	60%	Top

Among the objects, the cylinder can performs better with power grasps, since the fingers adapt better to the round shape of the cylinder. On the other hand, the box has better results when fingertip grasps are used because of its planar facets. Actually, fingertip grasp fails easily in the cylinder can when the computed centre of mass is not very close to the real one. Also, the mug is clearly the object which has a worst success rate. This is due to several factors: the handle provokes a displacement of the centre of mass and also it is difficult to get stable contacts on the handle. Besides, the material of the mug has made the object to slip from the hand twice.

Between the two types of modelling possibilities, as expected, using a full 3D model leads to better results than using the visible point cloud. This result is not surprising since it is obvious that the point cloud introduces bigger errors in both the centre of mass and axes computation.

Finally, figure 3.27 shows the final finger positions in different successful grasps of the different objects. On the left, the box lying on the table is grasped using a figertip grasp. In the middle, the mug is grasped from the top, this time, the fingertips did not slip over the mug surface and ring finger stabilised on the handle. On the right, the cylinder can is grasped from the side using a power grasp. Besides, figures 3.28 and 3.29 show the movement of the hand-arm system while moving from the starting pose to the grasp pose. In the fist case, the hand moves towards a top grasp of the box lying on the table , while in the second, the goal pose corresponds to a side grasp of the cylindrical can standing on the table.

3.4 Conclusions

This chapter presents a complete framework for object grasping: starting with how to model the environment, followed by grasp synthesis and path planning, and ending with a control scheme for treating a hand-arm system imitating a robot formation, for simplifying and high-dimensional problem. While for the first two phases use common approaches found in the literature, the application of the concept of a robot formation based control algorithm for the hand-arm system is completely new.

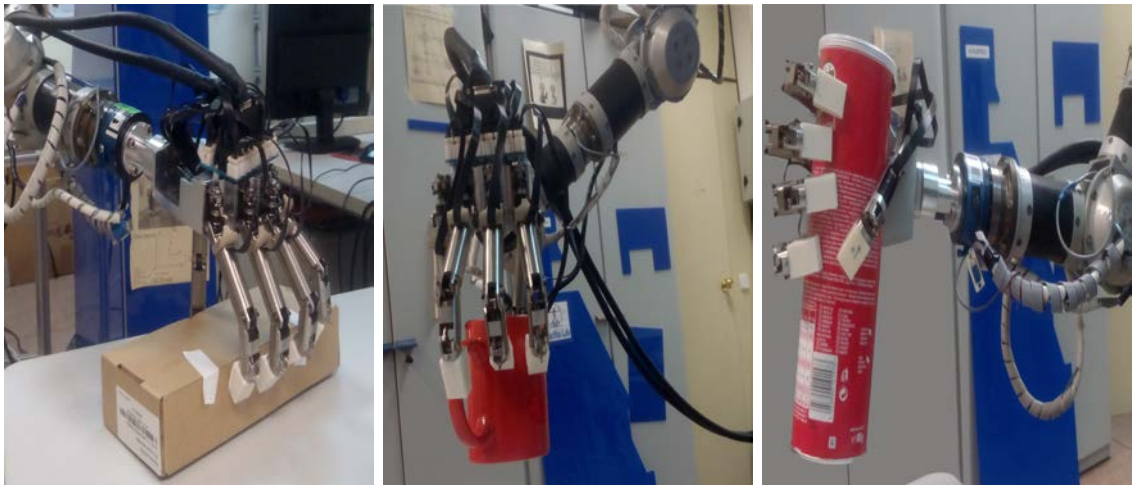


Figure 3.27: Examples of finger positioning in successful grasps.

The necessary movements of the hand to evolve, from the starting pose to the computed pre-grasp configuration, are controlled by treating the hand as a robot formation. This methodology is based on the algorithm introduced in chapter 2 and based on the FM^2 path planning method.

The use of FM^2 determines the need of an occupancy grid to model the environment, which then influences the choice of the algorithms used throughout the framework. When building an occupancy grid, it is important to pay special attention to the selection of the resolution. A high resolution implies a more accurate model of the environment, but it comes with larger computation needs in: object voxelizing, path planning and collision detection.

Simulations, carried out with Matlab, show examples of the use of the concept of robot formation based path planning an control in the case of both, side and top grasps. Then, test in real on-the-table scenarios have been performed. Although simple scenarios are used, it is always challenging performing real world grasping tests.

Results of the real tests show that it is possible to perform grasping actions executed under the robot formation concept presented. However, it is obviously not a final and perfect solution, with the majority of failures caused by the errors occurred in the object modelling phase, which are then propagated to grasp synthesis and path planning.

From the grasping results, two main improvements are suggested. When visible point clouds are used to model the objects, the grasping selection should be probably focused on the known parts of the point cloud. Otherwise, inference mechanisms to improve the point cloud are advised. Also, in the case of using fingertip grasping, it

would be interesting to compute approximate grasp positions and take into account the local shape around them to avoid slippery of the objects.

It is important to note that when a successful grasp is achieved, this approach does not guarantee any certain object pose in the hand. This is the reason that motivates the work presented in chapter 4.

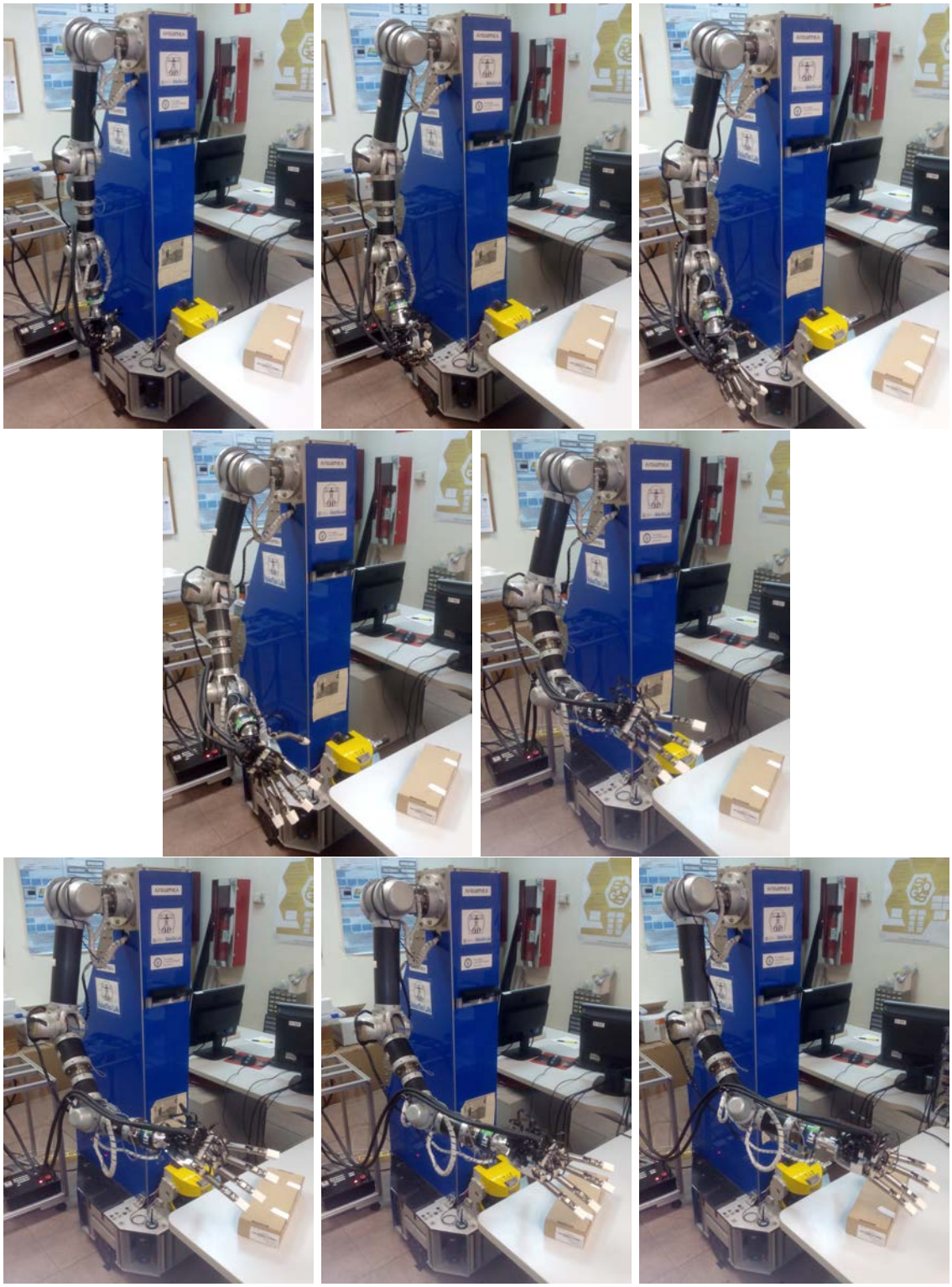


Figure 3.28: Movement of the hand-arm system towards a top grasp of the box lying on the table.

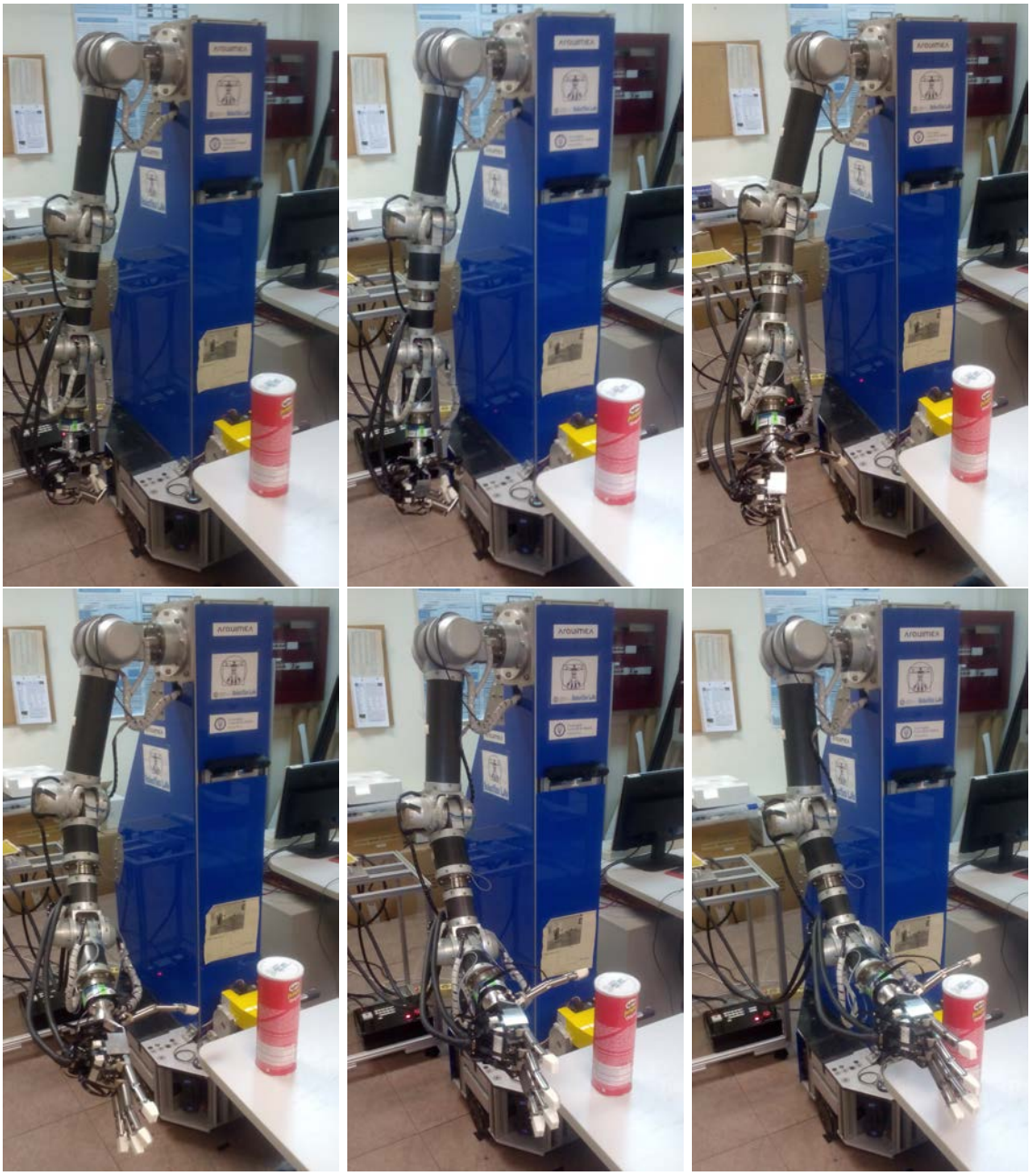


Figure 3.29: Movement of the hand-arm system towards a side grasp of the cylinder standing on the table.

Chapter 4

In-hand Object Pose Estimation

4.1 Introduction

Despite some uncertainties in the sensing, modelling, planning or execution phases, actually the grasping action often succeeds. However, for further manipulation actions, different than just pick and place, boolean success/fail is usually not enough information, since it is very hard to plan any further action if the pose of the object inside the hand is unknown, this is an important piece of information to retrieve.

In-hand object's pose estimation is a natural cognitive process made online by humans while grasping or manipulating objects. Several types of information are commonly used through this process: action, vision, tactile and semantic systems are linked in a dynamic and interactive manner [117]. When the object is inside the hand, tactile sensing is specially important since vision may be partial or totally occluded. In fact, experiments have shown that even humans fail to perform accurate manipulation tasks when their sense of touch is impaired [118].

The problem of finding an object's pose when it is grasped by a robot hand, either relying on vision, tactile sensing or a combination of both has been extensively researched. The most common combination of these modalities starts with a vision based pose estimation, which is later improved or corrected with the information provided by the tactile sensors. Nevertheless, several different approaches can be found in the literature. In [119], the vision estimate is refined by minimising the distance from the object's point cloud to the contact locations in the hand. Honda et al [120] used a combination of tactile and vision sensing to estimate an object's pose assuming that the object is composed of plain and quadratic surfaces. A different approach uses a description of the object's facets that is done offline and, during runtime, finds possible combinations of facets that match the current sensor measurements [121]. The authors in [122] include the idea of preventing the estimation of physically unfeasible solutions avoiding collisions between the object and the hand parts, the estimation process is made using a Particle Filter. In [123], also a Particle Filter is used to locate an object when it is being pushed with the hand in order to separate objects in cluttered environments. In [124], an approach that uses tactile sensors arrays to allow the recognition of the object and its 6D pose, by means of exploring the object's surface and edges using Iterative Closest Point combined with a particle filter, is presented. Object pose uncertainty can also be reduced by gaining tactile information from attempted grasps and replanning the grasp to increase the chances of success [125]. Estimation of an object's pose combining stereo vision and a force-torque sensor mounted on the wrist of a robot was reported by Hebert et al [126], who also used the joint position to estimate the location of the fingers with respect to the object's faces, an idea which similarly used in [127].

Other similar works can be found around these modalities. [128] addresses the problem of a global localization problem of the object, however, instead of directly

looking at the full 6D pose, Scaling Series is used together with a Bayesian Monte Carlo technique to iteratively refine the solution by increasing the granularity of the search region. In [129], a particle filter approach is used to estimate a tube's pose using both positive and negative contact information. Another approach was to model discrete states that contain the possible combinatorial arrangements between fingers and object surfaces using an hybrid systems estimator, estimating these discrete contact modes as well as continuous state variables, the object's pose [126]. Prats et al [130] combined tactile, force and vision information to locate and open a door handle and compared the performance under different sensing settings (only force, force and vision and tactile, force and vision), where the combination of the three modalities proved to outperform the other two, which is analogue to the previously mentioned results of Rothwell et al [118].

This chapter introduces a tactile based in-hand object's pose estimation. Reflex Takktile hand by RightHand Robotics [131], equipped with pressure based tactile sensors [132] and described in appendix B.5 is used. The methodology and algorithms used are explained along the next sections.

4.2 Problem Description

The method proposed in this research followes the one used in the work by Chalon [122]. In general terms, the used workflow included in a grasping framework would be: an initial object's pose estimation is provided by a vision based system (any other system used would also be valid). This estimation is used to compute a grasping pose, which is later executed and, hopefully, reached by a robotic arm. Once the arm has stopped, a reference frame located at the wrist joint, the one that connects the arm and the hand, does not move while the grasp is being executed. Therefore, the initial pose estimation can be transformed into the reference frame of the wrist, providing an initial guess of the pose of the object with respect to the base of hand. The parameters which define this estimation are shown in equation 4.1:

$$\begin{aligned} x &= [q, t]^T \\ x &= [q_x, q_y, q_z, q_w, t_x, t_y, t_z]^T \end{aligned} \tag{4.1}$$

where q corresponds to a rotation expressed in quaternion form and t corresponds to a translation vector. The quaternion representation was chosen to describe the rotations for its advantages in terms of computational efficiency. Furthermore, quaternions allow smoother interpolation when compared to other rotation conventions [133, 134].

Then, the fingers start closing towards the object and the tactile sensory system in the hand is used to detect when the fingers are in touch with the object. This event

makes the object pose estimation starts. The problem of finding an object's pose with respect to the base of the hand can be formulated as finding a set of parameters that define a transformation (a translation and a rotation, as in equation 4.1) that matches the current tactile and proprioceptive information.

The robotic hand used, ReFlex TakkTile Hand, is equipped with two types of sensors: pressure sensors located along the fingers and in the palm, and magnetic encoders in the proximal joint and the spool of the tendon that moves the fingers, as can be seen in figure 4.1. Pressure sensors are located along the fingers, while the hole in the proximal joint contains the encoder. The encoders provide the joint angular position allowing for the location of the proximal and distal phalanges, and therefore the pressure sensors, with respect to the wrist of the hand. Also the normals of the surface of the finger at sensor positions are known. Besides, pressure sensors deliver force measurements in each of the sensors based on the pressure produced on the rubber that covers the fingers and the palm. With the force readings, contact values are computed based on a force threshold. However, since the pressure flows through the rubber of the sensors, one single contact of an object with a finger often causes positive contact readings on consecutive sensors. When this situation occurs, a linear combination of the reading is performed to compute the actual contact as:

$$\begin{aligned} s_i &= [x_i, f_i] \\ x_i &= [q_i, t_i]^T \\ c_i = t_i + |(1 - \frac{f_i}{f_i + f_{i+1}})| &\times (t_{i+1} - t_i) \end{aligned} \tag{4.2}$$

being s_i the known information from each sensor: x_i the pose transformation and f_i the force reading of sensor i. Then c_i the computed contact location.

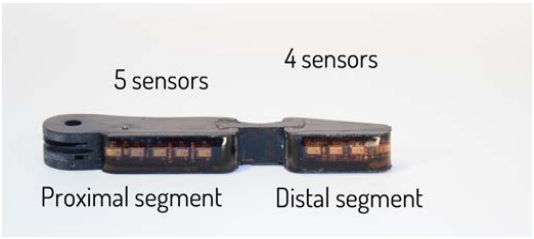


Figure 4.1: Overview of the pressure sensors in the fingers of the ReFlex TakkTile Hand.

With the information provided by the sensors, the search of a solution to this problem is done by combining the next general ideas:

- When a positive contact is detected by a sensor, the estimated position of the object must produce a contact at the same location. Figure 4.2 shows one example of this situation. The 3D models that correspond to the palm and the proximal and distal links of Reflex Takktile Hand are represented in green. Red dots on the models indicate the locations of the pressure sensors. The red arrows reveal the force reading of the sensors. Finally, the model of the object which is being grasped is presented in dark red. This figure illustrates a case in which two sensors are detecting a contact, however, the estimated pose of the object is far from producing the same contacts. This is an undesired situation.

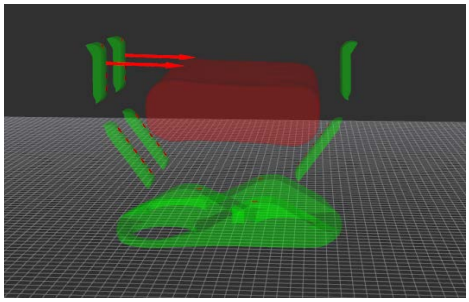


Figure 4.2: A positive contact is detected but the estimated pose of the object is far from producing a contact at the sensor location.

- The estimated position of the object should not be in collision with the hand, note the difference between contact and collision. Besides, the object cannot be floating in the air, not contacting with the hand at all. Figure 4.3 shows one example in which the fingers are in collision with the object (left), and other one in which there is no contact between the fingers and the object (right), both are undesired results.

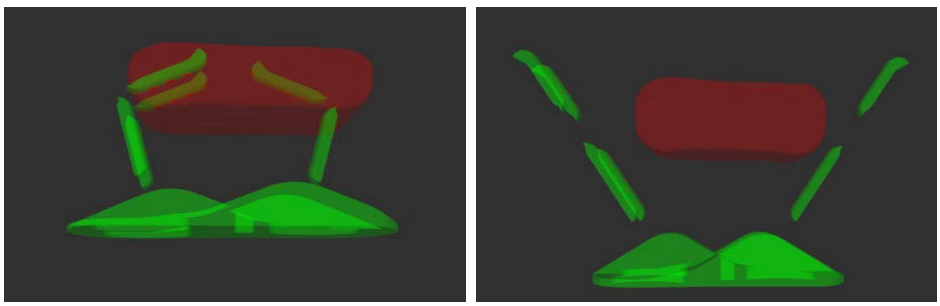


Figure 4.3: Examples of the fingers in collision with the object, and an object floating.

- The grasp execution is assumed to be successful, therefore, the normal of the surface of the object where the contact occurs and the normal of the contact locations in the hand should be close to parallel and pointing in opposite directions. The maximum deviation from being parallel depends on the friction produced between the object and hand materials, which is not known. In order to generalise, this angle is aimed to be as close to 0 as possible.

- The pose estimation is started when any of the fingers contacts with the object, therefore, if the fingers move, the object will be moved by the fingers. In figure 4.4, left image, the left fingers have made contact with the object and it is detected by the sensors (see the red arrows). Then, in the central image, the fingers have moved forward but the object stays at the same pose, this is not feasible. In the right image, the pose of the object is updated following the fingers' movement.

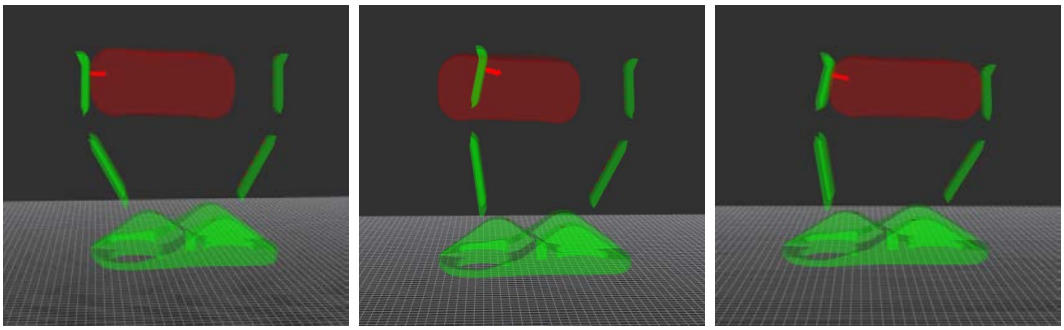


Figure 4.4: When the fingers are in contact with the object and move, the object should move with them.

Besides, two important assumptions are made before starting the object's pose estimation.

1. An initial pose estimation with respect to the hand is provided before starting. In this case, this corresponds to the object's pose detected by a vision system with respect to the pose of the base of the hand.

2. Geometric shape descriptions of the object and the hand are available. 3D models defined by vertex points and normals are used, as the ones shown in the previous figures.

Based on this information, a particle filter is used to generate new estimations of the object pose.

4.3 Bayesian Filtering

Bayesian filtering is a general probabilistic approach for estimating, recursively, an unknown probability density function over time using incoming measurements and a mathematical process model. One of the most popular methods for performing Bayesian filtering is the Kalman Filter. The Kalman filter can be applied to optimally solve a linear and Gaussian dynamic model [135]. When the linearity or Gaussian conditions do not hold, its variants, the extended Kalman filter and the unscented Kalman filter, can be used. However, for highly non-linear and non-Gaussian problems they fail to provide a reasonable estimate [136]. Besides, Kalman filters (regular or extended) perform better with an analytic description of the measurement model [122].

The proposed estimation method is also based on a Bayesian Filtering algorithm, the Bootstrap Particle Filter, a type of Sequential Monte Carlo method (SMC). These are a set of simulation-based methods which provide a convenient approach to computing posterior distributions sequentially. They avoid making linearity or normality assumptions required by related methods such as the Kalman filter. Importance sampling is used to approximate the distribution with a set of discrete values, usually known as particles, each with a corresponding weight.

In order to mathematically define how the Bayesian filtering works, let us start by defining two sets of random (vector) processes:

$$\begin{aligned} X_t &:= x(0), ..., x(t) \\ Y_t &:= y(0), ..., y(t) \end{aligned} \tag{4.3}$$

where X_t is a set of random variables of interest and Y_t is a set of measurements of the process of interest. Bayes' theorem for the conditional distribution can be expressed as

$$Pr(X_t|Y_t) = \frac{Pr(Y_t|X_t) \times Pr(X_t)}{Pr(Y_t)}. \tag{4.4}$$

Equation 4.4 shows how the posterior distribution, $Pr(X_t|Y_t)$, can be decomposed in terms of the prior distribution, $Pr(X_t)$, its likelihood , $Pr(Y_t|X_t)$, and the normalizing constant or evidence $Pr(Y_t)$. Furthermore, if the system is assumed to be Markovian, the distribution can be sequentially espressed as

$$Pr(X_t|Y_t) = W(t, t-1) \times Pr(X_{t-1}|Y_{t-1}) \tag{4.5}$$

and the weight is defined by

$$W(t, t-1) := \left[\frac{Pr(y(t)|x(t)) \times Pr(x(t)|x(t-1))}{Pr(y(t)|Y_{t-1})} \right] \tag{4.6}$$

This means that knowing the posterior distribution at the previous stage, $t-1$, scaled by the weighting factor, is sufficient in order to sequentially propagate the posterior to the next stage. However, this solution is not realisable unless the distributions are known in closed form and the underlying multiple integrals or sums can be analytically determined [137]. In fact, a more useful solution is the marginal posterior distribution [136] given by the update recursion as

$$Pr(X_t|Y_t) = \frac{Pr(y(t)|x(t)) \times Pr(x(t)|Y_{t-1})}{Pr(y(t)|Y_{t-1})} \tag{4.7}$$

where the update distribution can be considered as a weighting of the prediction distribution as

$$Pr(x(t)|Y_t) = W(t, t-1) \times Pr(x(t)|Y_{t-1}) \tag{4.8}$$

where the weighting factor is defined by

$$W(t, t-1) := \frac{Pr(y(t)|x(t))}{Pr(y(t)|Y_{t-1})} \tag{4.9}$$

4.3.1 Monte Carlo Approach

Monte Carlo (MC) methods, initially proposed by [138], are stochastic computational algorithms capable of efficiently simulating highly complex systems. They offer an alternative for solving classical numerical integration and optimization problems using Markov chain theory as its underlying foundation. Thus, under certain assumptions, the chain converges to the desired posterior distribution through proper random sampling as the number of samples become large [139], a crucial property.

The principal idea under MC techniques is representing a probability distribution as a set of samples from this distribution. The more samples taken from the distribution, the better representation is achieved. For example, let us define a function $f(X)$ with a probability distribution $Pr(X)$ so that the expectation can be defined as

$$E_X\{f(X)\} = \int f(X)Pr(X)dX. \tag{4.10}$$

Then MC integration evaluates equation 4.10 by drawing samples, $X(i)$ from $Pr(X)$. Assuming perfect sampling, this produces the estimated distribution given by the following probability mass distribution

$$\hat{Pr}(X) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(X - X(i)) \tag{4.11}$$

which used for substituting in equation 4.10 results in

$$E_X\{f(X)\} = \int f(X)\hat{Pr}(X)dX \approx \frac{1}{N}\sum_{i=1}^{N} f(X(i)) \equiv \bar{f} \tag{4.12}$$

where $\bar{f}$ is a MC estimate of $E_X\{f(X)\}$.

Typically, it is impossible to get such samples since $Pr(X)$ is multivariate, known only up to a constant of proportionality and non-standard. Markov chain Monte Carlo methods may be used in similar situations, but they are not well-suited to recursive problems. Alternatively, *importance sampling* can be used. Let us define

$$\begin{aligned} I = \int_X p(x)dx = \int_X \left(\frac{p(x)}{q(x)}\right) \times q(x)dx \\ for \int q(x)dx = 1 \end{aligned} \tag{4.13}$$

where $q(x)$ is the importance sampling distribution. It samples the target distribution $p(x)$ non-uniformly so that some values of $p(x)$ have more importance than others, overlapping the support region of the samples of $p(x)$. The key in importance sampling is choosing the distribution $q(x)$ which approximates best the target distribution $p(x)$.

Besides, we can use importance sampling to draw from the posterior distribution $Pr(X_t|Y_t)$ indirectly using an importance sampling density $q(X_t|Y_t)$ and a corresponding importance weight for each draw, as

$$\begin{aligned} \hat{f}(t) := E\{f(X_t)\} = \int f(X_t)\left[\frac{Pr(X_t|Y_t)}{q(X_t|Y_t)}\right] \times q(X_t|Y_t)dX_t \\ \tilde{W}(t) := \frac{Pr(X_t|Y_t)}{q(X_t|Y_t)} = \frac{Pr(X_t|Y_t) \times Pr(X_t)}{Pr(Y_t) \times q(X_t|Y_t)} \end{aligned} \tag{4.14}$$

Unfortunately, $\tilde{W}(t)$ is not useful because it requires knowledge of the evidence $Pr(Y_t)$, which is usually not available. However, it can be demonstrated [137] that the expectation $E\{f(X_t)\}$ can be approximated drawing samples from the proposal $X_t(i) \sim q(X_t|Y_t)$ and defining normalized weights as

$$\begin{aligned} \hat{q}(X_t|Y_t) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(X_t - X_t(i)) \\ W_i(t) = \frac{Pr(Y_t|X_t(i)) \times Pr(X_t(i))}{q(X_t(i)|Y_t)} \end{aligned} \tag{4.15}$$

and the final estimate is

$$\hat{f}(t) \approx \sum_{i=1}^{N} W_i(t) \times f(X_t(i)) \tag{4.16}$$

The importance estimator asymptotically converges to the true statistic and the central limit theorem holds [139]. However, this approach is still not well-suited for recursive problems. This is why *Sequential Importance Sampling* (SIS) is used, in order to be able to compute an estimate of $\hat{Pr}(X_t|Y_t)$ using the past simulated samples $X_{t-1}(i) \sim q(X_{t-1}|Y_{t-1})$. In order to do this, the importance distribution $q(X_t|Y_t)$ must admit a marginal distribution $q(X_{t-1}|Y_{t-1})$, implying a Bayesian factorization like in equation 4.17

$$q(X_t|Y_t) = q(x(t), X_{t-1}|Y_t) \tag{4.17}$$

which leads to the desired sequential solution [140]. The sequential update of the weight at each time-step can be written as:

$$W(t) = W(t-1) \times \frac{Pr(y(t)|x(t)) \times Pr(x(t)|x(t-1))}{q(x(t)|X_{t-1}, Y_t)} \tag{4.18}$$

which enables us to formulate a generic *Bayesian Sequential Importance Sampling* algorithm, a constrained version of importance sampling. As pointed out for equation 4.13, the selection of the importance sampling distribution is also crucial in equation 4.18. Several alternatives can be used here, being one of the most popular the use of the prior distribution as the importance proposal [136]. Unfortunately, it is well known that importance sampling is usually inefficient in high-dimensional spaces [141, 142] since as t increases, the weight of the particles degenerates, meaning that the number of particles with a weight value of 0 increases. Consequently, the particles with meaningful information of the posterior decreases and fail to represent the posterior distribution of interest.

A solution to this problem is proposed in the commonly named *Bootstrap Particle Filter*. It introduces a new step, resampling, which is intended to eliminate particles with low importance weights. Therefore, the fundamental concept in resampling theory is to preserve particles with large weights (large probabilities) while discarding those with small weights [143]. The overall strategy, when used with importance sampling, is called *Sequential Importance Resampling* (SIR) [136]. In order to apply it, a measure of the degeneracy of the particles has to be defined. A commonly used term is the effective particle sample size [139] defined by

$$N_{eff}(t) := \frac{1}{\sum_{i=1}^{N_p} W^2(t)} \tag{4.19}$$

which evaluates the effectiveness of the actual samples by looking at their weights, so that a smaller N_{eff} means a larger variance for the weights, hence more degeneracy. Therefore, resampling is performed if the effective sample size, N_{eff}, drops below a certain threshold, N_{thresh}, as in [139]:

$$\hat{N}_{eff}(t) = \begin{cases} Resample \leq N_{thresh} \\ Accept > N_{thresh} \end{cases} \tag{4.20}$$

Note that resampling is done so that a particle with a large weight is likely to be drawn multiple times and, conversely, particles with small weights are not likely to be drawn at all. Besides, the weights of the new particles are all set to be equal to $1/N$. Therefore, the resulting samples are uniform such that the sampling induces the mapping of

$$\begin{aligned} x_i(t), W_i(t) &\rightarrow \hat{x}_i(t), \hat{W}_i(t) \\ \hat{W}_i(t) &= \frac{1}{N} \ \forall i \end{aligned} \tag{4.21}$$

There are four commonly used resampling schemes within particle filtering literature, summarised in [144, 145]. All of them are unbiased, have $O(n)$ implementation and are based on a resampling of N particles (with replacement) from the original particle set of particles in which the probability of selecting a particle is equal to its weight [145]. The selected method to perform this step is the multinomial resampling, the simplest one among them. This resampling is achieved by repeated uses of the inversion method following these two simple steps:

- Generate N ordered uniform random numbers, u_k on the interval $(0, 1]$.
- Use them to select $\hat{x}_i(t)$ according to the multinomial distribution. That is,

$$\begin{aligned} \hat{x}_k(t) &= x(F^{-1}(u_k)) \\ u_k &\in \left(\sum_{s=1}^{i-1} w_s, \sum_{s=1}^{i} w_s \right] \end{aligned} \tag{4.22}$$

  where F^{-1} denotes the generalised inverse of the cumulative probability distribution of the normalised particle weights.

Graphically, if we draw a circle with the size if the perimeter assigned to the different particles in such a way that the length of the perimeter associated to each

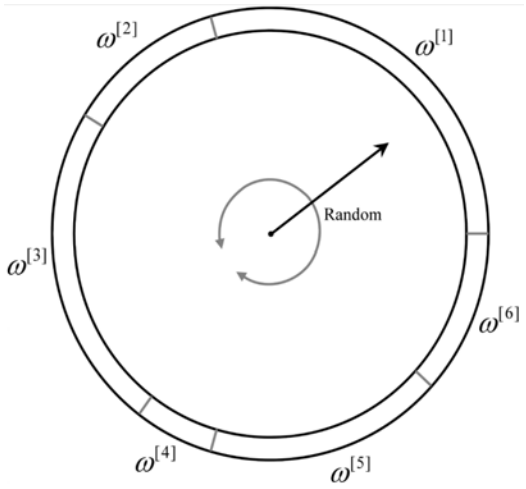


Figure 4.5: A circle whose perimeter is assigned to the particles in such a way that the length of the perimeter associated to each particle is proportional to its weight.

particle is proportional to its weight, as shown in figure 4.5, this method consists of picking N independent random directions from the centre of the circle and taking the pointed particle.

To sum up, the *Bootstrap Particle Filter* is implemented as shown in algorithm 1.

Algorithm 1 Bootstrap Particle Filter

1: **procedure** BPF($N_p, prior\_estimation$)
Initialization:
2: $x_i(0) \sim Pr(x(0))$ $W_i(0) = \frac{1}{N_p}$ ▷ Sample the prior
Importance Sampling:
3: $x_i(t) \leftarrow system\_model(x_i(t-1), input_t)$ ▷ State transition
4: $w_i \sim Pr(w_i(t))$
Weight Update:
5: $W_i(t) = W_i(t-1) \times measurement(y(t)|x_i(t))$
Weight Normalization:
6: $W_i(t) = \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)}$
Resampling:
7: $if\ \hat{N}_{eff}(t) \leq N_{thresh},\ then\ \hat{x}_i(t) \Rightarrow x_j(t)$ ▷ Posterior

4.4 Implementation

Once the theoretical framework has been presented, the actual implementation of each of its parts is explained.

4.4.1 System Model

The system model relates the state at time i which respect to the state at time $i-1$ and the input to the system at time i. In this case, since the state is the object's pose, this relation is given by the displacement. In order to compute the object's displacement based on finger movements, only the location of sensors which detect contact is taken into account. Let

$$s_i = [x_i, f_i > \textit{contact threshold}] \tag{4.23}$$

be a sensor location with positive contact. Then

$$\Delta x_i = [\Delta q_i, \Delta t_i]^T \tag{4.24}$$

is the displacement and rotation of sensor i between two consecutive readings. Then the total object movement is computed as the average translation and rotation, Δx_{ia}, among the sensors in which a contact is detected.

4.4.2 Measurement Model

The measurement model gives to each of the particles in the filter a weight that means how similar is, the state expressed by that particle, to the truth. This value is given based on the measurements provided by the sensors in the hand. In order to compare these data to the object's pose encoded by each particle, a simulation of the spatial relationships between the object and the hand is done. Since the 3D models of the object and the links of the hands are available, they are used to compute collisions between the object's pose encoded in each particle and the hand. These collisions are checked using the Flexible Collision Library (FCL) [146]. Using this library, the objects are modelled as bounding volume hierarchies (BVH) [147] and therefore fast queries on collision and smallest distance between the objects can be obtained. Consequently, it is used to compute a shortest distance (no collision) or deepest penetration (in collision) between each sensor and the object. The distance from each sensor position to the object surface, d_i^o, is negative if the bodies are colliding and positive if there is no collision.

Furthermore, kinematic data provided by the joint sensors of the hand, allows the calculation of the poses of the hand links, and therefore the positions of the force

sensors. Then, using real and simulation-based data and in the context of in-hand object localization, three kind of measurements are considered:

- For every sensor in which no collision is detected in the real hand, a probability value of the object's pose defined by each particle is computed based in its distance to the object by:

$$p_{nc,i}(d_i^o) = 0.5 * \left(1 + erf\left(\frac{d_i^o}{\sqrt{2}\sigma_d}\right)\right) \tag{4.25}$$

  where σ_d is a standard deviation value that can be adjusted to match the inaccuracy of the measurements, and erf corresponds to the error function. This function is chosen to assign high weight values to positive distances and small values to negative distances. The resulting function can be seen in figure 4.6 for $\sigma_d = 0.01$m (left) and $\sigma_d = 0.005$m (right).

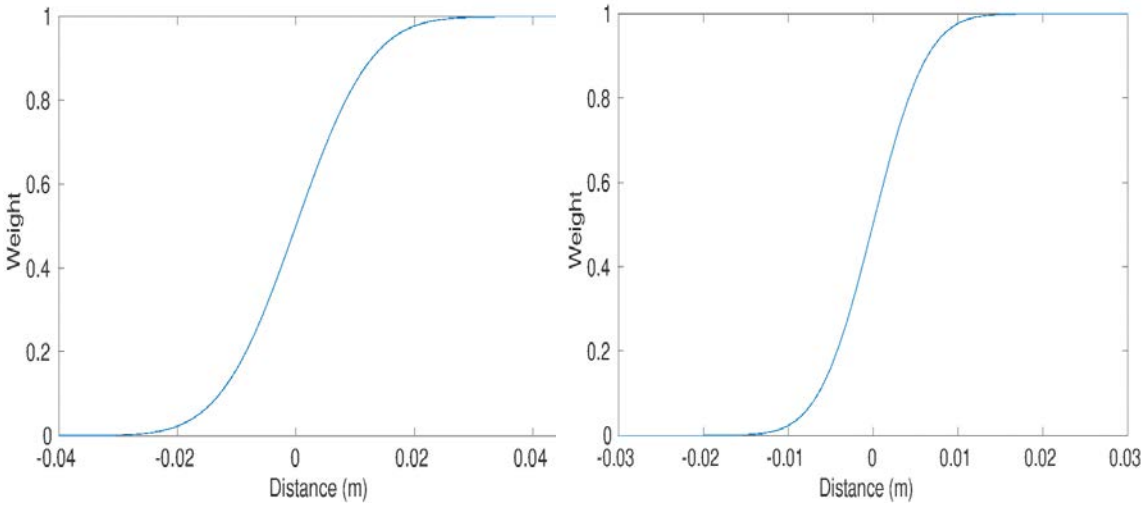


Figure 4.6: Function used in the case no contact is detected. In left picture, $\sigma_d = 0.01$m, in right one, $\sigma_d = 0.005$m.

- In the case of sensors in which contact is detected, measurements from tactile sensors are incorporated in a similar way. For each of them, the distance to the object represented by a particle is computed, d_i^o. Then, the probability for that measurement is computed as:

$$p_{c,i}(d_i^o) = e^{-0.5\left(\frac{d_i^o}{\sigma_c}\right)^2} \tag{4.26}$$

where σ_c is a standard deviation of the distribution that can be adjusted to account for the uncertainty in the force measurement. This function assigns high weights to values that are close to zero, so being close to contact. The resulting function can be seen in figure 4.7 for $\sigma_c = 0.01$m (left) and $\sigma_c = 0.005$m (right).

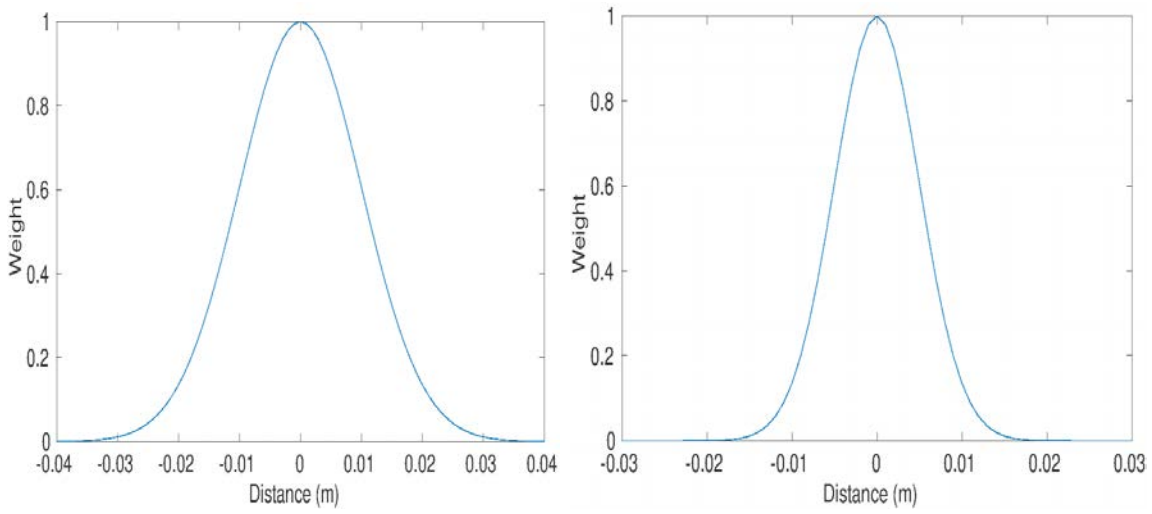


Figure 4.7: Function used in the case a contact is detected. In left picture, $\sigma_c = 0.01$m, in right one, $\sigma_c = 0.005$m.

- Since the obtained grasp is assumed to be stable, the contact surfaces in the object and the hand should be almost parallel. Therefore, for the sensors detecting contact, the angle (α_i) between the normals' surfaces is computed and evaluated by the next function:

$$p_{a,i}(\alpha) = e^{-0.5\left(\frac{\alpha_i}{\sigma_a}\right)^2} \tag{4.27}$$

where σ_a is a standard deviation of the distribution that can be adjusted to account for the uncertainty in the pose of the sensors. This function assigns high weights to values that are close to zero.

Under the next assumption: the weighting factors computed are independent from each other, a combined weight for each particle (w_p) can be expressed as:

$$w_p = \prod_{k=1}^{N_m} p_{nc,i} * p_{c,i} * p_{a,i} \tag{4.28}$$

where N_m is the number of measurements in each particle. This results in a weight calculated for every particle during the update step:

$$W_p = \left\{ w_p^{[1]}, w_p^{[2]}, ..., w_p^{[N_p]} \right\} \tag{4.29}$$

where N_p is the number of particles.

4.4.3 Inference of the best estimate

The objective of the in-hand localisation is to provide a best estimate $E(x_t)$ of the object pose among the set of particles. Assuming that the posterior of the particle filter represents a probability density function, similar to a Gaussian distribution, of the actual in-hand pose of the object, the expected value at each time step is the weighted average of this posterior. Let us recall that the object's pose is defined, as stated in equation 4.1, by a translation vector and a quaternion. Whereas for the translation, the weighted average is a straightforward computation:

$$\hat{x}_t = \frac{\sum_{p=1}^{N_p} w_p t_p}{\sum_{p=1}^{N_p} w_p} \tag{4.30}$$

where N is the number of particles, w_i is the weight of each particle and x_i is the 3D translation vector. However, the weighted mean of a set of quaternions is not so intuitive. If a similar simple procedure as for the translation is performed:

$$\hat{q}_t = \frac{\sum_{p=1}^{N} w_p q_p}{\sum_{p=1}^{N} w_p} \tag{4.31}$$

the given solution has two problems. The first one is that $\hat{q}_t$ is not a unit quaternion, thus it does not represent a rotation. This can be easily fixed by dividing $\hat{q}_t$ by its norm. The second problem comes from a property of quaternions when representing rotations which states that: q and $-q$ represent the same rotation, so that the quaternions provide a 2:1 mapping of the rotation group [148]. Therefore, if the sign of any quaternion q_p is changed, the resulting average should not change. However, it is clear that equation 4.31 does not have this property.

It is important to note that the goal is to average a rotation and not a quaternion. Following this observation and defining the vector and scalar parts of a quaternion as $q = [\varrho^T q_4]$, which obbey the normalization condition $\|\varrho\|^2 + q_4 = q^T q = 1$. Then, [149] proves that a 4×4 matrix M can be computed as:

$$M = \sum_{p=1}^{N} w_p q_p q_p^T \tag{4.32}$$

so that the average quaternion can be found following the next maximization procedure:

$$\hat{q} = arg\ max(q^T M q) \tag{4.33}$$

The solution to this maximization problem is known [150] and it can be found as the eigenvector of M corresponding to the maximum eigenvalue. This solution avoids both of the problems presented in equation 4.31. The eigenvector is chosen to have unit norm to avoid the first one. The second is avoided because changing the sign of any q_p does not change the value of M. Besides, the averaging procedure only determines q up to a sign, which is consistent with the 2:1 nature of the quaternion representation.

4.5 Experimental Setup

ReFlex TakkTile Hand has been used for real tests of the work presented in this chapter, which has been done in a research visit at the Institute of Robotics and Mechatronics of the German Aerospace Centre (DLR).

Although the robotic hand is made to be attached to a robotics arm for grasping purposes, this could not be possible to do because of technical issues, which could not be solved in the period of the visit to DLR. This makes impossible to solve the problem exactly as described in section 4.2, in which a common grasping pipeline (object pose estimation, grasping synthesis and execution) is performed. However, since the pose estimation is meant to be done with respect to the base of the hand, it is possible to perform the necessary tests while the robotic hand is held by a human operator, as can be seen in figure 4.8, left image. In the image, it is important to note the tag the hand has on the cover of the electronics and the one located on the object. Both of them are Apriltags [151], which are used to compute the ground truth pose of the hand and the object. Since the tag is located in the cover of the hand and not in its base point, a further transformation relating the tag on the cover to the base of the hand is needed. This transformation is extracted from the right picture in figure 4.8. Finally, the estimated pose of the object pose is related to its centre of mass, and not the one in which the tag is positioned, therefore, a transformation which relates the tag of the object and its centre of mass is calculated based on the size of the object.

Figure 4.8: On the left, the robotic hand lies on the table ready to perform a grasp. On the right, the tags needed to locate the reference frame for the pose estimation of the base of the hand.

Once the frame, from which the object's pose is computed, is available, the grasping is done as follows:

- Both the hand and the object are placed on a table surface. A camera is positioned so that the scene is recorded, extracting 640x480 pictures at a rate of 25Hz[1]. The Apriltags on the object and the hand cover are both visible, which allow for a ground truth computation at post-processing.

- The data recoding is started, then an operator grabs the hand from its base (while keeping the tag visible) and positions the hand at place from where a grasp can be done. 10 seconds after the recording started, the fingers close towards the object. Once all of them are in contact, a constant velocity is kept to make the grasp stable.

- At the moment the grasp seems completed to the operator, the object is lifted so that the grasp is proved to be stable, then the object is again put on top of the surface and the fingers open.

When applying the particle filter presented before for the pose estimation of the object inside the hand, one important question arises: when should the filtering start. There are two main possibilities:

- Waiting until the grasp is stable, and then look for the actual pose from the initial pose estimate before any contact was performed.

[1]All the images of the real scene in this chapter are taken from the colour sensor of an ASUS Xtion Pro Live.

- Start when any finger contacts with the objects. In this case, if the finger in contact moves, the object is assumed to be pushed and moved by it.

The first one depends on how much the object has moved from the first touch of the fingers to the moment the grasp is stable. If this movement is big, it might be difficult to have a particle representing a similar pose. Besides, it is not so clear how to determine that the grasp is stable. The latter is, in principle, more likely to be able to find a good solution if the movement towards the stabilization of the grasp is big. Therefore, the second one is used during the tests.

One important issue when using a particle filter is to decide the number of particles to use. It has to be a compromise between having a very big population, which increases the likelihood of obtaining a good estimate at the cost of increasing the computation effort, and having a small number which augments the frequency of the estimation but decreases the probability of having a good outcome. In order to choose the number of particles, both properties have been taken into account.

In order to choose the number of particles, tests have been performed using the scenario 1, which can be seen in figure 4.8, left image. The main characteristics of this scenario is that the hand lies on the table when executing the grasp. While performing the pose estimation, the period of time needed for a new result is computed against the number of particles. Since the whole grasping process (from the moment fingers start closing to the moment they open again) lasts for a few seconds (around 10 seconds), at least 1 new estimation per second is needed. 5 tests have been done using 100, 600 and 1000 particles. Computation times are shown in table 4.1. Time grows almost proportionally with the number of particles, but only the time for 100 particles is smaller than 1 second.

Table 4.1: The average time used to compute a new prediction and the number of particles.

Particles	Time (s)
100	0.3170
600	1.8772
1000	2.9871

Further tests are performed for different number of particles which would lead to update periods lower than 1 second: 100, 150, 200, 250 and 300 particles. This time, the results are analised in two different ways: again the time used for each update and the standard deviation of the posterior. Tests are repeated 10 times and results are averaged over the different the repetitions. The former can be seen in the left picture of figure 4.9, the green line shows the average time used among the trials, while the red area shows the standard deviation. In all cases the needed time is lower

than one second, which meets the set requirement, and there is a small difference among the trials. The latter is shown in the right picture of figure 4.9. Blue line is the standard deviation in the position estimation, while the red line is the orientation one. Theoretically, as the number of particles grows, the estimation of the particle filter is more accurate and therefore the standard deviation of the estimations should be smaller. Results show a tendency, which is that the estimations get better until 200 particles. Results are not very conclusive, since the values are really small and very similar among them (the number of particles changes very little), but in the case of the position, the standard deviation does not get any better for values higher than 200. Based on these results, 200 particles are chosen to be used in the rest of the tests.

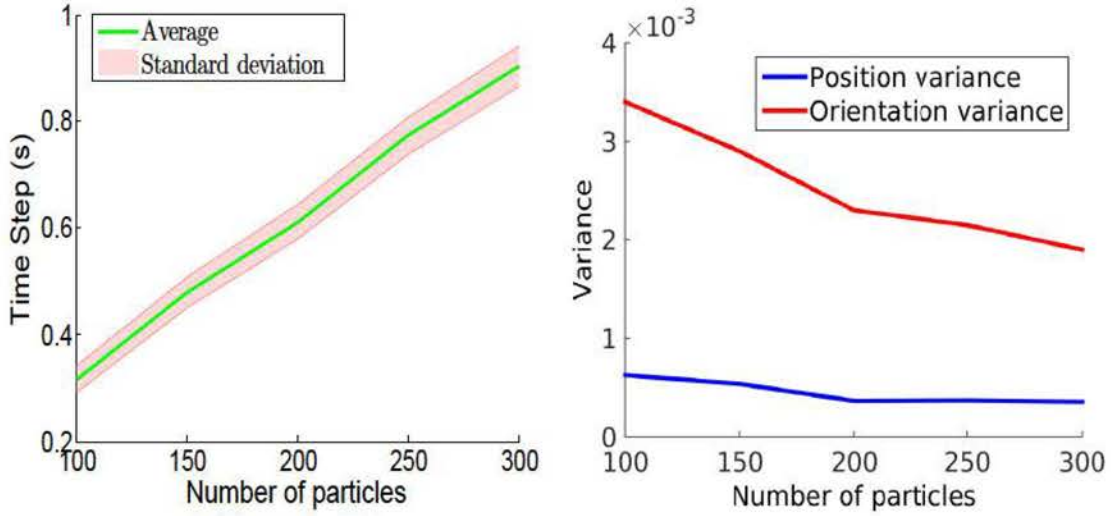


Figure 4.9: On the left, the average and standard deviation if the time used by the filter in each iteration against the number of particles. On the right, the standard deviation of the estimated pose against the number of particles.

Then, the pose estimation tests are performed, as described before, while grasping three different objects selected from the YCB database [152]. Figure 4.10 shows the real objects (on the top row) used and their corresponding 3D models (on bottom row), as provided in the database, used in the measurement system, table 4.2 shows their size and weight. It is possible to appreciate that the 3D models include errors, specially in the edges and the texture, although texture is not used in this project.

Table 4.2: Characteristics of the objects used for pose estimation.

Object	Size (mm)	Weight (kg)
Cheez-it	60 x 160 x 230	0.453
Pringles	75 x 250	0.205
Jell-o	35 x 110 x 89	0.187

Figure 4.10: On the top row, pictures of the real objects used in for testing of the system. On bottom row, the 3D models provided in the YCB database.

Furthermore, the values used for parameters explained through this chapter are summarised in table 4.3. For the input to the system (displacement between iterations), a Gaussian noise, with 0 mean, is added to both position and rotation. To the contrary, no noise is added to the measurements, since the data provided by the sensors is quite noisy itself, so no artificial one is included. For the weighting function, values have been chosen empirically. Specially, σ_d and σ_c were tried to be 0.5 mm, but this made all the particles to have a weight value of 0 in most of the iterations, which is completely useless.

Table 4.3: Parameters used for the pose estimation with the proposed filter.

Input Noise	Position Mean	0
	Position Standard Deviation	0.7 mm
	Rotation Mean	0
	Rotation Standard Deviation	0.35º
Measurement Noise	Mean	0
	Standard Deviation	0
Weighting Function	σ_d	1 mm
	σ_c	1 mm
	σ_a	8º

4.5.1 Results

The pose estimation for each of the object has been performed 10 times. The results shown next are an average of all of them. In all the cases, the pose estimation is started when the first contact between the object and the hand occurs, and it is stopped when the hand is commanded to open. For all the tests, results show the estimation of the position (cm) in the three axes and the rotation (º) around each of them. Since the estimation is made with respect to the base of the hand, the orientation of the axes of this frame is always indicated, in the first image of the figure that shows pictures of the real scenario, to help understanding the results. All the results show the ground truth computed using the AprilTags (GT) in a red line. Also, the best particle of the population at each iteration is drawn in a purple line. Besides, the mean of the posterior and its standard deviation are plotted as a blue line and a shaded green area respectively. Note that the scale of the graphics is different among them images on the results. Also, it is important to keep in mind that all the given data are poses of the centre of mass of the object with respect to the base of the hand.

Pringles can

Tests on the pringles can have been done twice. In the first scenario, the hand lies on the table while grasping the object In the second one it is held by the operator and the object is grasped around the middle of the height of the cylinder.

Figure 4.11, shows images of the whole process of grasping for the first case. The sequence of images is ordered column-wise, meaning that the first four images correspond to the first column, then from the fifth to the eight correspond to the second column. Its main characteristic is that the hand performs the grasp while lying on the table surface, and no operator is needed. In the first image, the starting pose of the object and the hand are presented. Besides, in the lower corner of the left

side of the picture, there is the reference frame which is associated to the base of the hand. In the second image, the fingers already started moving towards the object. In the third one, a contact between the proximal link and the object is produced, then the fingers keep closing and in the fourth image a new contact is made with the distal link. Note how, in the fifth and sixth images, due to the force applied by the fingers to stabilise the grasp, the object moves towards the palm, which corresponds to the direction of -Z axis. Besides, although it is difficult to appreciate visually, the object also in the direction of the -X axis. This movement happens because the two fingers in the same side of the can, at the top of the image, perform larger force than the one that actuates in the opposite direction. The last two images correspond to the opening movement.

Results of the pose estimation can be seen in figure 4.12. The left column shows the estimation of the position in X, Y and Z axis respectively, while the right column shows the estimation in rotation around each axis. The main two displacements are produced when the hand grasps the object, which is moved towards the palm, around 1 cm, and in the -X direction, around 8 mm. Both movements are well detected in the ground truth measurements (GT) and in the estimation by the filter. However, the changes in the estimation are slower. Note how the GT also moves in the Y axis, around 5 mm in total, although this is not possible since the base of the can never moves from the table. Measurements in this axis are a little noisy in all the experiments, which are intrinsic errors of the pose estimation provided by the tags. It is also interesting that the estimation of the movement in the Y axis is very similar, although the filter does not have any measurement in the Y axis and therefore it is not possible to establish at what height the object is being grasped. This movement is likely to be caused by the noise introduced in the update of the system. Note that the standard deviation is much smaller in the case of the X axis, due to a higher constraints produces by the fingers. Among the rotations, GT does not detect almost any movement, and the filter estimates small turns of the object which would not produce an unfeasible situation.

It is important to note how the particle with highest weight, drawn in purple, is much more unstable the solution provided by the mean of the posterior.

Figure 4.11: Scenario 1, a Pringles can grasped while the hand lies on the surface of the table.

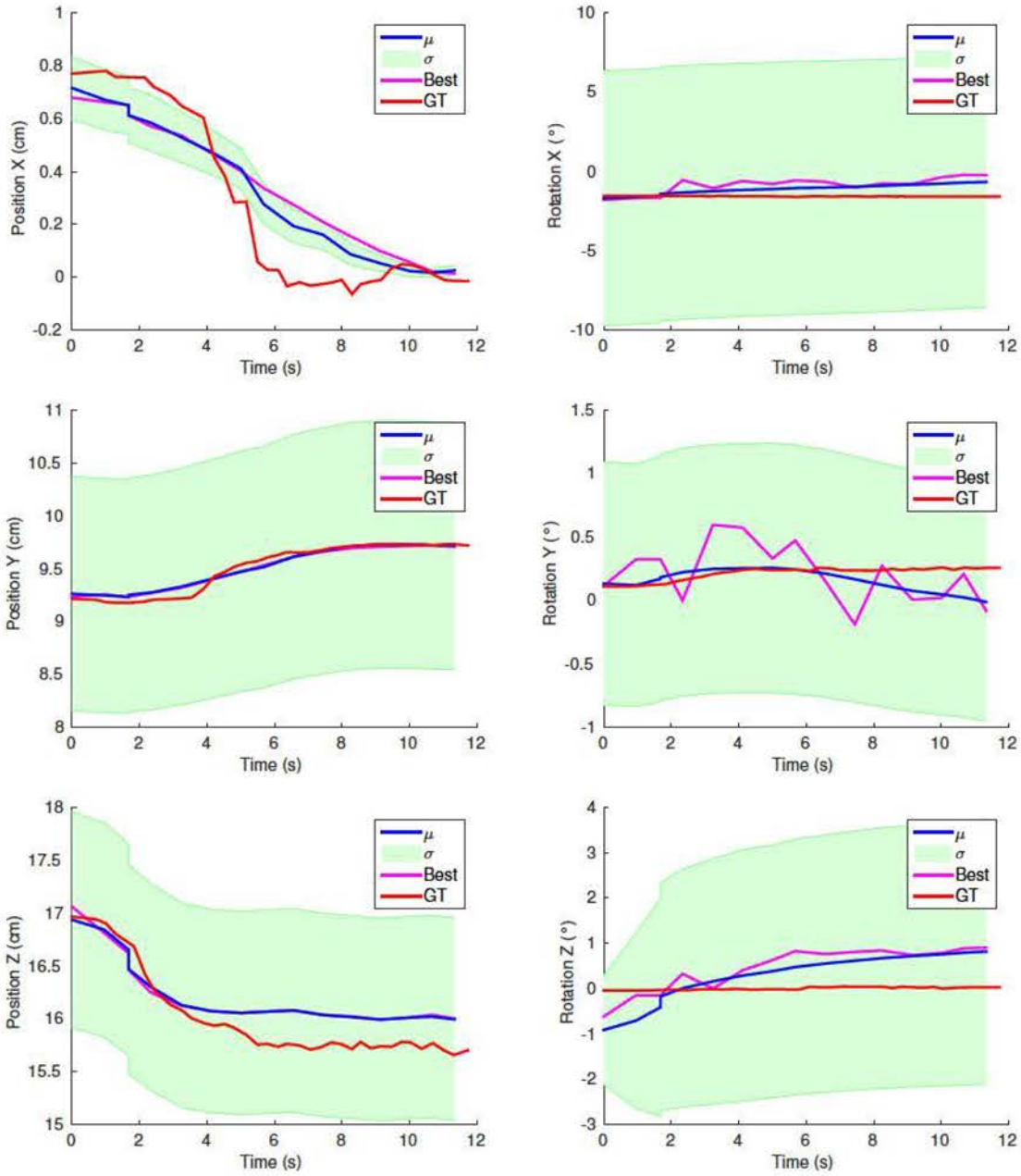


Figure 4.12: Results of scenario 1, the hand on the table grasping a Pringles can. Left column corresponds to the position in X, Y, and Z axes. Right one is the rotation around the same axes.

Figure 4.13 shows the grasping process in the second experiment performed with the Pringles can. Again, the sequence is ordered column-wise. In the first image, the starting pose of the object and the hand are presented, on the left, note the hand of the operator holding the robotic hand. In the second one, the proximal link of the fingers have made contact with the object, the fingers keep closing until the distal link also makes contacts, as shown in the third image. Note how the can has turned around Y axis (parallel to the normal of the table plane) from the second to the third image. In the fourth and fifth images, the hand is lifted and turned towards the camera, and the object does the same because there is a stable grasp. However, since the computed pose is referenced to the base of the hand, this movement does not affect the actual pose of the object in the hand. Then, in the next images, the hand is moved down and the object is put back on the table and the fingers open.

Figure 4.14 shows the results of the pose estimation for this scenario. Both the ground truth and the estimation show slight movements in X and Y axis, but they are not very relevant. The one in the Y axis can be considered as noise, since is has a range of less than 1 mm. The one in the X axis is likely to be caused by the different moments the fingers contact the object and the force made for the stabilisation of the grasp, as also happens in the Z axis. In fact, results are very similar to scenario 1, although in this case the estimation provided by the filter is more accurate. So the introduction of an operator holding the hand instead of using a robotic arm seems to be useful to test this algorithm.

In a similar way as happened in scenario 1, ground truth does not detect almost any rotation at all, but the estimations show slight variations around 0^o. Note that the rotations around Y axis make the cylinder rotate around its axis of revolution. From a textureless 3D model point of view, these rotations do not make any difference, which still fulfills the physical constraints of the real object being grasped.

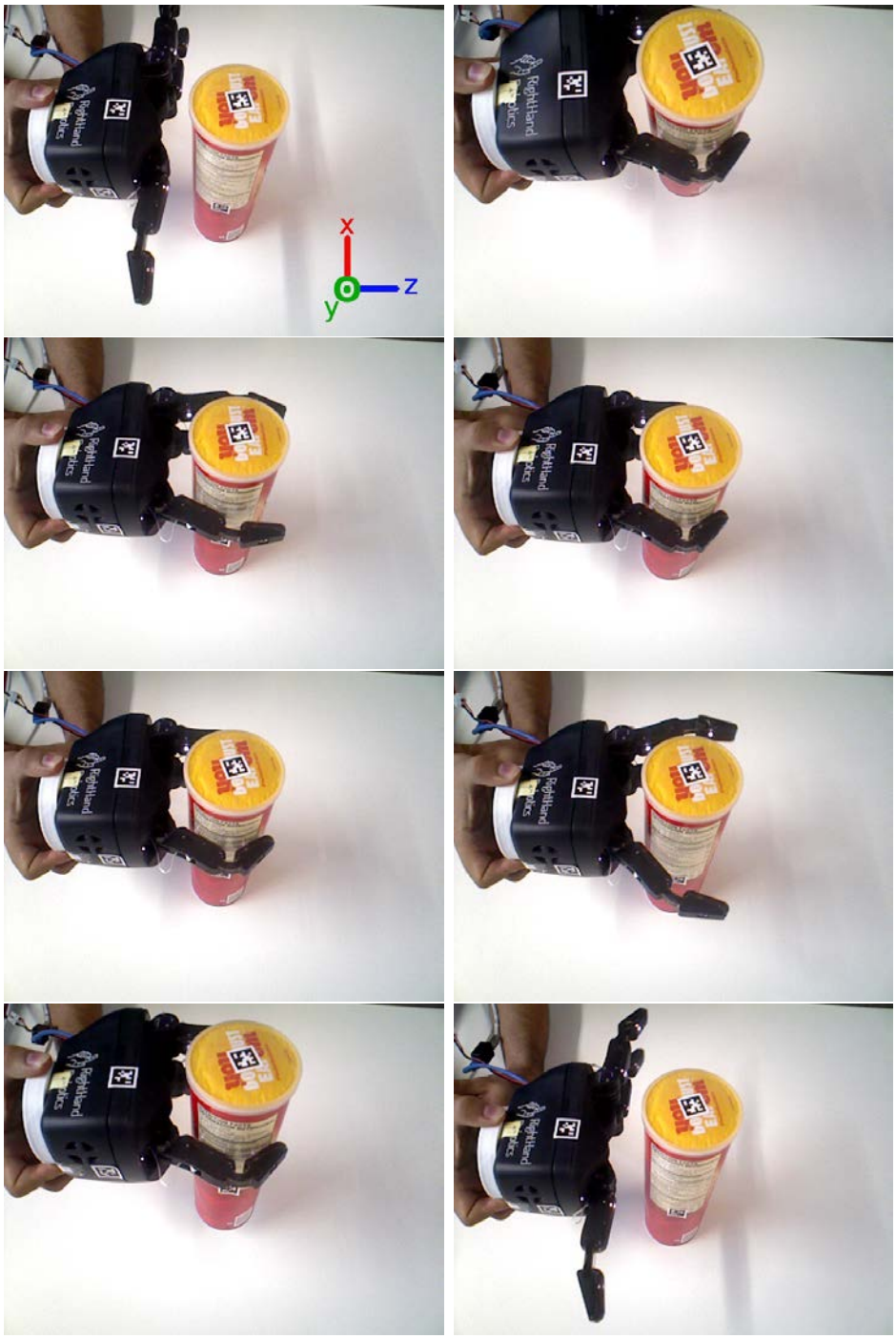


Figure 4.13: Scenario 2, a Pringles can grasped from the side while the hand is held by an operator.

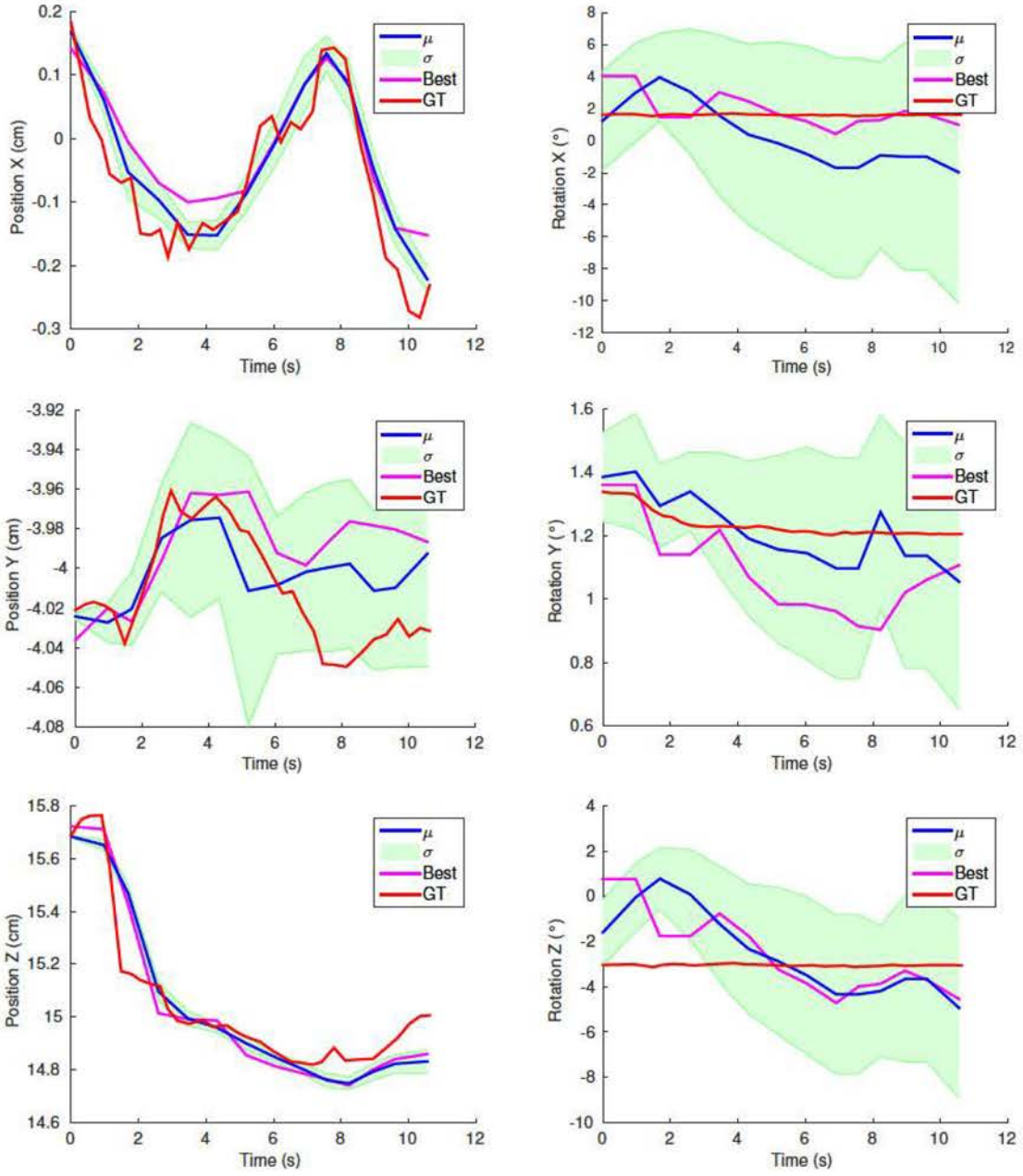


Figure 4.14: Results of scenario 2, the hand on the table grasping a Pringles can. Left column corresponds to the position in X, Y, and Z axes. Right one is the rotation around the same axes.

Cheez-it box

The Cheez-it box is a heavy and big object for the hand with a slippery surface. A side grasp has been chosen to lift it since in case of slippage, the palm of the hand might contribute to holding the object. The resulting grasp and the lifting and turning processes can be seen in figure 4.15. As in the previous cases, the sequence is ordered column-wise. In the first image, the starting pose of the object and the hand are presented, besides, in the lower-right corner of the picture, there is the reference frame which is associated to the base of the hand. In the second and third pictures, the proximal and distal links make contact with the object respectively. Next, the lifting and rotation of the hand are shown, which provokes the same movement on the object. Then the box in left again on the table and the fingers open.

Figure 4.16 shows the results of the pose estimation for this scenario. In all the cases of displacements and rotations, ground truth only measures really small movements, less than 1 cm and 1^{o} respectively. Looking at the images of the scenario, it is possible to think that there is an evident rotation around the Y axis, and the displacement towards the palm (-Z axis) between the 2^{nd} and $3^{rd}/4^{th}$ images. However, due to the weight of the box, the operator provokes these movements, not affecting the hand-object pose. Rotation estimations are very stable, but have a big standard deviation.

The estimations behave in a similar way, although more noisy. It is of special interest the results in the Z axis, more precisely, the standard deviation in this axis, which is quite high. The problem is that it is very likely that the pressure sensors only detect a contact between the fingertips and the surface of the box. So, when there are not additional contacts with the palm or the proximal phalanges, the fingertip contacts could be produced anywhere along the surface of the box, as long as the box does not collide with the palm. This allows the particles to have the same weight although their position in the Z axis changes.

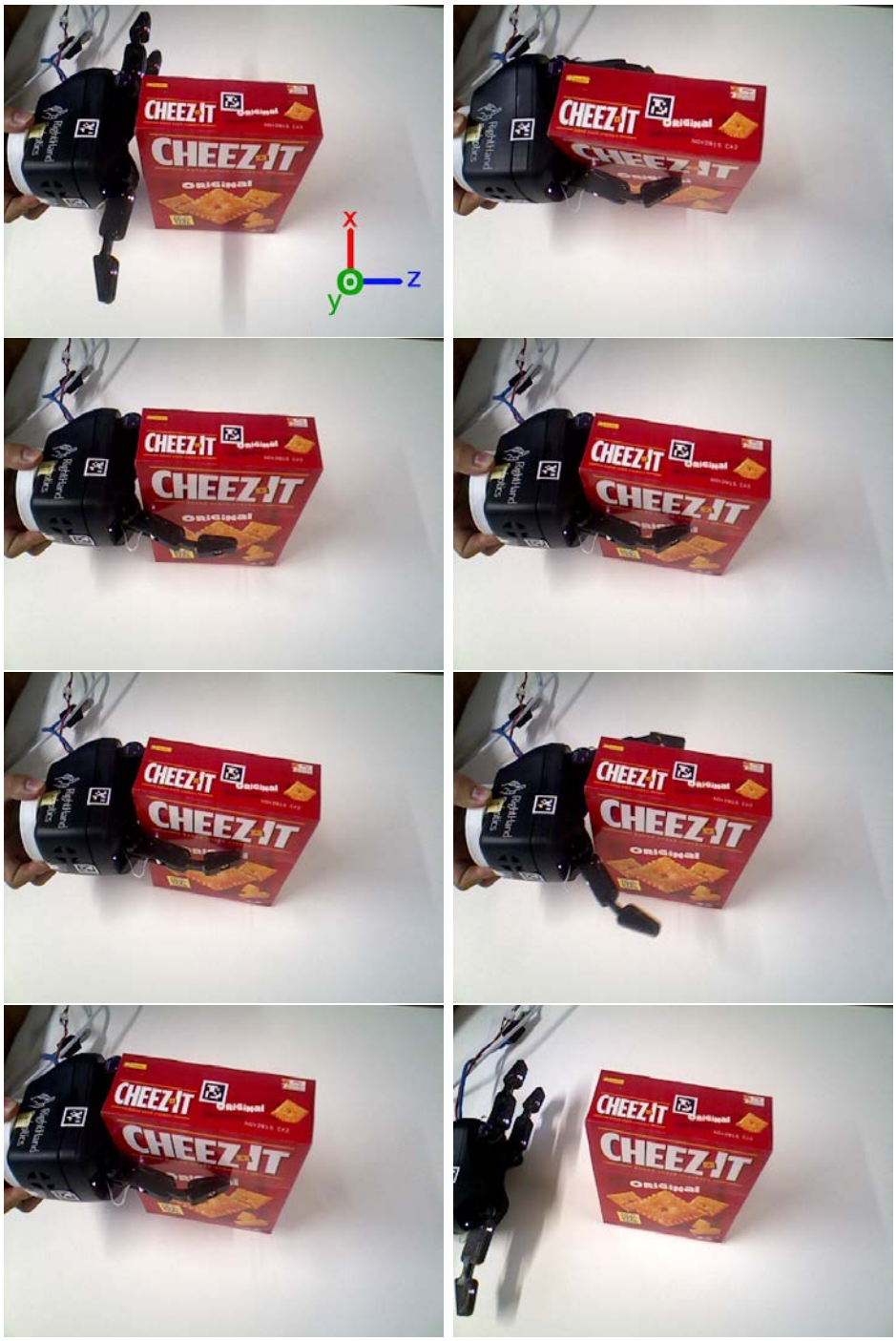


Figure 4.15: Scenario 3, a cereals heavy box grasped from the side while the hand is held by an operator.

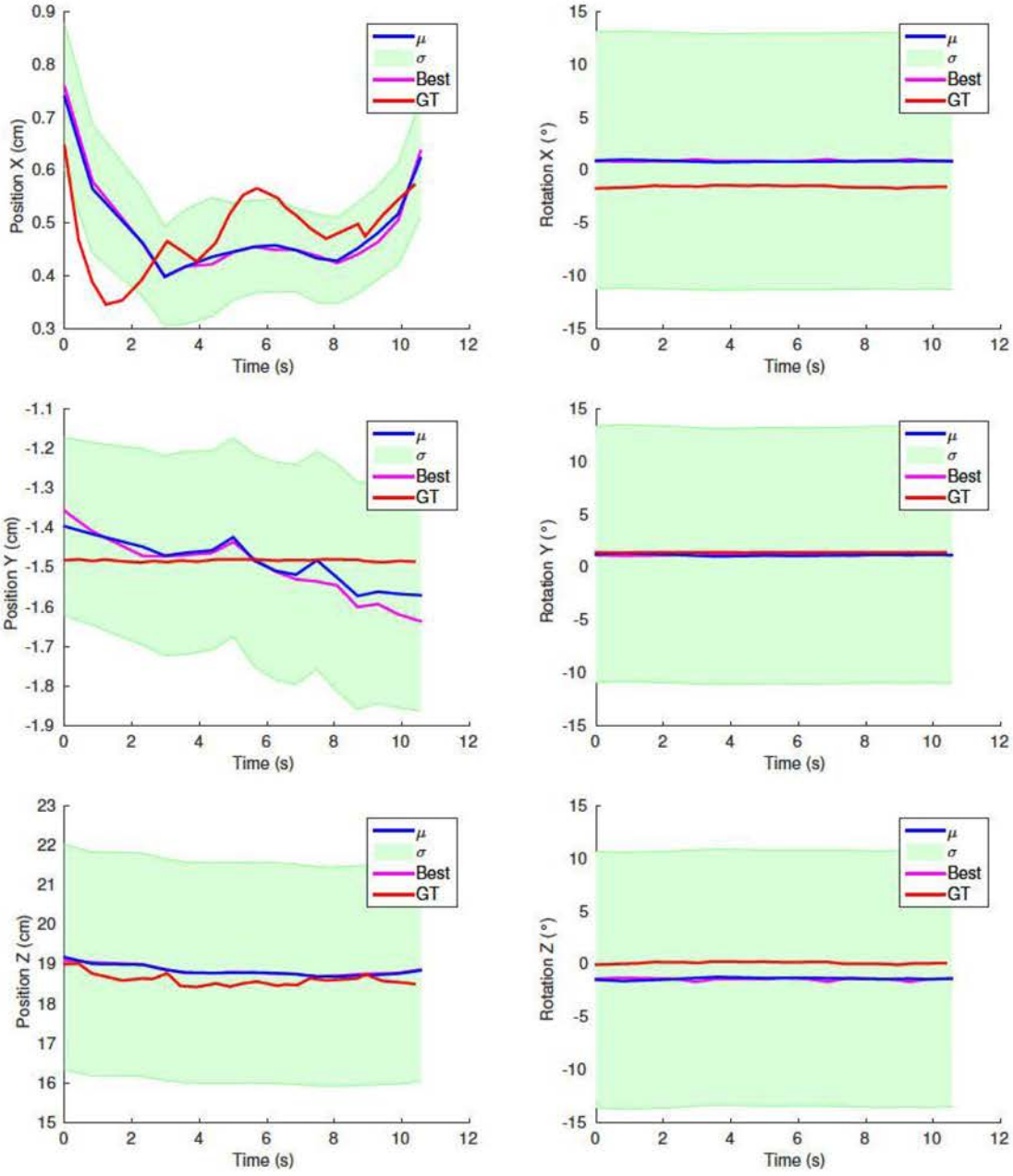


Figure 4.16: Results of scenario 3, the hand on the table grasping a Pringles can. Left column corresponds to the position in X, Y, and Z axes. Right one is the rotation around the same axes.

Jell-o box

Finally, a chocolate powder box is used. In this case, the object lies on the table so that it is only possible to grasp it from the top. The grasping and lifting processes can be seen in figure 4.17, which are organised column-wise. Note that the orientation of the reference frame of the base of the hand changes with respect to the previous tests. Nevertheless, it is depicted in the lower-left corner of the first picture.

The first image presents the starting pose of the object and the hand. In the second one, only two fingers have moved and made contact with the object. This is due to a false positive contact measurement in the pressure sensors of the third finger as if it had already contacted with the object. After that, when more force is applied on the object by the fingers (to stabilise the grasp), the finger on the left starts closing, while the fingers on the right provoke a movement of the box towards the left of the image, -X axis direction. This can be appreciated in the 2^{nd} and 3^{rd} images. This movement stops when the left finger makes contact (4^{th} image). Finally, next images show how the box is lifted with the hand and put back down on the table before the fingers open.

Figure 4.18 shows the results of the pose estimation for this scenario. There are two main displacements which can be visually appreciated in the pictures, towards -X (as explained before) and -Z axis. The second one is produced when all the fingers are making force against the object. Note the difference in the orientation of the fingertips among 3^{rd}, 4^{th} and 5^{th} images, which provokes the displacement. Both movements are well detected by the fingers. In the case of the rotations, none are detected by the ground truth, and only very slight movements estimated in the filter.

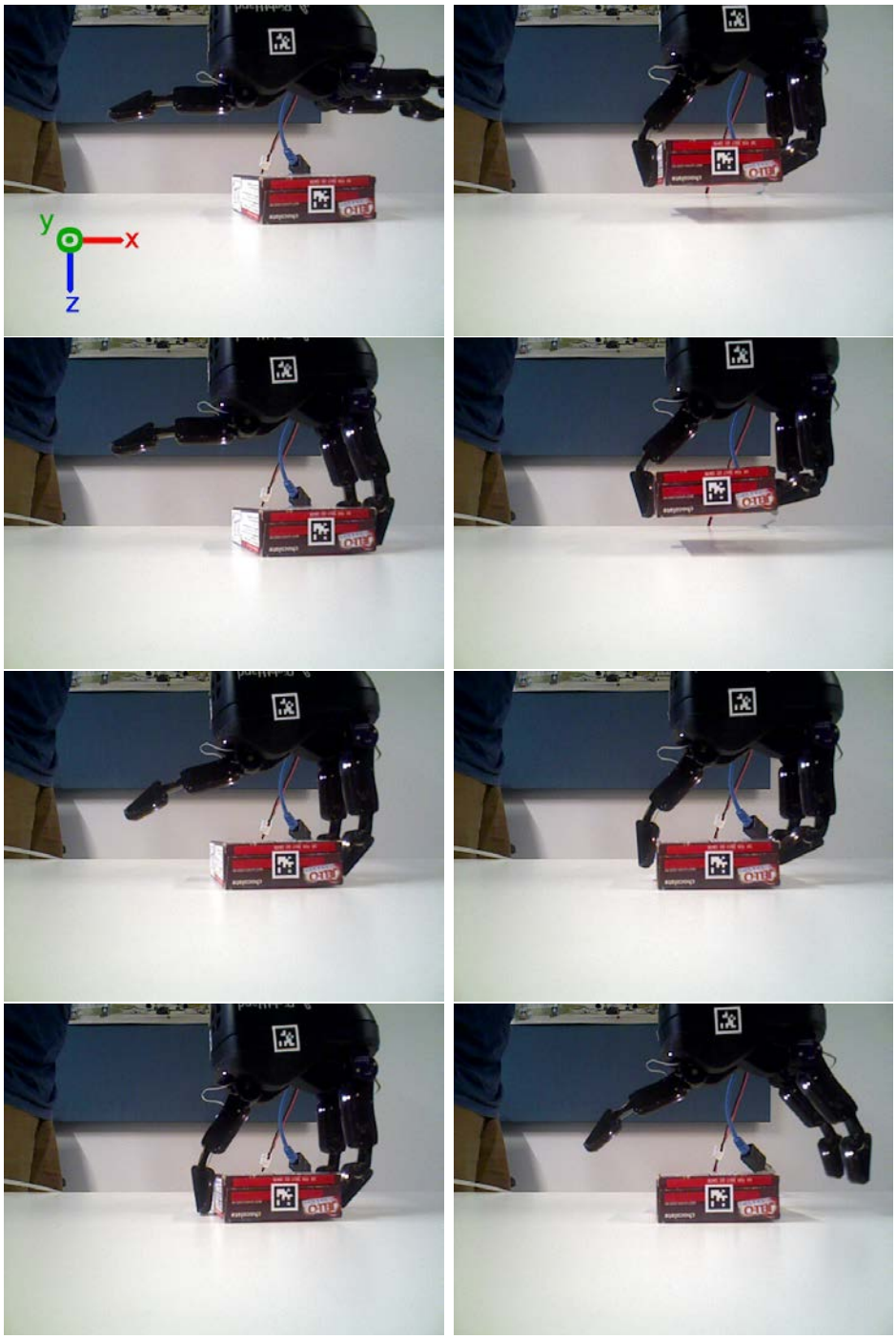


Figure 4.17: Scenario 4, a small chocolate powder box grasped from the top while the hand is held by an operator.

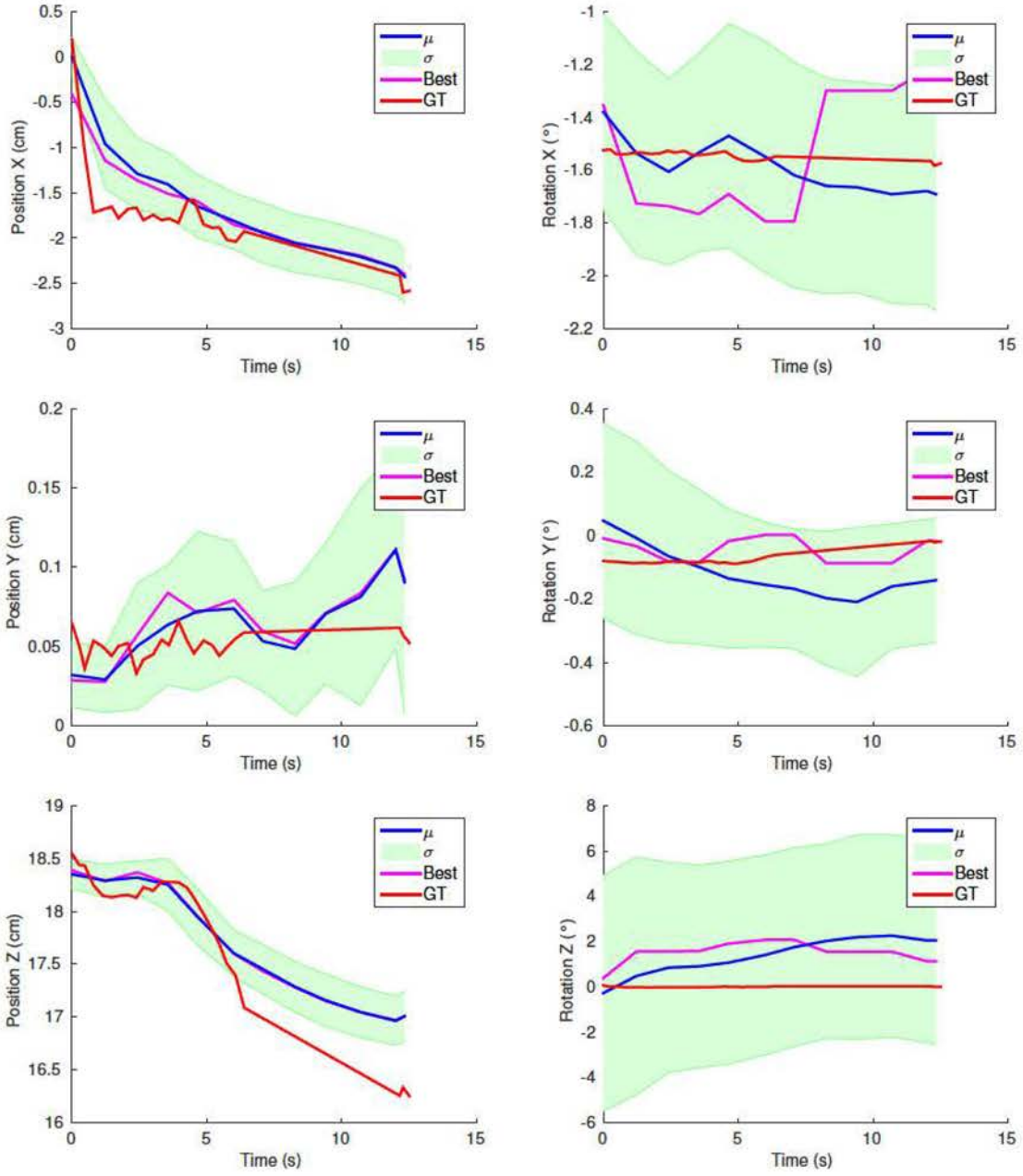


Figure 4.18: Results of scenario 4, the hand on the table grasping a Pringles can. Left column corresponds to the position in X, Y, and Z axes. Right one is the rotation around the same axes.

4.6 Conclusions

In this chapter, an in-hand pose estimation algorithm is proposed to detect the movement of the object that occurs from the first contact between the hand and the object, until the moment the grasp becomes stable. A Bootstrap particle filter is selected for the estimation mainly because of its flexibility in the measurement system.

The proposed estimation process is based on the measurements provided by the sensors of the hand: tactile sensors for contact detection and joint encoders for proprioceptive information. The measurement model proposed for the filter is based on a simulation of the position of 3D models of the hand and the object. Then, a collision detection library is used to compare real measurements with simulation based ones. Two new characteristics have been include in the measurement system: the actual force detected by the sensors is used to compute more accurate contact locations. Besides, the orientation of the surfaces in contact is taken into account.

The algorithm is tested in real grasping scenarios with three different objects. Results show that the main movements produced while stabilising the grasp can be estimated.

The main problem arises in the deviation of the estimations in the degrees of freedom in which there is not measurement update. Results may improve by introducing a new measurement term which penalises movements higher then those used in the input of the system. However, increasing the number of variables in the measurement computation makes it harder to predict its behaviour. A more straightforward solution would be decreasing the search space to those axes in which a measurement is not provided. However, it is important to avoid introducing case-specific restrictions, but instead it could be a combination of object characteristics extraction (such as symmetries) and how to relate them to the selected grasp pose.

In my opinion, the most promising manner of improving this system is its combination with a visual based pose estimation. Although, in general, after the grasping has been executed, the visual information is likely to be occluded (at least partially), it would be probably very useful to provide insights on the degrees of freedom that cannot be detected with tactile sense.

Chapter 5

General Conclusions

This Thesis shows a complete scheme to perform grasping actions in on-the-table situations. Besides, an in-hand pose estimation scheme is introduced in order to close the control loop after the grasping is executed. In my opinion, having a continuous action-sensing loop is a key factor for the success of robots performing complex manipulation tasks.

In chapter 2 , previous works are extended by adapting the formation shape control scheme to 3D environments, and includes the necessary changes to apply the algorithm to non-static environments. Besides, it introduces a novel approach to model mobile obstacles in the environment in a way that they are, later on, easily treated by the obstacle avoidance algorithm explained. Finally, quantitative analysis of this approach has been carried out. The results explain how the formation evolves avoiding obstacles while covering the planned path. The tests show that the proposed shape deformation method, in combination with the FM^2 path planner, is robust enough to manage autonomous movements through an indoor static and dynamic 3D environment.

The Fast Marching Square method has been used along the first 2 chapters as a basis for the algorithms built on top of it. First, it has been shown how to make improvements to the control scheme for robot formations to adapt it to its use in 3D environments. The key factors of this control method are its simplicity and flexibility, since it is easy to implement and adapt to different amount of robots or geometrical formations, since the deformation schema is based on basic 3D translations and rotations which are be computed very fast using standard algebraic tools. The introduction of function-based geometry deformation is very powerful since it permits setting complex behaviours to the followers by simply modifying these functions. As an example of this, the use of priorities in the formation is showed. These functions can be modified dynamically, an important property that is worthy to explore in the future.

In general, it is shown that it is possible to decouple a high dimensional path planning problem into two simpler ones: 3D path planning for one robot and formation control and coordination for the others. On top of this idea, next chapter proposes the use of a formation-like modelling of a robotic hand-arm system. To the author's knowledge, an adaptation of mobile robots formation control schemes to hand control was never proposed before. Using this idea, the intrinsic high-dimensional space of the hand-arm system is simplified so that the path planning phase is performed in 3 dimensions, and the control scheme solves the orientation-related dimensions.

Chapter 3 presents a complete framework for object grasping: starting with how to model the environment, followed by grasp synthesis and path planning, and ending with a control scheme for treating a hand-arm system imitating a robot formation, for simplifying and high-dimensional problem.

The use of FM^2 determines the need of an occupancy grid to model the environment, which then influences the choice of the algorithms used throughout the framework. When building an occupancy grid, it is important to pay special attention to the selection of the resolution. A high resolution implies a more accurate model of the environment, but it comes with larger computation needs in: object voxelizing, path planning and collision detection.

In order to test the concept, simulations are carried out with Matlab, showing examples of the use of the robot formation based path planning and control in the case of both, side and top grasps. Then, test in real on-the-table scenarios have been performed. Although simple scenarios are used, it is always challenging performing real world grasping tests. Results of the real tests show that it is possible to perform grasping actions executed under the robot formation concept presented. However, it is obviously not a final and perfect solution, with the majority of failures caused by the errors occurred in the object modelling phase, which are then propagated to grasp synthesis, path planning and execution.

Finally, chapter 4 an in-hand pose estimation algorithm is proposed to detect the movement of the object that occurs from the first contact between the hand and the object, until the moment the grasp becomes stable. In the case further manipulation actions are required, the final grasp state needs to be known. A Bootstrap particle filter is proposed for the estimation of the pose of the object inside the hand. The estimation process is based on the measurements provided by the sensors of the hand: tactile and proprioceptive information. The measurement model proposed for the filter is based on a simulation of the position of 3D models of the hand and the object. Then, a collision detection library is used to compare real measurements with simulation based ones.

The algorithm is tested in real grasping scenarios. Results show that the main movements produced can be detected by the estimation scheme. However, the estimation is done in 6D, but the sensors in the hand can only measure movements in 4D. This causes that the estimation in unmeasured axes to have a larger variance.

5.1 Future Work

Nevertheless, the methods proposed have gaps for improvement, therefore some interesting future works are now introduced.

Since the works presented in the first two chapters are based on the Fast Marching Method, they suffer from the curse of dimensionality. Although its use in this work is constrained to 3 dimensions, it would be not efficient to use it in the case of very big environments or the need of a very high resolution. When using grid based models for the environment, octomaps can be used to reduce the amount of information needed,

however, the Fast Marching Method is not designed to work with multi-resolution maps, so an adaptation of this method would be needed.

Dealing with 3D robot formations, future work is also related to getting the simulations as close as possible to the use of real unmanned aerial vehicles. This requires to include dynamics of the vehicles in use in order to prove that the computed paths are smooth enough. From the formations point of view, it would be interesting to test formations in which, for example, the leader is not always in front of the team. Also, allowing dynamic goals for the formation would be an interesting improvement, taking into account the maneuvers needed when drastic pose changes are required to achieve the new goals.

The experimental evaluation using an on-the-table scenario allows for a proof of the concept of the formation-based control of the hand-arm system, however, in the case of cluttered environments, the use of the 3D path computed with FM^2 might not be safe enough, since the arm configuration is not included in the computations and therefore collisions might appear. The use of geometrically constrained path planning [153] or forcing certain configurations for the elbow joint when solving the inverse kinematics problem could be possible solutions.

The grasping synthesys method built for chapter 2 can be clearly improved. Results suggest it would be useful to treat fingertip and power grasps in a different way, paying special attention to the contact points in fingertip grasping. Similarly, point cloud modelling requires more sophisticated algorithm to treat the data than the one used in this thesis.

From the grasping results, two main improvements are suggested. When visible point clouds are used to model the objects, the grasping selection should be probably focused on the known parts of the point cloud. Otherwise, inference mechanisms to improve the point cloud are advised. Also, in the case of using fingertip grasping, it would be interesting to compute approximate grasp positions and take into account the local shape around them to avoid slippery of the objects.

In-hand pose estimation is mainly based on joint encoders and pressure sensors of the hand. While the former are quite robust and have a repetitive performance, the latter very commonly activate under no contact and their response changes over time under similar contacts [154]. Therefore, a filtering process is needed after data gathering. Obviously, an improvement of pressure/contact sensing capabilities would also be desired. Furthermore, it is obvious that human do not only rely on tactile information, but it is often fused with visual feedback. I believe fusing these two types of information is a key factor for a better performance of an in-hand pose estimator, because in this case, measurement on all the degrees of freedom would be available. Finally, once the pose estimation is improved, it would be interesting to use it to close the loop with the grasping system, so that the resulting hand-object configuration is more stable.

Appendix A

Fast Marching and Path Planning

Appendix A.1: Introduction

The Fast Marching Method (FMM) has been extensively applied since it was firstly proposed in 1995 [155] as a solution to isotropic control problems using first-order semi-Langragian discretisations on Cartesians grids. Their main field of application are robotics [6, 156, 157] and computer vision [158], mainly medical image segmentation [159, 160]. However, it has proven to be useful in many other applications such as tomography [161] or seismology [162].

The first approach was proposed by Tsitsiklis [155], but the most popular solution was given few months later by Sethian [43] using first-order upwind-finite differences in the context of isotropic front propagation. Differences and similarities between both works can be found in [163].

FMM was originally proposed to simulate a wavefront propagation through a regular discretisation of the space. However, many different approaches have been proposed, extending these methods to other discretisations and formulations. For a more detailed history of Fast Marching methods, interested readers are referred to [164].

This Appendix introduces FMM, its formulation and its application to path planning. In particular, next section provides an intuitive explanation to FMM. Then, section A.3 details the formulation and how to solve the Eikonal equation in n-dimensions. Section A.4 details the FMM and, finally, section A.5 outlines how the FMM can be applied to path planning.

Appendix A.2: Introduction to Fast Marching Methods

The FMM can be intuitively understood considering the expansion of a wave. If a stone is thrown into a pond a wavefront is originated, and this wave expands with a circle shape around the point where the stone fell. In this example the fluid is always water, thus the wave expansion velocity is always the same, and thus the wavefront is circular. Instead, if this experiment is repeated mixing water and oil, it would be observed that the wave expands at different speeds in each medium. As a consequence, the wavefront will not be circular anymore. If another point on the fluid is considered (a target point), the wavefront will arrive to that point after a certain time. The path that the wavefront has followed from the origin to the target point will be the shortest path (in terms of time), considering that the traveling speed along the path is the expansion velocity of the wavefront (which differs depending on the fluid). This path can be computed by using gradient descent (following the direction of maximum change) from any given point. The smoother the variations are

in the wave propagation velocities, the smoother the path will be. These concepts are plotted in figure A.1. In few words, FMM computes the arrival times of the wavefront from a starting point to all the points of the reachable space.

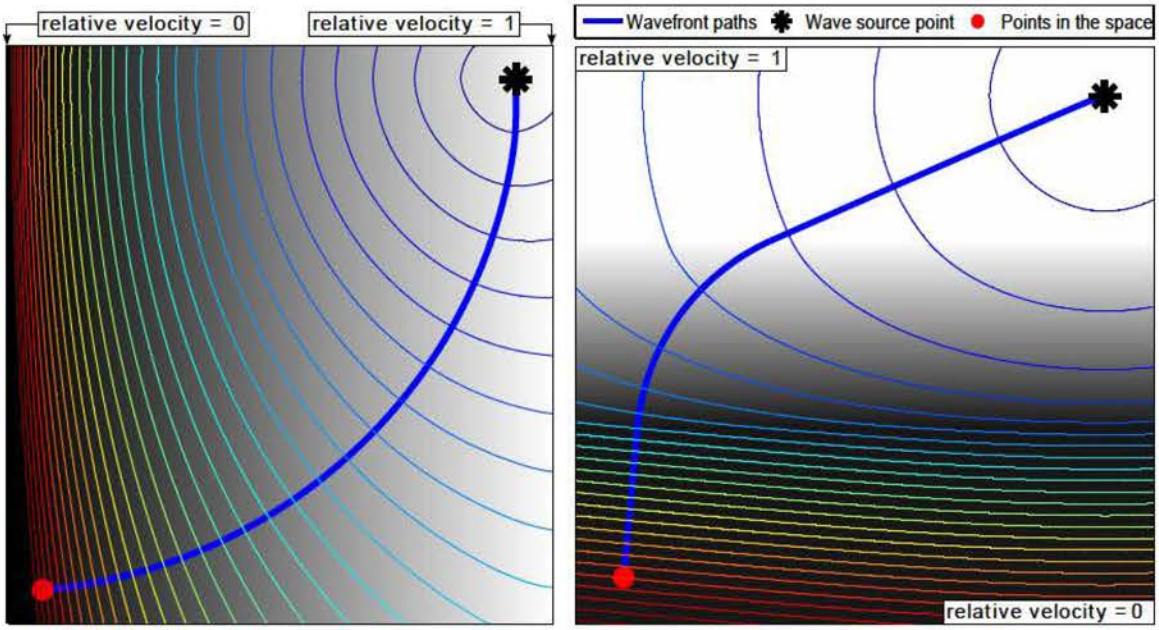
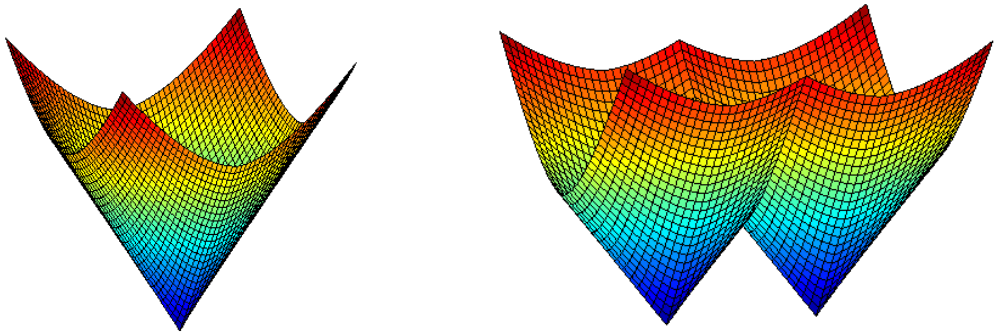


Figure A.1: Examples of a wave propagation through media with different velocities.

The FMM is a particular case of Level Set Methods, initially developed by Osher and Sethian [45]. The FMM assumes that the wave propagation velocity is always non-negative, and thus the wavefront will never contract. Usually, propagation velocities are defined by the environment or by higher-level algorithmic layers so that it is easy to satisfy this assumption. Therefore, if only one wave source is used FMM ensures that the time of arrival map will have only one minimum (both local and global) at the start point. Section A.5 details how FMM deals when obstacles are present in the environment.

Many different wave sources can be set. In that case, there will be as many minima as starting points, each one of them located at these wave sources. An example is shown in figure A.2 with a 3D interpretation of the arrival times represented in the Z axis.

From a high-level perspective, FMM computes the arrival time for one point at every iteration. This is done efficiently by selecting the non-visited points with lower value calculated so far. When evaluating a point, its value is increased taking into account current propagation velocity and only the neighbors with lower times in each dimension. As velocities are always non-negative current value will be at least as high



(a) One wave source (b) Two wave sources

Figure A.2: 3D representation of the FMM output in a 2D grid. The arrival time is shown in the Z axis .

as the neighbor with lower value, and thus no local minima can appear. However, saddle points are possible but they are not a problem as gradient descent (used to compute the paths) will be able to continue without problems. In this case, it means that two solution with the same propagation time are possible and which one is chosen uniquely depends on the gradient descent implementation.

Appendix A.3: Problem Formulation

Formally, FMM is built to solve the nonlinear boundary value problem[1]. That is, given a domain $\mathcal{X}$ and a velocity field function $F : \mathcal{X} \rightarrow \mathbb{R}_+$ which represents the local speed of the motion, drive a system from a starting set $\mathcal{X}_s \subset \mathcal{X}$ to a goal set $\mathcal{X}_{goal} \subset \mathcal{X}$ through the fastest possible path. The Eikonal equation computes the minimum time of arrival function $T(\mathbf{x})$ as follows:

$$\begin{array}{c} \left|\nabla T(\mathbf{x})\right| F(\mathbf{x}) = 1,\ \mathcal{X} \subset \mathbb{R}^N \\ T(\mathbf{x}) = 0, \mathbf{x} \subset \mathcal{X}_s \end{array} \tag{A.1}$$

Once solved, $T(\mathbf{x})$ represents a distance (time of arrival) field containing the time it takes to go from any point $\mathbf{x}$ to the closest point in $\mathcal{X}_s$ following the velocities on $F(\mathbf{x})$.

[1]This problem formulation closely follows [165]

It is assumed, without loss of generality, that the domain is a unit hypercube of N dimensions: $\mathcal{X} = [0,1]^N$. The domain is represented with a rectangular Cartesian grid $\mathcal{X} \subset \mathbb{R}^N$, containing the discretisations of the functions $F(\mathbf{x})$ and $T(\mathbf{x})$, $\mathcal{F}$ and $\mathcal{T}$ respectively. The grid points $\mathbf{x}_{\mathrm{ij}} = (x_i, y_i), \mathbf{x}_{\mathrm{ij}} \in \mathcal{X}$ represents the point $\mathbf{x} = (x, y)$ in the space corresponding to a cell (i, j) of the grid (for the 2D case). For notation simplicity, $T_{\mathrm{ij}} = T(\mathbf{x}_{\mathrm{ij}}) \approx T(\mathbf{x}), T_{\mathrm{ij}} \in \mathcal{T}$, that is, T_{ij} represents an approximation to the real value of the function $T(\mathbf{x})$. Analogously, $F_{\mathrm{ij}} = F(\mathbf{x}_{\mathrm{ij}}) \approx F(\mathbf{x}), F_{\mathrm{ij}} \in \mathcal{F}$. The set of Von-Neumann neighbors (4-connectivity in 2D) of grid point $\mathbf{x}_{\mathrm{ij}}$ is denoted as $\mathcal{N}(\mathbf{x}_{\mathrm{ij}})$. For a general grid of N-dimensions, cells will be referred by their indices (or keys) i as $\mathbf{x}_{\mathrm{i}}$, since a flat representation is more efficient for such datastructure.

A.3.1 n-Dimensional Discrete Eikonal Equation

In this section the most common first-order discretisation of the Eikonal equation is detailed. There exist many other first-order and higher-order approaches on grids, meshes and manifolds [166, 167, 168, 169].

The discrete Eikonal equation is derived in 2D for better understanding. The most common first-order discretisation is given in [45], which uses an upwind-difference scheme to approximate partial derivatives of $T(\mathbf{x})$ ($D_{\mathrm{ij}}^{\pm x}$ represents the one-sided partial difference operator in direction $\pm x$ and Δx and Δy are the grid spacing in the x and y directions):

$$\begin{aligned} T_x(\mathbf{x}) &\approx D_{\mathrm{ij}}^{\pm x} T = \tfrac{T_{i\pm1,j} - T_{\mathrm{ij}}}{\pm \Delta_{\mathrm{x}}} \\ T_y(\mathbf{x}) &\approx D_{\mathrm{ij}}^{\pm y} T = \tfrac{T_{i,j\pm1} - T_{\mathrm{ij}}}{\pm \Delta_{\mathrm{y}}} \end{aligned} \tag{A.2}$$

$$\left\{ \begin{array}{c} \max(D_{\mathrm{ij}}^{-x} T, 0)^2 + \min(D_{\mathrm{ij}}^{+x} T, 0)^2 + \\ \max(D_{\mathrm{ij}}^{-y} T, 0)^2 + \min(D_{\mathrm{ij}}^{+y} T, 0)^2 \end{array} \right\} = \frac{1}{F_{\mathrm{ij}}^2} \tag{A.3}$$

Simpler but less accurate solution to equation A.3 is proposed in [170]:

$$\left\{ \begin{array}{c} \max(D_{\mathrm{ij}}^{-x} T, -D_{\mathrm{ij}}^{+x} T, 0)^2 + \\ \max(D_{\mathrm{ij}}^{-y} T, -D_{\mathrm{ij}}^{+y} T, 0)^2 \end{array} \right\} = \frac{1}{F_{\mathrm{ij}}^2} \tag{A.4}$$

Replacing equation A.2 in equation A.4 and letting

$$\begin{aligned} T &= T_{i,j} \\ T_{\mathrm{x}} &= \min(T_{i-1,j}, T_{i+1,j}) \\ T_{\mathrm{y}} &= \min(T_{i,j-1}, T_{i,j+1}) \end{aligned} \tag{A.5}$$

the Eikonal equation can be rewritten for a discrete 2D space as:

$$\max\left(\frac{T-T_{\mathrm{x}}}{\Delta_{\mathrm{x}}},0\right)^2+\max\left(\frac{T-T_{\mathrm{y}}}{\Delta_{\mathrm{y}}},0\right)^2=\frac{1}{F_{\mathrm{ij}}^2} \tag{A.6}$$

Since it is assumed that the speed of the front is positive ($F>0$), T must be greater than T_{x} and T_{y} whenever the front wave has not already visited the point at coordinates (i,j). Therefore, it is safe to simplify equation A.6 to:

$$\left(\frac{T-T_{\mathrm{x}}}{\Delta_{\mathrm{x}}}\right)^2+\left(\frac{T-T_{\mathrm{y}}}{\Delta_{\mathrm{y}}}\right)^2=\frac{1}{F_{\mathrm{ij}}^2} \tag{A.7}$$

Equation A.7 is a regular quadratic equation of the form $aT^2+bT+c=0$, where:

$$\begin{aligned} a &= \Delta_{\mathrm{x}}^2+\Delta_{\mathrm{y}}^2 \\ b &= -2(\Delta_{\mathrm{y}}^2T_{\mathrm{x}}+\Delta_{\mathrm{x}}^2T_{\mathrm{y}}) \\ c &= \Delta_{\mathrm{y}}^2T_{\mathrm{x}}^2+\Delta_{\mathrm{x}}^2T_{\mathrm{y}}^2-\tfrac{\Delta_{\mathrm{x}}^2\Delta_{\mathrm{y}}^2}{F_{\mathrm{ij}}^2} \end{aligned} \tag{A.8}$$

In order to simplify the notation for the n-dimensional case, let us assume that the grid is composed by (hyper)cubic cells, that is, $\Delta_{\mathrm{x}}=\Delta_{\mathrm{y}}=\Delta_{\mathrm{z}}=\cdots=h$. Let us denote T_{d} as the generalization of T_{x} or T_{y} for dimension d, up to N dimensions. F denotes the propagation velocity for point with coordinates $(i,j,k,\dots)$. Operating and simplifying terms, the discretisation of the Eikonal is a quadratic equation with parameters:

$$\begin{aligned} a &= N \\ b &= -2\sum_{d=1}^{N}T_{\mathrm{d}} \\ c &= (\sum_{d=1}^{N}T_{\mathrm{d}}^2)-\tfrac{h^2}{F^2} \end{aligned} \tag{A.9}$$

A.3.2 Solving the nD discrete Eikonal equation

Wavefront propagation follows causality. That is, in order to reach a point with higher time of arrival, it should firstly travel through neighbors of such point with smaller values. The opposite would imply a jump in time continuity and therefore the solutions would be erroneous.

The proposed Eikonal solution (quadratic equation with parameters of equation A.9) does not guarantee the causality of the resulting distance map, as F and h can

have arbitrary values. Therefore, before accepting a solution as valid its causality has to be checked. For instance, in 2D the Eikonal is solved as:

$$T = \frac{T_\mathrm{x} + Ty}{2} + \frac{1}{2}\sqrt{\frac{2h^2}{F^2} - \left(T_\mathrm{x} - T_\mathrm{y}\right)^2} \tag{A.10}$$

called the *two-sided* update, as both parents T_x and T_y are taken into account. The solution is only accepted if $T \geq \max\left(T_\mathrm{x}, T_\mathrm{y}\right)$. The *upwind condition* [164] shows that:

$$T \geq \max\left(T_\mathrm{x}, T_\mathrm{y}\right) \Longleftrightarrow \left|T_\mathrm{x} - T_\mathrm{y}\right| \leq \frac{h}{F} \tag{A.11}$$

If this condition fails, the *one-sided update* is applied instead:

$$T = \min\left(T_\mathrm{x}, T_\mathrm{y}\right) + \frac{h}{F} \tag{A.12}$$

This is a top-down approach: the parents are iteratively discarded until a causal solution is found. To generalize equation A.11 is complex. Therefore, a bottom-up approach is chosen: equation A.12 is solved and parents are iteratively included until the time of the next parent is higher than the current solution: $T_k > T$. The procedure is detailed in algorithms 2 and 3. The MinTDim() function returns the minimum time of the neighbors in a given dimension (left and right for $dim = 1$, bottom and top for $dim = 2$, etc.). The experiments found this approach more robust for 3 or more dimensions with negligible impact on the computational performance.

Algorithm 2 Solve Eikonal equation

```
1:  procedure SolveEikonal(x_i, T, F)
2:      a ← N
3:      for dim = 1 : N do
4:          min_T ← MinTDim(dim)
5:          if min_T ≠ ∞ and min_T < T_i then
6:              T_values.push(min_T)
7:          else
8:              a ← a − 1
9:      if a = 0 then                ▷ Fast Sweeping Method can cause this situation.
10:         return ∞
11:     T_values ← Sort(T_values)
12:     for dim = 1 : a do
13:         T̃_i ← SolveNDims(x_i, dim, T_values, F)
14:         if dim = a or T̃_i < T_values,dim+1 then
15:             break
16:     return T̃_i
```

Algorithm 3 Solve Eikonal for n dimensions

procedure SolveNDims($\mathbf{x}_\mathrm{i}, dim, T_{\texttt{values}}, \mathcal{F}$)
  if $dim = 1$ **then**
    return $T_{\texttt{values},1} + \frac{h}{F_\mathrm{i}}$
  $sumT \leftarrow \sum_{i=1}^{dim} T_{\texttt{values},i}$
  $sumT^2 \leftarrow \sum_{i=1}^{dim} T^2_{\texttt{values},i}$
  $a \leftarrow dim$
  $b \leftarrow -2sumT$
  $c \leftarrow sumT^2 - \frac{h^2}{F_\mathrm{i}}$
  $q \leftarrow b^2 - 4ac$
  if $q < 0$ **then** ▷ Non-causal solution
    return ∞
  else
    return $\frac{-b+sqrt(q)}{2a}$

Appendix A.4: Fast Marching Method

The Fast Marching Method (FMM) [43] is the most common Eikonal solver. It can be classified as a label-setting, Dijkstra-like algorithm [171]. It uses a first-order upwind-finite difference scheme to simulate an isotropic front propagation. The main difference with Dijkstra's algorithm is the operation carried out on every node. Dijkstra's algorithm is designed to work on graphs. Therefore, the value for every node $\mathbf{x}_\mathrm{i}$ only depends on one parent $\mathbf{x}_\mathrm{j}$, following the Bellman's optimality principle [172]:

$$T_\mathrm{i} = \min_{\mathbf{x}_\mathrm{i} \in \mathcal{N}(\mathbf{x}_\mathrm{i})} (c_{ij} + T_\mathrm{j}) \tag{A.13}$$

In other words, a node $\mathbf{x}_\mathrm{i}$ is connected to the parent $\mathbf{x}_\mathrm{j}$ in its neighborhood $\mathcal{N}(\mathbf{x}_\mathrm{i})$ which minimizes (or maximizes) the function value (in this case T_i) composed by the value of T_j plus the addition of the cost of *traveling* from $\mathbf{x}_\mathrm{j}$ to $\mathbf{x}_\mathrm{i}$, represented as c_{ij}.

The FMM follows Bellman's optimality principle but the value for every node is computed following first-order upwind discretisation of the Eikonal equation, which is described in detail in section A.3. This discretisation takes into account the spatial representation (i.e. a rectangular grid) and the value of all the causal upwind neighbors. Thus, the time-of-arrival field computed by FMM is more accurate than Dijkstra's.

The algorithm labels the cells in three different sets: 1) `Frozen`: those cells which

value is computed and cannot change, 2) `Unknown`: cells with no value assigned, to be evaluated, and 3) `Narrow` band (or just `Narrow`): frontier between `Frozen` and `Unknown` containing those cells with a value assigned that can be improved. These sets are mutually exclusive, that is, a cell cannot belong to more than one of them at the same time. The implementation of the `Narrow` set is a critical aspect of FMM. A detailed discussion on different implementations can be found in [173, 174].

The procedure is detailed in algorithm 4. Initially, all points[2] in the grid belong to the `Unknown` set with infinite arrival time. The initial points (wave sources) are assigned a value 0 and introduced in `Frozen` (lines 2-7). Then, the main FMM loop starts by choosing the element with minimum arrival time from `Narrow` (line 9). All its non-`Frozen` neighbors are evaluated: for each of them the Eikonal is solved and the new arrival time value is kept if it is improved. In case the cell is in `Unknown`, it is transferred to `Narrow` (lines 10-16). Finally, the previously chosen point from `Narrow` is transferred to `Frozen` (lines 17 and 18) and a new iteration starts until the `Narrow` set is empty. The arrival times map $\mathcal{T}$ is returned as the result of the procedure.

Appendix A.5: Path planning with the Fast Marching Method

Analyzing the FMM formulation given in section A.3, it can be seen that there are many common components with the path planning problem formulation detailed by LaValle in [81]. Therefore, it is possible to solve the path planning method with FMM.

The configuration space $\mathcal{X}$ corresponds with the domain $\mathcal{X}$ of the FMM (reason why they are named the same). $\mathcal{X}_{\text{obs}}$ corresponds to the subset of $\mathcal{X}$ and represents those points in the space in which the wave cannot propagate. Although theoretically obstacles and zero-velocity cells are different, in practice they are treated the same as they output the same infinite value as the wave never reach such point. Consequently, $\mathcal{X}_{\text{free}}$ contains the rest of the cells.

For path planning, it is assumed that a $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ occupies at least one cell of the configuration space discretisation. In other words, a well-behaved goal set will be in practice at least of the size of a cell. Analogously, the start set $\mathcal{X}_{\text{s}} \subset \mathcal{X}_{\text{free}}$ is also represented by at least one cell.

In order to compute the path, the wave is propagated from $\mathcal{X}_{\text{s}}$ to $\mathcal{X}_{\text{goal}}$, obtaining a time-of-arrival map $\mathcal{T}$. Applying gradient descent over $\mathcal{T}$ from $\mathcal{X}_{\text{goal}}$, it is satisfied that $\sigma(1) \in \text{cl}(\mathcal{X}_{\text{goal}})$. Gradient descent will compute the path to the unique minimum of $\mathcal{T}$ and therefore $\sigma(0) = \mathcal{X}_{\text{s}}$. In terms of implementation, this will actually return the path inverted, as its waypoints will travel from $\mathcal{X}_{\text{goal}}$ to $\mathcal{X}_{\text{s}}$. Thus the wave is

[2]From now on, point, cell or node will indistinctly used to refer to each element of the grid.

Algorithm 4 Fast Marching Method

procedure FMM($\mathcal{X}, \mathcal{T}, \mathcal{F}, \mathcal{X}_\mathrm{s}$)
Initialization:
  $\texttt{Unknown} \leftarrow \mathcal{X}, \texttt{Narrow} \leftarrow \emptyset, \texttt{Frozen} \leftarrow \emptyset$
  $T_\mathrm{i} \leftarrow \infty \ \forall \mathbf{x}_\mathrm{i} \in \mathcal{X}$
  for $\mathbf{x}_\mathrm{i} \in \mathcal{X}_\mathrm{s}$ **do**
    $T_\mathrm{i} \leftarrow 0$
    $\texttt{Unknown} \leftarrow \texttt{Unknown} \backslash \{\mathbf{x}_\mathrm{i}\}$
    $\texttt{Narrow} \leftarrow \texttt{Narrow} \cup \{\mathbf{x}_\mathrm{i}\}$

Propagation:
  while $\texttt{Narrow} \neq \emptyset$ **do**
    $\mathbf{x}_\mathrm{min} \leftarrow \arg\min_{\mathbf{x}_\mathrm{i} \in \texttt{Narrow}} \{T_\mathrm{i}\}$ ▷ `Narrow` top operation.
    for $\mathbf{x}_\mathrm{i} \in (\mathcal{N}(\mathbf{x}_\mathrm{min}) \cap \mathcal{X} \backslash \texttt{Frozen})$ **do** ▷ For all neighbors not in Frozen.
      $\widetilde{T}_\mathrm{i} \leftarrow \textsc{SolveEikonal}(\mathbf{x}_\mathrm{i}, \mathcal{T}, \mathcal{F})$
      if $\widetilde{T}_\mathrm{i} < T_\mathrm{i}$ **then**
        $T_\mathrm{i} \leftarrow \widetilde{T}_\mathrm{i}$ ▷ `Narrow` increase operation if $\mathbf{x}_\mathrm{i} \in$ `Narrow`.
      if $\mathbf{x}_\mathrm{i} \in$ `Unknown` **then** ▷ `Narrow` push operation.
        $\texttt{Narrow} \leftarrow \texttt{Narrow} \cup \{\mathbf{x}_\mathrm{i}\}$
        $\texttt{Unknown} \leftarrow \texttt{Unknown} \backslash \{\mathbf{x}_\mathrm{i}\}$
    $\texttt{Narrow} \leftarrow \texttt{Narrow} \backslash \{\mathbf{x}_\mathrm{min}\}$ ▷ `Narrow` pop operation: add to Frozen.
    $\texttt{Frozen} \leftarrow \texttt{Frozen} \cup \{\mathbf{x}_\mathrm{min}\}$
  return $\mathcal{T}$

often propagated from $\mathcal{X}_{\text{goal}}$ or the path is inverted. However, this has no effect in the path, as isotropic FMM are being considered.

Regarding optimality, the cost function is defined as $c = \mathcal{T}$. That is, the time of arrival of each cell actually represents its cost from the start point. FMM guarantees that the path returned by gradient descent are optimal, as every cell has the lowest possible value T_{i} assigned, and therefore there is not better alternative to reach that cell.

Concretely, the maximum gradient direction is computed applying the Sobel operator over the grid map.

$$grad_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \star \mathcal{T} \qquad grad_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star \mathcal{T} \tag{A.14}$$

For tracing the path between the initial and the goal points the maximum gradient direction has to be follown starting at the initial point. The path is computed iteratively. $grad_{ix}$ and $grad_{iy}$ are computed at every point p_i. From p_i is computed p_{i+1} (equation A.15) successively until the minimum is reached. The step size ($step$) is user-defined. Thus one advantage of FMM is that the extracted paths have a large number of points, a useful feature when implementing path following on a real robot. As the goal point is located at the global minima it is always reached (whenever there is path).

$$\begin{aligned} mod_i &= \sqrt{grad_{ix}^2 + grad_{iy}^2} \\ alpha_i &= \arctan(\tfrac{grad_{iy}}{grad_{ix}}) \\ p_{(i+1)x} &= p_{ix} + step \cdot \cos(alpha_i) \\ p_{(i+1)y} &= p_{iy} + step \cdot \sin(alpha_i) \end{aligned} \tag{A.15}$$

Figure A.3 shows an example of a path planning problem solved with FMM in 2D. The velocity map $\mathcal{F}$ is a binary map: every cell in $\mathcal{X}_{\text{free}}$ has velocity 1 (white) and $\mathcal{X}_{\text{obs}}$ is represented with cells with zero velocity (black). Usually, obstacles are dilated in order to provide minimum safety guarantees, but this is very application dependent.

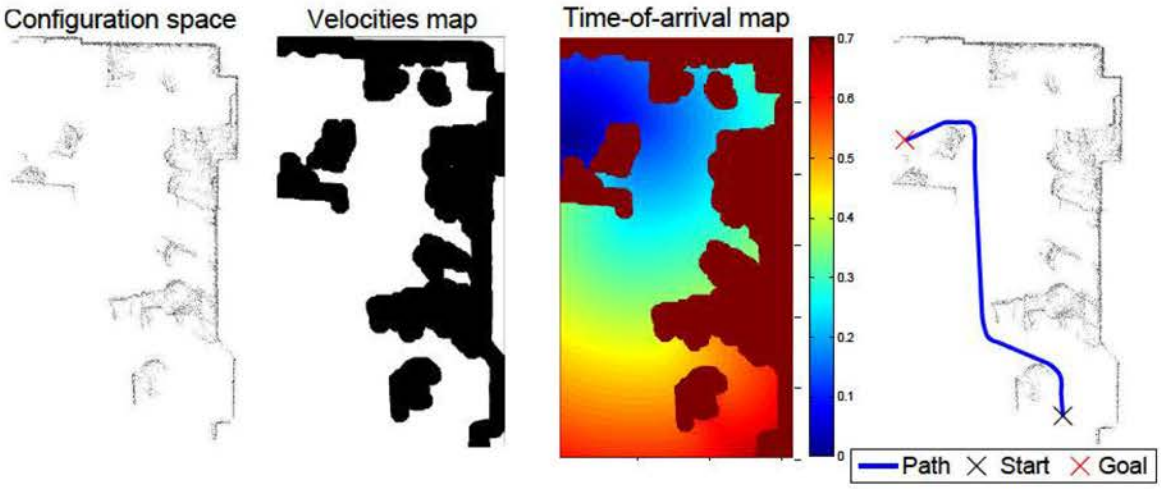


Figure A.3: Example of a path planning problem solved with Fast Marching.

The main drawbacks of Fast Marching-based planning methods are: 1) robot dimensions are not explicitly taken into account, and 2) kinematic constraints are not taken into account. 1) is partially solved in practice by obstacle dilation as shown in figure A.3. Sometimes, if enough computation resources are available, the obstacle set in the grid already takes the shape of the robot into account, commonly by using the maximum radius of the robot projection on the ground or by including a third dimension to the configuration space in order to represent the yaw (heading angle) of the robot. However, 2) is still an open issue. Some approaches to include kinodynamic constraints have been proposed. For instance, [175] uses a two-step approach which first computes a geodesic-based path initialization (Fast Marching can be thought as a geodesic finder) and then an optimization procedure based on Bézier curves is applied to satisfy robot's kinodynamic constraints. Or [176] which samples the space mixing a wavefront-propagation schema and forward-simulation of the system's kinematics and dynamics. However, these approaches represent a mixture of problems and further testing is required before using them in real robotic applications.

Appendix B

Experimental Platforms

The experimental platform ManfredV2, fully developed at the Robotics Lab of the Carlos III University of Madrid, is presented in this appendix. Most of this information and figures have been taken from the work by Álvarez [177].

The mobile robot ManfredV2 is a mobile manipulator whose purpose is to serve as experimental platform for R&D in the mobile robots area.

One of the main objectives of this research is to build an autonomous robot for an indoor office area. In other words, ManfredV2 must be able to navigate autonomously in an environment typically composed of a corridor and offices. For example, one specific task that the robot must perform is to move from one room to another by opening a door.

This robot has been built because it is necessary to have an experimental platform with a robust and reliable hardware that allows researchers to focus on the real problem: the implementation of an artificial intelligence that allows the robot to be autonomous and perform multiple tasks.

The robot design is inspired by planetary rovers and communications satellites. These systems are composed of several subsystems that need to be interconnected to make the whole system work. These subsystems are: onboard computer, power distribution system, sensors, drive system, etc. More instruments to explore the surroundings, such as articulated arms, can also be implemented depending on its application, but it is necessary to distinguish between the mobile platform and the inserted accessories. One important characteristic is that the subsystems are designed as independent units or boxes that are interconnected to each other by an internal wiring.

Summarising, the design of ManfredV2 is based on independent units that are interconnected to each other by using electric and mechanical interfaces. This modular concept facilitates the integration, repair, and future expansion of the robot.

ManfredV2 is presented in figure B.1. It has eight DOFs, divided in a differential-type mobile base with two DOFs and ananthropomorphic light arm with six DOFs. It is able to perform several tasks such as: opening and passing through doors, obstacle avoidance, and picking up and manipulating objects. In order to do that, the robot needs all the basic capabilities to move safely and independently around the environment, motor coordination between the base and manipulator, and sensory coordination to manipulate objects.

As it was previously said in this appendix, any robotic system consists of a set of subsystems that enable (through networking) meeting the objectives for which it was designed. These modules use the environment information to generate data that is used to develop the movement skills in the robot's base and the robotic arm. The main components of the systems that constitute the mobile manipulator are described in the following sections.

Figure B.1: ManfredV2, mobile manipulator with robotic arm.

Appendix B.1: Mechanical Design - Robot Structure

The design of the mobile robot must meet the following specifications: high mobility, mechanical and electrical robustness, high repeatability in its movements, and easy integration and repairing (modular concept).

A brief description of the mechanical design of ManfredV2 and a breakdown of the most important elements are given in this section. The mechanical design of the robot's base is also based on the robustness and reliability that must satisfy the robot when it is performing a task. It is crucial that the robot movement does not cause instability or inaccuracy.

The base has also been designed following a modular philosophy which has two important advantages: it is easy to access to all elements of the mobile robot, and the change of elements due to repairs or improvements is immediate.

The general design also focuses on the improvement of the structure rigidity. The force distribution is more balanced than the distribution of the previous version (ManfredV2). The location of the base elements has been optimized in order to counterbalance when the robotic arm is executing critical tasks, which means that the distribution of the elements in the robot's base gives stability to the mobile robot.

Some mechanical characteristics and their associated advantages are given below. Some of them are compared to the previous version of the mobile robot.

- When the arm is at rest, it does not collide neither interfere with the base. If the system runs out of power, the arm can fall freely without damage to itself or to the base.
- The gravity centre of the base has been moved closer to the ground. This implies an improvement in the stability.
- The main mast has been extended to the bottom plate and more columns have been placed between the plates. These changes give more rigidity to the system.
- It has independent carcasses that are easy to remove and place. It is easier to access to any component of the mobile robot.
- An internal communication system from the mast to the bottom plate has been designed. This system is simple and facilitates the changes or incorporation of new elements.
- All switches, buttons, and safety mushrooms are located in a single panel. This allows an easy and fast access to each element of the control and security systems.
- A second robotic arm that will be added to the robot has been taken into account, trying to make its future implementation as simple as possible.
- A height adjustment system for the drive wheels has been designed. This allows an accurate calibration.

The robot's weight and the weight of each one of its components are shown in table B.1. It is important to remark that most of the weight is concentrated in the bottom part, which benefits the stability.

ManfredV2 is formed by a metal structure that can integrate all the components needed for operation (figure B.2). It can be divided into three parts:

Table B.1: ManfredV2's weight.

Element	Unit weight (kg)	Total weight (kg)
Batteries	15.40	61.60
Aluminum structure	29.00	29.00
Drivers	0.68	5.44
Computer	5.00	5.00
Electronic devices	2.75	2.75
DC-DC Converters	2.00	2.00
Carcasses	2.50	2.50
Caster wheels	0.42	1.26
Drive wheels	7.00	14.00
Wiring	6.00	6.00
Total		129.55

Figure B.2: ManfredV2, lateral view.

- Mobile base:

  The robot's base is composed of two steel platforms with a diameter of 61 cm

Figure B.3: Power supply system.

and a height around 65 cm. It is equipped with wheels that allow movement. The battery system that generates the power to operate autonomously is also stored in the base.

The motion system is included in the base. It has five wheels: three of them are support wheels to improve the stability and facilitate the movement, and the other two are drive wheels with brushless motors and their corresponding servo-amplifiers. The drive wheels generate a differential displacement that allows the robot to turn around its axis.

The power supply system that gives autonomy to the robot consists of batteries that are located in the base. There are four batteries of 12 V connected in series that provide a voltage of 48 V. The selected batteries are Power-Sonic PS-12450 B (figure B.3), which provide an output voltage of 12 V and a capacity of 45 Ah.

In addition, as a security system, the robot has a monitoring system through a PIC16F818 microcontroller that measures the voltage provided by the batteries and the current flowing through them. This system can continuously communicate the power status to the control computer, as well as stopping the motors in a controlled way in case of low voltage or too high current.

- Body:

An structure that forms the robot body and holds multiple components has been mounted on the base. The body contains all the wiring for connecting several subsystems: arm to computer, power from battery to motors, and external sensors. It has also the servo amplifiers associated with the arm.

This structure serves as dock for the robotic arm, the laser sensor, and the computer vision cameras. The onboard computer that is responsible for add intelligence to the robot is also inside this part of the robot. This computer has the PMAC2-PCI card installed, which is a controller card that can control jointly the eight DOFs corresponding to the base and the manipulator arm.

- Robotic arm:

  The manipulator arm LWR-UC3M-1 is an essential element of the robot. It is composed of rigid elements connected by revolution joints. Each joint gives an additional DOF to the robot. The total number of DOFs is six for the arm. It has been designed to provide a remarkable flexibility to perform manipulation tasks (grasping and and movement of objects) by combining the available DOFs.

  The robotic arm that is presented in figure B.4 has been fully developed by the Robotics Lab of the Carlos III University of Madrid. Its main characteristics are:

  1. Kinematic redundancy similar to the human arm.
  2. Weight: 18 kg.
  3. Maximum load capacity: 4.5 kg at the end of the arm.
  4. Load/weight ratio: between $1:3$ and $1:4$.
  5. Range: around 955 mm.

  The developed arm is mounted on the lateral side of the mobile robot in such a way that the computer vision and the laser telemetry systems are not obstructed by the arm. The arm joints are composed of DC brushless motors and Harmonic Drives that reduce the speed and increase the torque.

  Since the installed encoders obtain relative information (they provide information about the motor current position with respect to an initial or home position), an initial *home* function must be executed in order to fix the robotic arm initial position. This facilitates the conversion between relative and absolute positions. This function has been designed using the programming language of the PMAC2-PCI. It establishes that the initial position of the robotic arm is that one in which it is pointing straight to the ground. This position has been chosen because it requires a low energy consumption because most of the engines are not doing any work.

Figure B.4: LWR-UC3M-1(robotic arm).

- End-effector: Gifu Hand III

Gifu Hand III consists of a thumb and four fingers. The thumb and fingers are articulated at each joint and can move independently. The thumb has four joints with four DOFs, and each of the fingers has four joints with 3-DOFs. The fourth joint is linked to the their third one via a planar four-bar linkage mechanism, which generates 1:1 movement. Thus one hand contains a total number of twenty joints, encompassing 16-DOFs. These functions closely approximate to the motion of a human hand.

Figure B.5 shows the workspaces of the fingers and the thumb. One of the most important characteristics of the hand is the high degree of opposability of the thumb. This means that the workspace of the thumb is highly overlapped with the workspaces of the fingers, making it suitable for object manipulation [108].

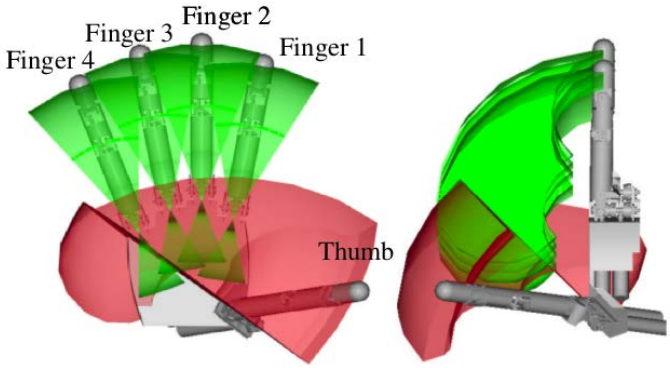


Figure B.5: Virtual workspaces of the fingers (green) and the thumb (red) in frontal and lateral view.

The structure of the hand is made of aluminium and the motors are built-in the mechanical design. The total weight of the hand is $1.40kg$. It also includes relative encoders in each of the joints. The hand is powered by an external power box which also provides current measurements for indirect force control. Both the hand and the control box can be seen in figure B.6.

Appendix B.2: Sensory System

The sensory system can transform the physical variables that characterize the environment into a data set that will be processed by other modules, such as the localization system, the security system, and the motion planner, in order to increase the robot intelligence and be able to execute certain tasks. This information will be provided by the robot's sensory system, which consists of the following elements:

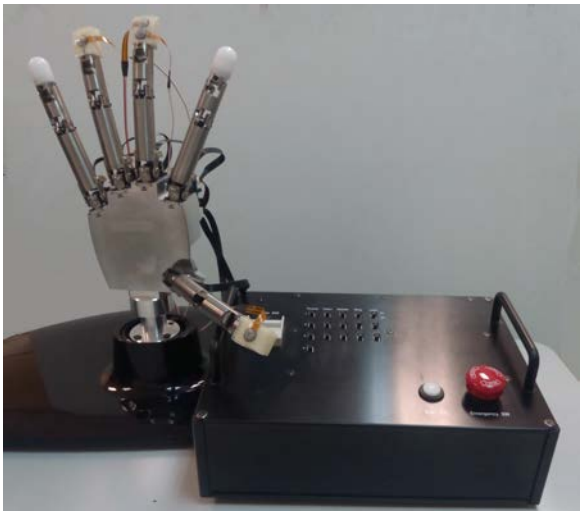


Figure B.6: Gifu Hand III and its control and power supply box.

- Laser telemetry subsystem:

  Its aim is to provide the robot with information about its surrounding environment by measuring the distance to objects. This information is primarily used in navigation and localization in order to model the workspace.

  It is possible to use 2D or 3D data depending on the task characteristics and the complexity and degree of occupancy of the workspace.

  This subsystem is composed of the following laser range finders:

  1. Hokuyo UTM-30LX with 270 opening degrees (figure B.7) located in the rear of the vehicle. It has a detection range that varies from 100 mm to 30 m and a 25 ms period. Its angular resolution is equal to 0.25°. It is connected to the computer through a USB2.0 interface. Its power consumption is 700 mA and 12 V, which makes it suitable for battery-powered systems such as ManfredV2.
  2. SICK PLS with 180 opening degrees (figure B.8). The original measurements are 2D, but we have added a motor that lets it rotate up and down (±45°), being able to obtain 3D measurements (it can also be observed in the figure). The technical characteristics are summarized in table B.2.

     The 2D telemetry (horizontal plane parallel to the ground) can be used during navigation around environments with few obstacles to safe computational time.

     This sensor records 361 measurements in a planar sweep with medium resolution (separation between measurements equal to 0.5°). The SICK

Figure B.7: Hokuyo UTM-30LX (laser range finder).

Figure B.8: SICK PLS (laser range finder).

PLS measurement error is lower than 20 mm. This error is influenced by two parameters: the measuring distance and the angle of the laser beam shot (from 0° to 180°).

- Computer vision subsystem:

  This subsystem helps in the manipulation of objects in 3D environments, which is one of the abilities of ManfredV2. In order to do this, it is necessary to recognise the object to be manipulated, estimate its position and orientation relative to the mobile manipulator, and determine the grasping point. It also facilitates other tasks, such as opening doors, navigation, and localization.

Table B.2: SICK PLS technical characteristics.

Maximum range	80 m
Angular resolution	0.25° - 0.5° - 1° (variable)
Time response	26 ms
Distance resolution	10 mm
Transfer rate	500 kbaud
Power requirements	24 V - 6 A

Figure B.9: Colour cameras. Left: SONY EVI-D100. Right: SONY B/N XC-ES50CE.

The computer vision subsystem is composed of the following elements:

1. Colour camera: SONY EVI-D100 (figure B.9). This camera is employed to recognise objects and estimate their positions relative to the robot. It is located in the front of the mobile robot body.
2. Colour camera: SONY B/N XC-ES50CE (figure B.9). This is a mini-camera that is situated on the wrist of the robotic arm. It is used in manipulation tasks when the extreme of the arm is close to the object to be manipulated and the field of vision of the other camera is obstructed by the arm.
3. Time-of-flight camera (Kinect): the robot also incorporates a camera with time-of-flight technology (figure B.10) that obtains a 3D image composed of an array of distances to different objects and colour information. This information can be fused with the data of the other cameras in order to improve the manipulation capabilities.

Figure B.10: Time-of-flight camera: Kinect.

Figure B.11: JR3 67M25A-U560 (force/torque sensor).

- Force/torque sensor:

  ManfredV2 has a JR3 force/torque sensor (model 67M25A-U560, figure B.11) at the end of the robotic arm. Its purpose is to interact with the environment in manipulation tasks. This sensor is situated between the end of the arm and the clamp or terminal element.

  This device has the following features:

  - Maximum load capacity: 11 kg.
  - Weight: 175 gr.
  - Maximum operating frequency: 8 kHz.

  The JR3 sensor provides force and torque data in three axes that can be used in the force control loop of the mobile manipulator. It is based on a strain gauge system and a Digital Signal Processor (DSP) acquisition system that allow measurements with high bandwidth and signal-noise ratio. The main purpose of this sensor is to perform manipulation tasks based on force or torque control, such as opening doors, pulsation of switches, manipulating objects, etc.

- Motion sensors:

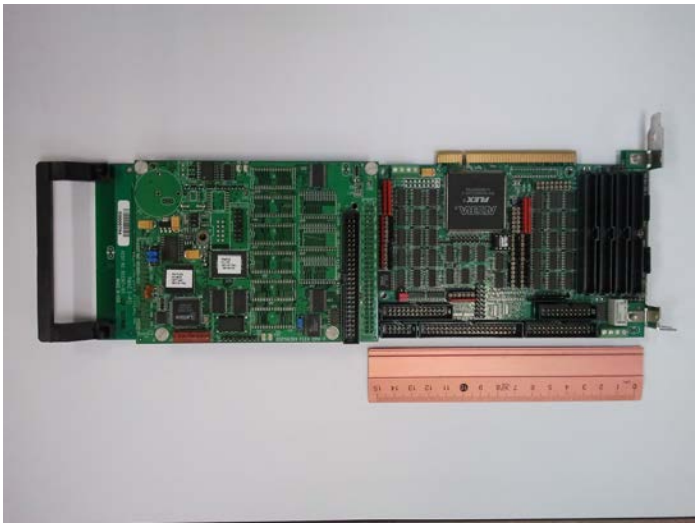


Figure B.12: PMAC2-PCI (controller card).

The main function of these sensors is to obtain information about the robot location and the arm posture. This information is obtained by encoders that are mainly coupled to the rotation axes of the motors. The relative or absolute position of each motor is computed by using this information. The motion sensors are high-resolution optical encoders of the HP company with reference HEDS550.

These motion sensors are complemented by inductive sensors that perform an initial routine that is usually named as *home* in order to establish the absolute position of each joint of the arm. This routine improves the safety and minimizes the power consumption. The inductive sensors have a diameter equal to 3 mm and a detection distance equal to 1 mm. Their basic principle is based on the inductive detection of ferromagnetic materials by flux variation caused by their presence near the sensor's detection area.

Appendix B.3: Control System

ManfredV2 has eight different motors to move its base (2) and its robotic arm (6). It is necessary to have a continuous control of these engines when the robot is navigating or it is moving its arm. This control is carried out by the PMAC2-PCI controller card (figure B.12).

The PMAC2-PCI is a Programmable Multi-Axis Controller card developed by

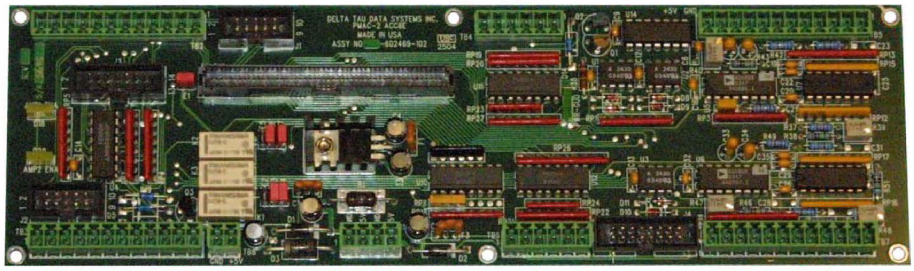


Figure B.13: ACC-8E. Interface between the PMAC2-PCI and the devices.

Delta Tau Data Systems[1]. It is a high performance device that can simultaneously control up to eight axes with high precision. It has a high performance/price ratio, with more than 1000 configuration variables and the high computing capacity of its DSP. The DSP that is incorporated in the PMAC2-PCI is the DSP56002 of 24 bits and operation frequency of 40 MHz.

This card offers multiple ways to control the motors. However, it has not been designed to be connected directly to the devices. There is a set of additional cards that can be used as interfaces . These cards are also offered by Delta Tau Data Systems.

In the case of ManfredV2, the additional card is the ACC-8E (figure B.13). Since each card can interact with two motors, it is necessary to implement four of them. Each ACC-8E card is connected to the PMAC2-PCI through a 100-pin bus that is called JMACH. Each ACC-8E card has four 18-bit Digital-to-Analog Converters (DAC) that command two analog input drivers and must be fed with 15 V. It has also two inputs to read the encoders and five inputs per axis that capture different types of events: error signal, *home* signal (starting position), motor limits (two signals), and user-defined signal (external events for a specific application).

The configuration of the PMAC2-PCI is a very laborious and tough task. There are two available manuals, the "Software reference manual" and the "PMAC2 user manual", together with a program provided by the manufacturer, the "PEWIN32 PRO", which runs under Windows. This software offers a set of tools to modify all the configuration parameters of the PMAC2-PCI. Some of these tools are:

1. Terminal: it sends commands to the card in ASCII coding.

2. Watch window: it is a window where it is possible to view the variable values in real time.

[1] `http://www.deltatau.com`

3. Tunning Pro: it configures the PMAC2-PCI parameters, such as: PID controllers, filters, DAC calibration, and so on.

4. Position window: it displays the position of the motors, in counts of encoder, and also their speed and tracking errors.

Finally, the controller card allows different types of programs:

- Motion programs: the most common task of the controller card is to move the motors according to a particular sequence of commands. These programs are executed line by line by the controller card. They are called by a specific command and, after that, they are executed once. It is possible to make a call to another program or terminal commands. The controller card can store and execute up to 256 motion programs.

- Programmable Logic Controller (PLC): the PLC programs exist because there are some programs that must be executed continuously. For example, there is a PLC program that computes the robot's position given the encoders information. These programs are written in the same way that the motion programs, except that they are defined as PLC in their title. They are called and executed in each cycle of the controller card.

- Motion commands: it is possible to send motion commands to the PMAC2-PCI through the terminal. They are simple commands that allow the motion of each motor. These commands were initially implemented to test the controller card, but they can perform simple movements in a motion program.

Appendix B.4: Software

B.4.1 MATLAB

MATLAB (abbreviation of MATrix LABoratory)[2] is a numerical computing environment developed by MathWorks. It is oriented to projects that imply high computation resources and graphical display. It allows multiple actions, such as: manipulation of matrix and vectors, handling and plotting of functions and data, implementation of algorithms, creation of graphical interfaces, and interfacing with programs in other languages (C, C++, Java, and Fortran).

One additional advantage of this tool is that it is very easy to learn, not being necessary to study a new language because the solutions are expressed by an easy syntax (similar to C).

[2]More information can be found in `http://www.mathworks.es/products/matlab/`.

MATLAB includes a wide range of pre-built functions called "toolboxes". These toolboxes perform multiple operations of multiple areas of engineering and simulation, such as: signal processing, control, statistics, financial analysis, symbolic mathematics, neural networks, fuzzy logic, system identification, dynamic systems simulation, and so on. An additional package called "Simulink" offers a graphical interface for these toolboxes. It allows the simulation of dynamic models.

This tool is widespread in engineering, science, and economics. It has been reported that it had around one million users in 2004. It is also widely used in academic and research institutions.

All these features make MATLAB a suitable tool to be used for our purposes. All the algorithms developed in this work have been implemented in MATLAB.

B.4.2 ROS

ROS (Robot Operating System)[3] is an open source operating system for robotic platforms formerly developed by Willow Garage. Nowadays it is maintained and improved by the Open Source Robotics Foundation. As it is said in its website, 'it provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license (Berkeley Software Distribution, family of permissive free software licenses)'.

ROS is based on a set of processes or nodes that are individually executed and linked by a communication infrastructure provided by ROS. This communication can be synchronous (client-server) or asynchronous (continuous data sending). The different data can be grouped into packages that are shared allowing a distributed collaboration.

The most remarkable characteristics are the following: light and easy to integrate with other systems(outside modules such as: OpenRAVE, Orocos, Gazebo and Player have been integrated in the ROS framework), programming language independent (it can be implemented in the most common languages, such as C++ and Python), easy error correction (because it has a testing unit), and appropriate in big systems with multiple modules.

It currently only works with Unix-based platforms. It has been extensively tested on Ubuntu (operating system of ManfredV2). Most of the software in ManfredV2 has been implemented using the ROS framework. All modules developed for the robot must follow its guidelines.

[3]More information can be found in `http://www.ros.org/wiki/`.

Appendix B.5: ReFlex TakkTile Hand

Since the work presented in chapter 4 has been developed in colaboration with the Insitute of Robotics and Mechatronich of the German Aerospace Centre (DLR), a different robotic platform, ReFlex TakkTile Hand, has been used. Next, a detailed description of the hardware is presented:

B.5.1 Mechanical System

The ReFlex hand consists of three modular finger assemblies and a central chassis assembly that includes all actuators, the preshape transmission, the palm electronics, and the interface electronics. The fingers are mechanically and electrically interchangeable so they can be easily replaced or upgraded. On the left of figure B.14, the hand is lying on its palm without the electronics cover. On the right, the outer appearance of the Reflex TakkTile Hand.

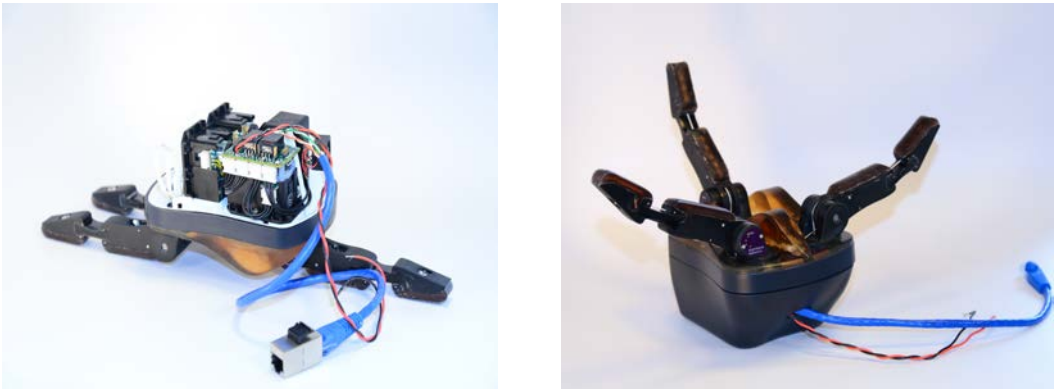


Figure B.14: Electromechanical system of Reflex TakkTile Hand.

There are two transmission systems in the ReFlex hand, the preshape and spool transmissions. For the preshape joints, a flat enclosed gear train is driven to spin two fingers at the same time.

The fingers are driven by a tendon and spool system. Each finger has a Spectra tendon running down the back, which is wrapped and secured around a pulley on the motor. This pulley can be easily be removed and reattached when installing or rethreading tendons. The tendon cable sometimes stretches over time but can be easily recalibrated. The tendon passes directly from the back of finger down through the knuckle to where it meets the spool, cutting down on both friction and tendon wear over time.

The ReFlex hand uses the Robotis Dynamixel MX-28T high-performance servomotor for all 4 actuators. In addition to providing servo capabilities, Dynamixels will report back speed, position, torque, temperature, and voltage data, which can be accessed through the ReFlex interface.

In order to allow better cooling of the dynamixel motors, we've removed the back plate. If you remove them from the chassis, be careful to avoid pulling the motor apart and scattering the gear train.

B.5.2 Degrees of freedom

Each finger is controlled by a single actuator that drives a tendon spanning both the proximal and distal joint. This allows the fingers to passively shape themselves to the shape of the object. Note that the distal joint does not actuate until the proximal phalange encounters an obstacle. A fourth actuator controls a coupled preshape degree of freedom, for a total of seven joints including the fingers.

The proximal revolute joint connects the proximal link to the knuckles, and rotates around the knuckle axle. The range goes from 0 radians (finger flat and fully open) to nearly π radians, when fully closed and resting against the palm. The angle of the proximal joint is directly measured by a 14-bit magnetic hall effect encoder in the knuckle. The distal flexure joint connects the distal link to the proximal link, and flexes around the cast urethane joint. The range goes from 0 radians (finger flat and fully open) to nearly $7\pi/8$ radians when fully closed and resting against the proximal link. The distal joint cannot be commanded directly because it is coupled to the proximal joint. The angle of the distal joint is calculated from the difference between the tendon spool encoder and the proximal joint encoder, and is less accurate than the proximal joint measurement because of that. The limits of the proximal and distal joints of the hand are represented in figure B.15.

The preshape joint changes the angle of the two fingers that are located in the same side of the palmand the are directly coupled together. The fingers can be closed (0 radians, aligned with the opposite finger) or opened ($\pi/2$ radians, perpendicular to the opposite finger) according to the type of object to be gripped. Both configurations can be seen in figure B.16.

Figure B.15: Angular limits of the joints of the fingers of Reflex TakkTile Hand.

Figure B.16: Angular limits of the preshape joint of Reflex TakkTile Hand.

B.5.3 Sensors

The ReFlex hand has thirty-eight MEMS barometer pressure sensors built into it, which are both sensitive and robust. Each finger has nine sensors, five in the proximal link and four in the distal link, including 1 in the fingertip which points along the length of the finger, as can be appreciated in figure B.17.

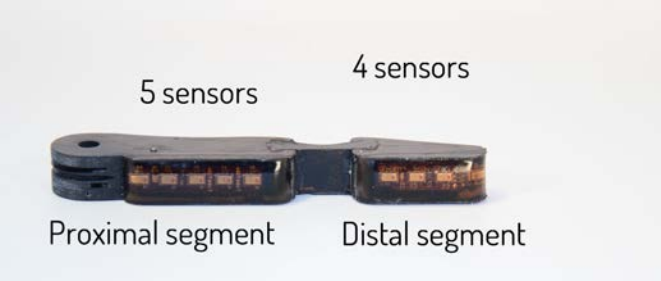


Figure B.17: Pressure sensors configuration along a finger.

Bibliography

[1] S. Kadhim, D. Blanco and L. Moreno, "MANFRED: Robot antropomórfico de servicio fiable y seguro para operar en entornos humanos," *Revista Iberoamericana de Ingeniería Mecánica*, vol. 9, no. 3, pp. 33–48, 2005.

[2] T. Mouri and H. Kawasaki, "A Novel Anthropomorphic Robot Hand and its Master Slave System," *InTech*, 2007.

[3] F. Martín, S. Garrido, D. Blanco and L. Moreno, "High-Accuracy Global Localization Filter for Three-Dimensional Environments," *Robotica*, vol. 30, no. 3, pp. 363–378, 2012.

[4] F. Martín, L. Moreno, S. Garrido and D. Blanco, "Kullback-Leibler Divergence-Based Differential Evolution Markov Chain Filter for Global Localization of Mobile Robots," *Sensors*, vol. 15, pp. 23431–23458, 2015.

[5] A. Valero-Gómez, J. Gómez, S. Garrido and L. Moreno, "The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories," *IEEE Robotics and Automation Magazine*, vol. 20, no. 4, pp. 111–120, 2013.

[6] J. V. Gómez, "Advanced Applications of the Fast Marching Square Planning Method," Master's thesis, Carlos III University, 2012.

[7] S. Garrido, L. Moreno, M. Abderrahim and D. Blanco, "FM^2: A Real-time Sensor-based Feedback Controller for Mobile Robots," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 3169–3192, 2009.

[8] M. Muñoz, S. Garrido, D. Blanco and L. Moreno, "Sensor-based global planning for mobile robot navigation," *Robotica*, vol. 25, no. 2, pp. 189–199, 2007.

[9] N. Burrus, J. Bueno, M. Abderrahim and L. Moreno, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2010.

[10] J. Bueno, M. González-Fierro, L. Moreno and C. Balaguer, "Distinguishing between Similar Objects based on Geometrical Features in 3D Perception," 2013.

[11] C. A. Arismendi, D. Álvarez, S. Garrido and L. Moreno, "Adaptive evolving strategy for dextrous robotic manipulation," *Evolving Systems*, vol. 2, no. 1, pp. 65–72, 2013.

[12] J. V. Gómez, D. Álvarez, S. Garrido and L. Moreno, "Fast Marching-based globally stable motion learning," *Soft Computing*, pp. 1 – 14, 2015.

[13] K. Pauwels and D. Kragic, "SimTrack: A Simulation-based Framework for Scalable Real-time Object Pose Detection and Tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1300–1307, 2015.

[14] E. Marchand, F. Spindler and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, pp. 40–52, December 2005.

[15] J. Rosell, R. Suárez, C. Rosales and A. Pérez, "Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures," *Autonomous Robots*, vol. 31, no. 1, pp. 87–102, 2011.

[16] S. Garrido, L. Moreno, J. V. Gómez and P. Lima, "General Path Planning Methodology for Leader-Followers based Robot Formations," *International Journal of Advanced Robotic Systems*, vol. 10, no. 64, pp. 1–10, 2012.

[17] J. V. Gómez, A. Lumbier, S. Garrido and L. Moreno, "Planning Robot Formations with Fast Marching Square including Uncertainty Conditions," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 137–152, 2012.

[18] M. Martin, P. Klupar, S. Kilberg and J. Winter, "TechSat 21 and Revolutionizing Space Missions Using Microsatellites," in *Proceedings of the AIAA/USU Conference on Small Satellites*, 2001.

[19] A. Dewan, A. Mahendran, N. Soni and K. Krishna, "Heterogeneous UGV-MAV exploration using integer programming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4144–4149, 2013.

[20] S. Hauert, J. Zufferey and D. Floreano, "Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots," in *IEEE Congress on Evolutionary Computation*, pp. 55–61, 2009.

[21] J. Acevedo, B. Arrue, I. Maza and A. Ollero, "Cooperative Large Area Surveillance with a Team of Aerial Mobile Robots for Long Endurance Missions," *Journal of Intelligent and Robotic System*, vol. 70, no. 1–4, pp. 329–345, 2013.

[22] M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, W. J. and I. Kaminer, "Planning for Opportunistic Surveillance with Multiple Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5750–5757, 2013.

[23] S. Bouabdallah, *Design and Control of Quadrotors with Applicationto Autonomous Flying*. PhD thesis, Feb 2007.

[24] S. Hrabar, "Reactive Obstacle Avoidance for Rotorcraft UAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4967–4974, 2011.

[25] S. Shen, N. Michael and V. Kumar, "3D Estimation and Control for Autonomous Flight with Constrained Computation," in *Proceedings of the IEEE Conference on Robotics and Automation*, 2011.

[26] T. Hino, "Simple Formation Control Scheme Tolerant to Communication Failures for Small Unmanned Air Vehicles," in *Congress of the Aeronautical Sciences*, no. 6.4.4, 2010.

[27] T. Balch and R. C. Arkin, "Behavior-based Formation Control for Multirobot Systems," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 12, pp. 926–939, 1998.

[28] D. Naffin and G. Sukhatme, "Negotiated Formations," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 181–190, 2004.

[29] J. Fredslund and M. Matari, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 837–846, 2002.

[30] M. Lemay, F. Michaud, D. Létourneau and J. Valin, "Autonomous Initialization of Robot Formations," in *Proceedings of the IEEE Conference on Robotics and Automation*, vol. 3, pp. 3018–3023, 2004.

[31] P. Ogren, M. Egerstedt and X. Hu, "A control Lyapunov function approach to multiagent coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 847–851, 2002.

[32] M. Zhang, Y. Shen, Q. Wang and Y. Wang, "Dynamic artificial potential field based multi-robot formation control," in *IEEE Instrumentation and Measurement Technology Conference*, pp. 1530–1534, 2004.

[33] Z. Cao, L. Xie, B. Zhang, S. Wang and M. Tan, "Formation constrained multi-robot system in unknown environments," in *Proceedings of the IEEE Conference on Robotics and Automation*, vol. 1, pp. 735–740, 2003.

[34] K. Tan and M. Lewis, "Virtual Structures for High-Precision Cooperative Mobile Robotic Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 132–139, 1996.

[35] W. Ren and R. Beard, "Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach," *AIAA Journal of Guidance, Control and Dynamics*, vol. 1, no. 1, pp. 73–82, 2004.

[36] A. Ahmad, T. Nascimento, A. Conceiçao, A. Moreira and P. Lima, "Perception-Driven Multi-Robot Formation Control," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1851–1856, 2013.

[37] K. Kanjanawanishkul and A. Zell, "A model-predictive approach to formation control of omnidirectional mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2771–2776, 2008.

[38] D. Álvarez, J. V. Gómez, S. Garrido and L. Moreno, "3D Robot Formations Planning with Fast Marching Square," in *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2014.

[39] W. Yu, G. Chen and M. Cao, "Distributed leader follower flocking control for multi-agent dynamical systems with time-varying velocities," *Systems and Control Letters*, vol. 59, no. 9, pp. 543–552, 2010.

[40] S. Garrido, L. Moreno and P. Lima, "Robot Formation Motion Planning using Fast Marching," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 675–683, 2011.

[41] D. Álvarez, A. Lumbier, J. V. Gómez, S. Garrido and L. Moreno, "Precision Grasp Planning with Gifu Hand III based on Fast Marching Square," in *IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2013.

[42] J. V. Gómez, S. Garrido and L. Moreno, "Adaptive Robot Formations using Fast Marching Square working under Uncertainty Conditions," in *IEEE Workshop on Advanced Robotics and its Social Impacts*, pp. 68–71, 2012.

[43] J. A. Sethian, "A Fast Marching Level Set Method for Monotonically Advancing Fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[44] R. P. Feynman, "Space-Time Approach to Non-Relativistic Quantum Mechanics," *Reviews of Modern Physics*, vol. 20, no. 2, pp. 367–387, 1948.

[45] S. Osher and J. A. Sethian, "Fronts Propagating with Curvature Dependent Speed: Algorithms based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[46] L. Yatziv, A. Bartesaghi and G. Sapiro, "O(N) Implementation of the Fast Marching Algorithm," *Journal of Computational Physics*, vol. 212, pp. 393–399, 2005.

[47] F. Frenet, "Sur les Courbes à Double Courbure," *Journal de Mathématiques Pures et Appliquées*, vol. 1, no. 17, pp. 437–447, 1852.

[48] J. A. Serret, "Sur Quelques Formules Relatives à la Théorie des Courbes à Double Courbure," *Journal de Mathématiques Pures et Appliquées*, vol. 1, no. 16, pp. 193–207, 1851.

[49] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[50] P. Felzenszwalb, R. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[51] R. Volpe, P. Khosla, D. McAllester and D. Ramanan, "Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments," *IEEE Transactions on Transactions on Systems, Man, and Cybernetics*, vol. 20, pp. 1423–1436, 1990.

[52] S. El-Khoury and A. Sahbani, "A new strategy combining empirical and analytical approaches for grasping unknown 3d objects," *Journal of Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 497–507, 2010.

[53] S. C. Jacobsen, J. E. Wood, D. F. Knutti and K. B. Biggers, "The UTAH/M.I.T. dextrous hand: work in progress," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 21–50, 1984.

[54] C. S. Lovchik and M. A. Diftler, "The robonaut hand: a dexterous robot hand for space," in *Proceedings of the IEEE Conference on Robotics and Automation*, vol. 2, pp.907–912, 1999.

[55] H. Kawasaki, T. Komatsu and K. Uchiyama, "Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 3, pp. 296–303, 2002.

[56] Shadow Robot Company, "Design of a dextrous hand for advanced clawar applications," in *Climbing and walking robots and the supporting technologies for mobile machines*, pp.691–698, 2003.

[57] J. Butterfass, M. Fischer, M. Grebenstein, S. Haidacher and G. Hirzinger, "Design and experiences with DLR hand II," in *Proceedings of the World Automation Congress*, vol. 15, pp.105–110, 2004.

[58] R. Suárez and P. Grosch, "Mechanical hand MA-I as experimental system for grasping and manipulation," in *Video Proceedings of the IEEE Conference on Robotics and Automation*, 2005.

[59] L. Biagiotti, F. Lotti, C. Melchiorri and G. Vassura, *How far is the human hand? A review on anthropomorphic robotic end-effectors*. Technical Report, University of Bologna, 2004.

[60] A. Bicchi, "Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

[61] M. T. Ciocarlie and K. A. Peter, "Hand Posture Subspaces for Dexterous Robotic Grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[62] J. Bohg, A. Morales, T. Asfour and D. Kragic, "Data-Driven Grasp Synthesis: A Survey," *IEEE Transactions on Robotics and Automation*, vol. 30, no. 2, pp. 289–309, 2014.

[63] A. Sahbani, S. El-Khoury and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[64] K. Shimoga, "Hand Robot Grasp Synthesis Algorithms: A Survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[65] R. N. Murray, Z. Li and S. Sastry, *A Mathematical Introduction to Robotics Manipulation*. CRC Press, 1994.

[66] A. Bicchi and V. Kumar, "Robotic grasping and contact," in *Proceedings of the IEEE Conference on Robotics and Automation*, invited paper, 2000.

[67] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proceedings of the IEEE Conference on Robotics and Automation*, vol. 3, pp. 2290–2295, 1992.

[68] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[69] R. Diankov and J. Kuffner, *Openrave: A planning architecture for autonomous robotics*. Technical Report, Robotics Institute, Pittsburgh, 2004.

[70] R. Diankov, *Automated construction of robotic manipulation programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, 2010.

[71] R. Balasubramanian, L. Xu, P.D. Brook, J. R. Smith and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics and Automation*, vol. 28, no. 4, pp. 899–910, 2012.

[72] J. Weisz and P. K. Allen, "Pose Error Robust Grasping from Contact Wrench Space Metrics," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 557–562, 2012.

[73] S. Ekvall and D. Kragic, "Learning and Evaluation of the Approach Vector for Automatic Grasp Generation and Planning," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 4715–4720, 2007.

[74] A. Morales, T. Asfour, P. Azad, S. Knoop and R. Dillmann, "Integrated Grasp Planning and Visual Object Localization For a Humanoid Robot with Five-Fingered Hands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5663–5668, 2006.

[75] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Annual Symposium on Foundations of Computer Science*, pp. 421–427, 1979.

[76] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami and J. Kuffner, "Grasp planning in complex scenes," in *Proceedings of the IEEE Conference on Humanoid Robots*, pp. 42–48, 2007.

[77] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner and M. VandeWeghe, "Herb: A home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2010.

[78] E. Drumwright and V. Ng-Thow-Hing, "Toward interactive reaching in static environments for humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 846–851, 2006.

[79] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2464–2470, 2009.

[80] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 995–1001, 2000.

[81] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K., Cambridge University Press, Cambridge, 2006.

[82] R. Krug, T. Stoyanov, M. Bonilla, V. Tincani, N. Vaskevicius, G. Fantoni, A. Birk, A. J. Lilienthal and A. Bicchi, "Velvet fingers: Grasp planning and execution for an underactuated gripper with active surfaces," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3669–3675, 2014.

[83] T. Stoyanov, N. Vaskevicius, C. Müller, et al, "No more heavy lifting: Robotic solutions to the container unloading problem," *IEEE Robotics and Automation Magazine*, 2016, to appear. Downloaded from: *www.aass.oru.se/Research/mro/publications/RobLog.pdf* .

[84] M. Santello, M. Flanders and J. F. Soechting, "Postural hand synergies for tool use," *Journal of Neuroscience*, vol. 18, no. 23, pp. 10150–10115, 1998.

[85] C. R. Mason, J. E. Gomez and T. J. Ebner, "Hand Synergies During Reach-to-Grasp," *Journal of Neurophysiology*, vol. 86, no. 6, pp. 2896–2910, 2001.

[86] J. Rosell, R. Suárez, A. Pérez and C. Rosales, "Including virtual constraints in motion planning for anthropomorphic hands," in *Proceedings of the IEEE International Symposium on Assembly and Manufacturing*, pp. 1–6, 2011.

[87] J. P. Saut and A. Sahbani and V. Perdereau, "A Global Approach for Dexterous Manipulation Planning Using Paths in n-fingers Grasp Subspace," in *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, 2006.

[88] N. I. Bernstein, *The Coordination and Regulation of Movements*. Pergamon Press, Oxford, 1967.

[89] M. L. Latash, *On the evolution of the notion of synergy*. In Motor Control, G. Gantchev et al. (eds.). Academic Publishing House, Sofia, 1999.

[90] A. D'Avella, P. Saltiel and E. Bizzi, "Combinations of muscle synergies in the construction of a natural motor behavior," *Nature Neuroscience*, vol. 6, pp. 300–308, 2003.

[91] Y.P. Ivanenko, et al., "Temporal components of the motor patterns expressed by the human spinal cord reflect foot kinematics," *Journal of Neurophysiology*, vol. 90, pp. 3555–3565, 2003.

[92] E. Todorov and Z. Ghahramani, "Analysis of the synergies underlying complex hand manipulation," in *Proceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4637–4640, 2004.

[93] R. Vinjamuri, M. Sun, C. C. Chang, H. N. Lee, R. J. Sclabassi and Z. H. Mao, "Temporal Postural Synergies of the Hand in Rapid Grasping Tasks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, pp. 986–994, 2010.

[94] C. Y. Brown and H. H. Asada, "Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2877–2882, 2007.

[95] M. Ciocarlie, C. Goldfeder and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3270–3275, 2007.

[96] T. Wimboeck, B. Jan and G. Hirzinger, "Synergy-level impedance control for a multifingered hand," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 973–979, 2011.

[97] D. Prattichizzo, M. Malvezzi and A. Bicchi, "On motion and force controllability of grasping hands with postural synergies," in *Proceedings of Robotics: Science and Systems*, 2010.

[98] G. Gioioso, G. Salvietti, M. Malvezzi and P. Prattichizzo, "Mapping synergies from human to robotic hands with dissimilar kinematics: an object based approach," in *Workshop on Manipulation Under Uncertainty, IEEE Conference on Robotics and Automation*, pp. 3669–3675, 2011.

[99] J. Steffen, R. Haschke and H. Ritter, "Towards dextrous manipulation using manipulation manifolds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2877–2882, 2008.

[100] D. Zarubin, F. T. Pokorny, D. Song, M. Toussaint and D. Kragic, "Topological Synergies for Grasp Transfer," in *Workshop on on Hand synergies, IEEE Conference on Robotics and Automation*, pp. 3669–3675, 2013.

[101] J. Romero, T. Feix, H. Kjellstrom and D. Kragic, "Spatio-temporal modelling of grasping actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2103–2108, 2010.

[102] A. Tsoli and O. Jenkins, "2d subspaces for user-driven robot grasping," in *Workshop on Robot Manipulation: Sensing and Adapting to the Real World, Robotics: Science and Systems*, 2007.

[103] N. Vahrenkamp, T. Asfour and R. Dillmann, "Simultaneous Grasp and Motion Planning: Humanoid Robot ARMAR-III," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 43–57, 2012.

[104] K. Harada, K. Kaneko and F. Kanehiro, "Fast grasp planning for hand/arm systems based on convex model," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1162–1168, 2008.

[105] S. Patil and B. Ravi, "Voxel-based representation, display and thickness analysis of intricate shapes," in Ninth International Conference on Computer Aided Design and Computer Graphics, pp.1–6, 2005.

[106] L. Desanghere and J. J. Marotta, "The influence of object shape and center of mass on grasp and gaze," *Journal of Frontiers in Physiology*, vol. 6, 2015.

[107] K. Price, R.M. Storn and J.A. Lampinen *Differential evolution: a practical approach to global optimization*. Natural computing series. Springer-Verlag Berlin, 2005.

[108] T. Mouri, H. Kawasaki, K. Yoshikawa, J. Takai and S. Ito, "Anthropomorphic Robot Hand: Gifu Hand III," in *International Conference on Control, Automation and Systems*, pp. 296–303, 2002.

[109] H. Kawasaki, T. Komatsu, M. Suda and K. Uchiyama, "Development of an Anthropomorphic Robot Hand Driven by Built-in Servo-motors," in *Proceedings of the International Conference on Advanced Manufacturing*, pp. 215–220, 1998.

[110] N. Hendrich and A. Bernardino, "Dexterous Postural Synergies from Teleoperation of the Shadow Robot Hand," in *Workshop on on Hand synergies, IEEE Conference on Robotics and Automation*, 2013.

[111] Z. Xue, P. Woerner, J. M. Zoellner and R. Dillmann, "Efficient grasp planning using continuous collision detection," in *International Conference on Mechatronics and Automation*, pp. 2752–2758, 2009.

[112] Y. Paulignan, C. MacKenzie, R. Marteniuk and M. Jeannerod, "The coupling of arm and finger movements during prehension," *Journal of Experimental Brain Research*, vol. 79, no. 2, pp. 431–435, 1990.

[113] M. Jeannerod, "The Timing of Natural Prehension Movements," *Journal of Motor Behavior*, vol. 16, no. 3, pp. 235–254, 1984.

[114] M. Jeannerod and B. Biguer, "Visuomotor mechanisms in reaching within extrapersonal space," *Journal of Advances in the analysis of visual behaviour*, pp. 387–409, Bonton: MIT Press, 1984.

[115] K. Shoemake, "Animating rotation with quaternion curves," *Newsletter of ACM SIGGRAPH Computer Graphics*, vol. 19, is. 3, pp. 245–254, 1985.

[116] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer-Verlag London Limited, 2005.

[117] Z. Macura, A. Cangelosi, R. Ellis, D. Bugmann, M. H. Fischer and A. Myachyko, "A Cognitive Robotic Model of Grasping," in *Proceedings of the International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, pp. 89–96, 2009. Editors: L. Cañamero, P.Y. Oudeyer and C. Balkenius.

[118] J. C. Rothwell, M. M. Traub, B. L. Day, J. A. Obeso, P. K. Thomas and C. D. Marsden,"Manual motor performance in a deafferented man," *Brain*, vol. 105, pp. 515–542, 1982.

[119] J. Bimbo, L. D. Seneviratne, K. Althoefer, H. Liu and H. Liu, "Combining touch and vision for the estimation of an object's pose during manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4021–4026, 2013.

[120] K. Honda, T. Hasegawa, T. Kiriki and T. Matsuoka, "Real-time pose estimation of an object manipulated by multi-fingered hand using 3D stereo vision and tactile sensing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1814–1819, 1998.

[121] S. Haidacher and G. Hirzinger, "Estimating Finger Contact Location and Object Pose from Contact Measurements in 3-D Grasping," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1805–1810, 2003.

[122] M. Chalon, J. Reinecke and M. Pfanne, "Online in-hand object localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2977–2984, 2013.

[123] M. C. Koval, M. R. Dogar, N. S. Pollard and S. S. Srinivasa, "Pose estimation for contact manipulation with manifold particle filters," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4541–4548, 2013.

[124] A. Aggarwal and F. Kirchner,"Object recognition and localization: The role of tactile sensors," *Sensors*, vol. 14, pp. 3227–3266, 2014.

[125] C. Zito, M. S. Kopicki, R. Stolkin, C. Borst, F. Schmidt, M. A. Roa and J. L. Wyatt, "Sequential Trajectory Replanning with Tactile Information Gain for Dexterous Grasping under Object-pose Uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4013–4020, 2013.

[126] P. Hebert, N. Hudson, J. Ma and J. Burdick, "Fusion of stereo vision, force-torque, and joint sensors for estimation of in-hand object location," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 5935–5941, 2011.

[127] J. Bimbo, P. Kormushev, K. Althoefer and H. Liu,"Global estimation of an object's pose using tactile sensing," *Advanced Robotic Systems*, vol. 29, pp. 37–41, 2015.

[128] A. Petrovskaya and O. Khatib, "Global localization of objects via touch," *IEEE Transactions on Robotics and Automation*, vol. 27, no. 3, pp. 569–585, 2011.

[129] C. Corcoran, R. Platt, "A measurement model for tracking hand-object state during dexterous manipulation," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 4302–4308, 2010.

[130] M. Prats, P. J. Sanz, A. P. del Pobil, "Vision-tactile-force integration and robot physical interaction," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3975–3980, 2009.

[131] Web: http://www.labs.righthandrobotics.com/!reflex-hand-1/svmm4. Last visit in July, 2016.

[132] Y. Tenzer, L. P. Jentoft, R. D. Howe, "The Feel of MEMS Barometers: Inexpensive and Easily Customized Tactile Array Sensors," *IEEE Robotics and Automation Magazine*, vol. 21, is. 3, pp. 89–95, 2014.

[133] E. Salamin, *Application of quaternions to computation with rotations*. Technical Report, Stanford University Artificial Intelligence Laboratory, 1995.

[134] J. Funda, R. Taylor and R. Paul, "On homogeneous transforms, quaternions, and computational efficiency," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 382–388, 1990.

[135] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[136] A. Doucet, N. de Freitas and N. Gordon, "An Introduction to Sequential Monte Carlo Methods," in *Sequential Monte Carlo Methods in Practice*, pp. 3–14, 2001. Editors: A. Doucet, N. de Freitas and N. Gordon, Springer New York .

[137] J. V. Candy, "Bootstrap Particle Filtering," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 73–85, 2007.

[138] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[139] J. Liu, "Monte Carlo Strategies in Scientific Computing," New York: Springer-Verlag, 2001.

[140] B. Ristic, S. Arulampalam and N. Gordon, "Beyond the Kalman Filter: Particle Filters for Tracking Applications," Norwood, MA: Artech House, 2004.

[141] W. R. Gilks and G. O. Roberts, "Improving MCMC mixing," *Markov Chain Monte Carlo in Practice*, eds. W R. Gilks, S. Richardson and D. J. Spiegelhalter, London: Chapman and Hall, 1996.

[142] C. Robert and G. Casella, "A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data," *Statistical Science*, vol. 26, no. 1, pp. 102–115, 2011.

[143] N. J. Gordon, D. J. Salmond and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *EE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[144] R. Douc and O. Cappe, "Comparison of resampling schemes for particle filtering," in *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64–69, 2005.

[145] J. Hol, T. Schon and F. Gustafsson, "On Resampling Algorithms for Particle Filters," in *IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79–82, 2006.

[146] J. Pan, S. Chitta and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[147] E. Larsen, S. Gottschalk, M. C. Lin and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3719–3726, 2000.

[148] M. D. Shuster, "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.

[149] F. L. Markley, Y. Cheng, J. L. Crassidis and Y. Oshman, "Averaging Quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[150] G. M. Lerner, *Three-Axis Attitude Determination.* In Spacecraft Attitude Determination and Control, J. R. Wertz (ed.). Kluwer Academic Publishers, The Netherlands, 1978.

[151] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 3400–3407, 2011.

[152] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *Proceedings of the IEEE International Conference of Advanced Robotics*, pp. 510–517, 2015.

[153] D. Alvarez, J. V. Gomez, S. Garrido and L. Moreno, "Geometrically Constrained Path Planning with Fast Marching Square," in *Mediterranean Conference on Control and Automation*, pp. 1014–1019, 2015.

[154] Q. Wan, R. P. Adams, R. D. Howe, "Variability and predictability in tactile sensing during grasping," in *Proceedings of the IEEE Conference on Robotics and Automation*, to be published, 2016.

[155] J. N. Tsitsiklis, "Efficient Algorithms for Globally Optimal Trajectories," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, 1995.

[156] Q. Do, S. Mita and K. Yoneda, "Narrow passage path planning using fast marching method and support vector machine," in *IEEE Intelligent Vehicles Symposium Proceedings*, pp. 630–635, 2014.

[157] Y. Liu, W. Liu, R. Song and R. Bucknall, "Predictive Navigation of Unmanned Surface Vehicles in a Dynamic Maritime Environment when using the Fast Marching Method," *International Journal of Adaptive Control and Signal Processing*, pp. 1099–1115, 2015.

[158] N. Forcadel, C. Le Guyader and C. Gout, "Generalized Fast Marching Method: Applications to Image Segmentation," *Numerical Algorithms*, vol. 48, no. 1–3, pp. 189–211, 2008.

[159] S. Basu and D. Racoceanu, "Reconstructing neuronal morphology from microscopy stacks using fast marching," in *IEEE International Conference on Image Processing*, pp. 3597–3601, 2014.

[160] N. Al Zaben, N. Madusanka, A. Al Shdefat and H. Choi, "Comparison of Active Contour and Fast Marching Methods of Hippocampus Segmentation," in *6th International Conference on Information and Communication Systems*, pp. 106–110, 2015.

[161] X. Qu, S. Liu and F. Wang, "A New Ray Tracing Technique for Crosshole Radar Traveltime Tomography based on Multistencils Fast Marching Method and the Steepest Descend Method," in *15th International Conference on Ground Penetrating Radar*, pp. 503–508, 2014.

[162] X. Zhang and R. Bording, "Fast Marching Method Seismic Traveltimes with Reconfigurable Field Programmable Gate Arrays," *Canadian Journal of Exploration Geophysics*, vol. 36, no. 1, pp. 60–68, 2011.

[163] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms," *SIAM Journal on Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[164] A. Chacon, *Eikonal Equations: New Two-scale Algorithms and Error Analysis*. PhD thesis, Cornell Univeristy, 1 2014.

[165] J. A. Sethian, "Fast Marching Methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[166] R. Kimmel and J. A. Sethian, "Computing Geodesic Paths on Manifolds," *Proceedings of the National Academy of Sciences*, vol. 95, no. 15, pp. 8431–8435, 1998.

[167] J. A. Sethian and A. Vladimirsky, "Fast Methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes," *Proceedings of the National Academy of Sciences*, vol. 97, no. 11, pp. 5699–5703, 2000.

[168] S. Ahmed, S. Bak, J. McLaughlin and D. Renzi, "A Third Order Accurate Fast Marching Method for the Eikonal Equation in Two Dimensions," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2402–2420, 2011.

[169] S. Luo, "High-order Factorizations and High-order Schemes for Point-source Eikonal Equations," *SIAM Journal on Numerical Analysis*, vol. 52, no. 1, pp. 23–44, 2014.

[170] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. No. 3, Cambridge University Press, 1999.

[171] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[172] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[173] J. V. Gómez, D. Álvarez, S. Garrido and L. Moreno, *Fast Methods for Eikonal Equations: an Experimental Survey*. Available at http://arxiv.org/abs/1506.03771.

[174] J. V. Gómez, *Fast Marching Methods in path and motion planning: improvements and high-level applications*. PhD thesis, Carlos III University of Madrid, RoboticsLab, 2015.

[175] I. Arvanitakis, A. Tzes and M. Thanou, "Geodesic Motion Planning on 3D-terrains Satisfying the Robots's Kinodynamic Constraints," in *Annual Conference of the IEEE Industrial Electronics Society*, pp. 4144–4149, 2013.

[176] D. Ogay and E. Kim, "Kinodynamic Motion Planning with Artificial Wavefront Propagation," *Journal of Information and Communications Convergence Engineering*, vol. 11, no. 4, pp. 274–281, 2013.

[177] D. Álvarez, "Cartesian controller for the LWR-UC3M-1 robotic arm in MANFREDV2," Master's thesis, Carlos III University of Madrid, Nov 2011.