**A NOVEL TECHNIQUE FOR EFFECTIVE GRASP SYNTHESIS AND THE INTRODUCTION OF A COMPUTATIONAL TOOLKIT FOR EXPLORING THE GRASPING PROBLEM**

APPROVED BY SUPERVISING COMMITTEE:

_____
First Name Last Name, Ph.D., Chair

_____
First Name Last Name, Ph.D.

_____
First Name Last Name, Ph.D.

_____
First Name Last Name, Ph.D.

Accepted: _____
Dean, Graduate School

# DEDICATION

# A NOVEL TECHNIQUE FOR EFFECTIVE GRASP SYNTHESIS AND THE INTRODUCTION OF A COMPUTATIONAL TOOLKIT FOR EXPLORING THE GRASPING PROBLEM

by

JORGE NICHO, M.E.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Engineering
Department of Mechanical Engineering
December 2010

## ACKNOWLEDGEMENTS

December 2010

# A NOVEL TECHNIQUE FOR EFFECTIVE GRASP SYNTHESIS AND THE INTRODUCTION OF A COMPUTATIONAL TOOLKIT FOR EXPLORING THE GRASPING PROBLEM

Jorge Nicho, M.E.
The University of Texas at San Antonio, 2010
Supervising Professor: Brent Nowak, Ph.D.

In today's industry, numerous types of industrial grippers are used to carry out grasping task that demand a great deal precision and reliability. Thus, the majority of commercial grippers can meet industry demands as long as the work environment is highly structured and controlled. Nevertheless, their capabilities become insufficient in the presence of uncertainties and so the quality of the products is significantly degraded. On the contrary, the flexibility of dexterous robotic hand devices may provide the necessary means for dealing with uncertain and highly complex work environments. But, the inherent redundancy of highly-articulated robotic hands makes it very challenging to synthesize a suitable grasp that meets the desired task requirements. This problem has motivated the development of various grasp synthesis methods for determining grasps that realize a hand's full potential; although this is still a work in progress.

In addition, the use of physical robotic prototypes for evaluating grasp synthesis methods is not always feasible because they are scarce and present numerous technical and financial challenges. Thus, preliminary studies of robotic grasping methods would be greatly benefited from simulation software since it offers an easier route for investigating the effectiveness of several grasping methods. Also, a software alternative is flexible enough to accommodate large

sets of test scenarios that can lead to stronger conclusions about a particular technique; this is not the case for experimental test-bed.

The work in this thesis addresses two main topics with the aim of presenting suitable solutions to the problems described above. The first goal of this thesis is to introduce a new technique for grasp synthesis that is applicable to dexterous hand models of any kinematic configuration and geometry. This technique relies heavily on continuous collision detection (CCD) in order to find suitable fingers postures candidates that may increase the quality of the grasps. In addition, it utilizes a search procedure involving 6–dimensional hyperplane half-spaces for seeking and selecting finger posture candidates that increase the grasp quality. Also, grasps are rejected whenever interferences between links of the hand are detected. The effectiveness of this method is demonstrated through three experiments involving three different hand models undergoing similar task conditions.

The second goal in this thesis is to present and describe the ongoing development of a computational toolkit that incorporates sufficient number of tools for examining and simulating a multitude of grasping tasks. Moreover, the addition of several modules allow conducting computations related to computational geometry, grasp mechanics, collision detection and robust physics simulation; which aid in the development of grasp synthesis methods. This toolkit also relies on scripting in order to describe an entire task, with various goals and constraints, in terms of compatible functions or word commands written on a script. By editing the script, it's possible to specify a new set of goals and constraint very quickly and the results become available almost immediately. In the end, an experiment that combines the grasp synthesis technique with the physics simulator serves to demonstrate the benefits of using the tools in the computational library towards the investigation of robotic grasping.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

## 1.1 A View of Robotic Manipulators Today

In most industries of today, a wide range of serial robotic arms have become an essential part of many large scale manufacturing processes. Robotic manipulators have proven to work very reliably in processes that demand strict precision and accuracy requirements. Despite the robustness offered by robotic arms, they need to be properly equipped and surrounded by a well controlled work-environment. Hence, the vast majority of robotic work cells in a typical industrial setting consist of a robotic manipulator arm, a structured work environment and a gripper. The most common manufacturing tasks, such as welding, painting and assembly, are carried out by full robotic systems. These tasks require that the robot arm performs a number of actions that involve gripping and manipulating objects in order to fulfill a given set of requirements.

The main means for a robotic arm to interact with its environment is through a device known as a gripper or end-of-tooling; the gripper makes it possible to carry out manipulation and gripping tasks. Most commercial grippers come in a variety of forms and target a very particular task or object. Some of them consist of two or three jaws with prismatic joints that perform simple open-close movements. These are the most common types of grippers and are typically used to hold and transport small object of cylindrical or box – like shapes. Other end – effectors rely on suction cups to lift delicate glass sheets that may not be handled by other grippers. Since only simplistic gripping and manipulative actions can be achieved with a commercial gripper, several offline preparations are necessary in order to meet the task requirement.

During offline planning, a human operator specifies all actions and movements to be carried out by the robot and gripper in accordance to his knowledge and understanding of the

tasks and constraints. This process of predefining all actions must be very meticulous in order to compensate the lack of adaptability that most industrial robotic systems suffer. Consequently, additional changes introduced in the process are difficult, time consuming and costly to incorporate. In many cases, a great deal of retooling and hardware changes must be made in order to meet the new requirements.

The inconveniences associated with offline planning and retooling have created a need for highly dexterous robots that can adapt to uncertainty in order to complete their task in an efficient and timely manner. Thus, this necessity has inspired the study and investigation of highly dexterous robotic hands. Generally, dexterous robotic hands posses numerous degrees of freedom and many of them are designed with close resemblance to the human hand. Furthermore, the effective use of robotic hand devices can increase the adaptability and flexibility of an entire robotic system. Consequently, they may reduce or completely eliminate the cost and time associated with retooling.

## 1.2 The Field of Robotic Grasping

The large degree of redundancy in robotic hand mechanisms makes it very challenging to select an appropriate grasp due to the vast number of possibilities that exists. Thus, it is desired to develop generalized methodologies for selecting suitable grasps that fulfill a number of objectives as best as possible; the study of these methodologies is part of the field of study known as robotic grasping. The field of robotic grasping seeks to understand the mechanics of interaction between a dexterous hand and an object with the aim of developing suitable solutions to highly complex grasp problems. Of special interest is the problem of grasp synthesis, which consists of finding an adequate set of contact locations required to form a grasp that meets a

desired condition. Furthermore, a number of fundamental grasp mechanics concepts have been developed in order to serve as tools for devising grasp synthesis methods that seek to exploit the full potential of robotic hands.

The concept of grasp mechanics help assessing the grasp in terms of the set of forces and moments that can be applied through contact points distributed across the surface of an object. In the context of grasping, a force and moment pair can be represented by a single entity known as a wrench. Hence, a set of wrenches associated with a unique grasp can be grouped together into a much larger entity known as the wrench space [1]. The wrench space is a theoretical construct, as applied to grasping, which characterizes the set of forces that a grasp can exert on an object.

A number of properties about the grasp can be inferred directly through the wrench space. In particular, Force-Closure and grasp quality [1] have been defined as properties unique to a particular wrench space. For instance, Force-Closure (FC) is defined as the ability to restrain the motion of an object in all directions. An equivalent interpretation of this concept implies that the origin must lie in the interior of the wrench space. Similarly, the grasp quality is a measure of how effectively a FC grasp can reject disturbance forces.

### 1.3 A New and Novel Technique for Grasp Synthesis

Through the years, a number of algorithms have been developed for synthesizing grasps for 2D and 3D objects. However, a generalized grasp synthesis method for quickly resolving grasps that account for various task constraints and requirements is still an open research problem. Hence, the first goal of this thesis is to introduce a novel technique for synthesizing grasps effectively. A sophisticated continuous collision detection [2] technique is used to

determine valid finger configuration candidates such that a finger may exert restraint forces on an arbitrary object through direct contact. Each valid finger configuration candidate contains at least one point of contact with the object. Each contact has an associated friction cone [1] that represents a possibility space of generalized forces that are applicable by the finger through the contact.

The grasp synthesis technique is divided in two main stages. The first stage consists of finding an initial Force–Closure grasp that immobilizes the object but may be easily disrupted by disturbance forces. The second stage of the procedure processes the initial FC grasp and attempts to improve it by evaluating new finger configuration candidate in order to choose the candidate that contributes the most towards strengthening the grasp. The procedure here relies on finding a set of hyperplanes that reside in the weakest region of the wrench space. These hyperplanes are use to determine a unique finger configuration whose associated friction cone(s) produces the largest expansion in the weakest region of the wrench space.

Additionally, this technique can filter out combinations of finger configurations that are tagged as invalid due to interference of links. Hence, only fully achievable grasps are processed and the resulting grasp can be executed by the hand model on the object. Furthermore, this technique is applicable to a wide range of hand models and objects of any complexity as long as the object lies within the reaches of the hand's workspace. This technique's speed and robustness make it very suitable for grasp planning applications where reliable grasps must be computed quickly. Results of the grasps found by this technique with three different hand models are shown in this paper.

## 1.4 Robotic Grasping Studies

Several types of studies have taken place in order to gain a greater understanding of robotic grasping. These studies have relied on mechanical analysis, physical test bed and simulation software. The mechanical analysis has focused on understanding how the distribution of contact points incurs in the generation of forces and moments that lead to the formation of an adequate grasp. Most of the theoretical foundation in robotic grasping has originated from exploring grasping through the mechanical analysis of the interaction between the hand and object; this is demonstrated in [1] and [3]. Consequently, analytical approaches have led the development of theoretical tools for studying robotic grasping.

However, analytical approaches can only tackle basic idealized problems and tend to be insufficient for providing solutions to real-world problems. The numerous uncertainties and nonlinearities inherent to the robotic grasping problem suggest that construction of an experimental test-bed may be necessary. Hence, experimental robotic hands have been used in a number of studies in order to obtain more concrete and practical insight of underlying issues in robotic grasping that can only be exposed through this approach. In addition, the use of experimental test-bed opens up many avenues to various topics that may not be explored in sufficient depth otherwise.

## 1.5 Robotic Hands

Experimental test-beds provide scientist with the means for exploring real world problems in order to find appropriate solutions that adhere more closely to reality. They also serve as tools for measuring the effectiveness of a particular approach or method. For these

reasons, a limited number of dexterous robotic manipulators have been built and studied in order to gain further understanding of robotic grasping.

One of the first experimental robotic hands to be built is the MIT Dexterous hand, which was developed at the University of Utah [4]. It features a quasi-anthropomorphic design and incorporates four fingers with four degrees of freedom each as well as a series of cables for indirect actuation. It had also been equipped with strain gages and Hall Effect Sensors housed within encoders. Other efforts include the Robonaut Hand that was developed at NASA as part of the Robonaut Unit [5]. In more recent times, a company known as the Shadow Robot Company produced the Shadow Hand [6]. This hand possesses a total of 24 degrees of freedom and features 40 artificial air muscle that offer a high level of compliance rarely seen on these devices.

Nevertheless, the construction of experimental robotic hands represents an immense technical and financial challenge. Most of the required components are not commercially available and custom made components are very costly. Also, adequate integration of such level of sophistication and cutting-edge technologies is a challenge in its own right. Electrical and mechanical issues unforeseen in the design phase may appear much after a fully functional hand device has been constructed. Hence, these problems may become very difficult to pinpoint and their correction may allocate too many resources for a very prolonged time.

## 1.6 Simulation Software in Robotic Grasping

Another alternative for conducting robotic grasping studies involves the use of virtual simulation environments because they provide an approximation of real physical phenomena.

Simulation environments have been used successfully in a number of areas such as haptics applications, flight simulators, training programs, industrial robotics and video games.

Almost all physics simulations are implemented with the use of a set of programmatic tools known as physics engines [7], [8], [9], and [10]. Physics engines can enforce the laws of physics through a number of sophisticated mathematical formulas that have been described in terms of computer code. Most of them contain a collision detection system that enables all elements within a physical system to interact with one another [11], [12]. Some of the most sophisticated physics engines have the ability to compute the effects of magnetic fields or gravitational pull between larger planetary bodies.

In relation to robotic grasping, one of the most beneficial aspects of simulation software lies in its ability to accommodate a multitude of complex test scenarios that allow conducting a thorough evaluation of a particular grasping method; thus arriving to stronger conclusions about the method. In addition, more confident decisions can be made about a particular hand design based on its performance in simulation. In this regard, a simulation oriented approach has a clear advantage over the use of physical test bed that only allows very few test cases to be studied. However, experimental test bed remains as the approach that provides the most accurate results and can serve to truly test a method after it has been thoroughly examined in simulation.

## 1.7 A New and Versatile Computational Tool for the Study of Robotic Grasping

The second major objective of this thesis is to present a modular and flexible programmatic toolkit that provides a wide range of tools for the study of robotic grasping. This toolkit features a very intuitive object–oriented design that minimizes the efforts required to plan and execute grasps in complex environments. In addition, it features a physics simulation

module that makes it possible to create very detailed work environments for evaluating the performance of a hand model.

The toolkit programs can process a generalized definition of a hand model rather than a particular type of hand. This feature facilitates the incorporation new hand models of any complexity; all hand models are stored into the toolkit database, which is accessible at any time. Furthermore, the modular design of the toolkit favors the accommodation of additional modules that extend its functionality. A well known scripting environment was selected for the user interface since scripting allows a greater level flexibility and control than classical graphical user interfaces (GUI).

## 1.8 Organization of the Thesis

Towards providing a thorough explanation of the goals described above, the contents of the thesis have been laid out as follows:

Chapter 2 outlines the most relevant concepts and contributions that constitute the theoretical foundation of robotic grasping. The importance of these concepts are further emphasized and put into context in the grasp synthesis section. A brief overview of the related collision detection paradigms are given in this chapter as well. The last section of this chapter summarizes the most notable works involving simulation software applied to robotic grasping.

In chapter3, a detailed explanation of the concepts that constitute the mechanics of grasping is given. In addition, the computational implementation of the related algorithms are discussed and described in great detail. Another object of this chapter is to introduce a number of important notations and conventions that will be adopted throughout thesis.

Chapter 4 presents the computational geometry methods that have been used in this work. It begins by describing the implementation of the GJK distance and Ray-Shooting algorithms and discusses their relevance to the 6-Dimensional grasp wrench space. This is followed by a discussion of hyper-plane half-spaces and their importance in the work presented here. Next, a brief mention is made about convex hull computations and how they are used to analyze the grasp wrench space. Lastly, this chapter introduces a recursive search technique that relies on continuous collision detection to find valid finger configuration candidates. Hence, the methods described in this chapter 3 and 4 set the tone for describing the novel grasp planning technique that is presented in this thesis.

Chapter 5 begins by providing a general definition of the grasp synthesis problem. Then, previous related works and their associated advantages and disadvantages are discussed. The rest of the chapter is devoted to describing each stage of the grasp planning technique in detail. Then, the results of three groups of experiments are presented. Each group of experiments was carried out with a different hand model and three objects. These experiments are meant to demonstrate the robustness of the technique by showing how it can adapt to different hand models of differing complexities. The last section of the chapter provides a discussion of possible improvements that could be made to the technique.

Chapter 6 explains the overall structure of the Robotic Grasping Toolkit as well as how robotic grasping research may benefit from it. Throughout this chapter, it is demonstrated how it became possible to achieve the design goals by exploiting object oriented programming practices in conjunction with scripting. Then, an overview of the physics simulation module available in the toolkit is given.

In chapter 7, an experiment demonstrates the use of the physics simulation in conjunction with the grasp synthesis technique for conducting a simple grasping task. Finally, chapter 8 provides a summary of the thesis and describes possible future work.

# CHAPTER 2: LITERATURE RESEARCH

The need for autonomous manipulators with a great deal of dexterity and precision has led to the study of highly dexterous robotic hands. Hence, the desire to produce a device with superior adaptability has given birth to the field of robotic grasping. One of the most common issues with robotic hands is that of redundancy. The numerous degrees of freedom found in robotic hands make it difficult to find a unique hand posture that produces a reliable grasp. Therefore, researchers have devoted tremendous efforts in developing conceptual tools for creating and exploring effective grasping methods. Consequently, the purpose of this chapter is to describe some of the most relevant contributions that are applicable to the field of robotic grasping.

## 2.1 Grasp Mechanics

The concepts of grasp mechanics constitute a number of conceptual tools for analyzing several grasping problems. This analysis can lead to the formulation of several grasping solutions. Hence, this section is devoted to discussing the many contributions that have led to the development of these concepts.

### 2.1.1 Friction Cone

One of the most important contributions in the field of robotic grasping is found in the work of Salisbury [17]. In his work, he defined the various types of contact that arise during hand-object interaction. A total of nine contact types encompassed all the possible pairings between lines, point and surface features. Each contact allows a unique set of forces and freedoms to develop between the two objects in contact.

In addition, the type of friction model selected can alter the set forces that develop at the contact. Generally, three types of friction models have been used to represent the restraint that a finger may exert on an object [18] – [22]. These friction models have come to be known as frictionless point contact (FPC), point contact with friction (PF), soft finger elliptical Contact (SFCe) and soft finger linear contact (SFCl).

The FPC model is the simplest one of them and can only exert a force in the direction of the inward contact normal. The PF, SFCe and SFCl models represent a space of frictional forces that may be exerted through the contact. Their mathematical models correspond to a set of nonlinear inequalities whose solution domain is said to encompass the friction cone [23] – [26]. A larger friction cone can provide a greater degree of immobility at the contact. Hence, the properties of a grasp depend entirely on the extent of friction cones that constitute it.

The formulation of methods that allows assessing the grasp has been difficult due to the nonlinear nature of the friction cones. Several investigators have proposed various approaches that attempt to cope with theses nonlinearities. In [29] Nakamura formulated the Force-Closure test as a 12 linear programming problem. In [30], Bicchi concluded that there was a clear similarity between the Force-Closure test and the stability of an ordinary differential equation. Ciocarlie formulated the SFCl frictional constraints as a linear complementary problem [31].

Other approaches have dealt with a polyhedral approximation of the constraints imposed by the friction cone. This polyhedral approximation has given light to the development of techniques that treat the grasp problem from a geometrical perspective. In addition, the discretization of the friction cone has allowed generalizing the formulation of several methods that were previously limited to a particular type of friction cone.

A method for linearizing the Soft Finger Contact constraint was provided by Zheng [32]. Then, he also proposed an algorithm for optimal force distribution of soft multifingered grasps. The work in [34] by Liu formulated the Force–Closure test to that of a Ray-Shooting problem [33]. In [35], the Ray-Shooting method is used to provide numerical solutions to various problems in robotic grasping that were expressed in terms of linearized friction cones. Hence, the work in this thesis makes extensive use of discrete approximations of the friction cones in order to examine a number of grasping problems.

### 2.1.2 Wrench Space

A force-moment pair that lies within the bounds of the space defined by a friction cone can be conveniently represented by a wrench vector. This wrench vector is 3-dimensional in the context of 2-dimensional problems. For 3-dimensional problems, the wrench vector has 6-dimensions [17]. Therefore, each friction cone can be equivalently represented by a set of wrench vectors that exists within a space of generalized forces. Consequently, a polyhedral approximation of the friction cone becomes a finite number of wrench vectors or wrench primitives that provide a discrete approximation of the space.

The wrench space comprises the set of all primitive wrenches that can be applied by a grasp through the contact points. Therefore, the set of wrenches associated with the contacts of a grasp form the Wrench Space of that grasp [13]. Another interpretation describes the wrench space as the convex hull that bounds the union of all the wrench vectors produced by each friction cone in the grasp. This last definition has been found to be more convenient when attempting to create the wrench space numerically.

The wrench space is a theoretical construct that is well understood throughout the literature of robotic grasping. However, there are alternatives to its numerical construction that

lead to different wrench spaces for the same grasp. The main reason for these variations lies in the decision of scaling the wrench space so that it may provide a more general assessment of the grasp. In [36], Pollard chose to scale the magnitude of the torque component of the wrench vector in relation to the maximum distance from the surface of the object to its center of gravity. This ensures that the size of the wrench space was independent of object scale.

The work in [37] by Ferrari and Canny defines the notion of a unit wrench space and provides two possibilities for its construction. The first option consists of limiting the magnitude of the contact normal forces to one. Then, the unit wrench space is the result of the convex hull applied to the Minkowski sum of all the wrench sets produced by the friction cones. The implementation of this operation has a very high computational cost since the Minkowsky sum performs a combinatorial procedure that adds the elements of one set to every other element in another set.

The second option proposed by the same authors places a bound on the sum of the magnitude of the contact normal forces. This alternative does not require a Minkowsky sum of the elements and thus, the computation becomes more manageable. It is important to mention that the wrench space computed as in the second alternative is a subset of wrench space computed in the first. The second alternative is the preferred method by most investigators and it is used in this thesis as well.

The importance of the Wrench Space lies in the fact that it provides a meaningful way to characterize the properties that define a particular grasp. The properties that can be inferred from the wrench space are the following: Form-Closure, Force-Closure [1], grasp quality [29], manipulability [26], [38] and stability [39], [40]. Form-Closure can be interpreted as a particular

case of Force-Closure. This thesis is mainly concerned with Force-Closure and Grasp-Quality and how they are used to formulate methodologies for effective grasp synthesis.

### 2.1.3 Force-Closure

A Force-Closure grasp is defined as a grasp that can guarantee the immobility of an object in the presence of disturbance forces [29]. A grasp may achieve this condition when the forces imparted through the contacts may nullify the effects of any external disturbance force. One key element for the formation of a Force-Closure grasp is the existence of internal forces. The internal forces are a subset of the contact forces so that their sum equals a zero net force. The redistribution of the internal forces is what enables a FC grasp to reject disturbance forces.

Several studies have investigated the conditions that are necessary to attain Force-Closure. In this regard, one of the pioneering works conducted by Reuleaux [41] demonstrated that restraining a planar object against disturbance forces required a minimum of four frictionless point contacts. Eventually, the work done by Lakshminarayana [42] showed that only seven frictionless contact points properly located were needed so as to fully restrain a 3-Dimensional object. Furthermore, it was proven by Markenscoff [43] that four frictionless contact points guaranteed immobility in 2-Dimensional space. In addition, he proved that full restraint of 3-Dimensional objects could be ensured by seven frictionless contact points as long as the objects had no rotational symmetry. His work also demonstrated that only four contact points were necessary whenever friction was taken into consideration. It was later found by Mirtich and Canny [44] that the number of frictional contacts could be lowered by one and still maintain immobility when rounded fingertips were used. Then, Nakamura [30] formulated 12 nonlinear programming problems to test for Force-Closure. In [29], Bichhi concluded that the stability of an ordinary differential equation was closely related to the Force-Closure condition.

Another definition of the Force-Closure has also been described in terms of the Wrench Space. The grasp is said to be Force-Closure whenever it is capable of exerting arbitrary wrenches in all directions within the 6–dimensional space of generalized forces [38]. This implies that when a grasp is Force-Closure, the convex hull of its corresponding Wrench Space must contain the origin it its interior. Force-Closure test methods that directly analyze the Wrench Space typically rely on geometric approaches rather than analyzing the placement of the contacts on the surface of the object. The work in [35] applied the Ray-Shooting method to several grasping problems including Force-Closure test, optimal force distribution and task-oriented grasp quality computation. This thesis uses a slight variation of the Force-Closure test presented in [35] and will be explained in great detail in chapter 3.

### 2.1.4 Grasp Quality

The need to perform a quantitative assessment of the reliability of a grasp has led to the development of a metric know as the grasp quality. The various facets of robotic grasping have made it almost impossible to provide a unique metric to grade all aspects of a grasp. Therefore, there are several quality metrics that allow measuring various attributes that are relevant to common grasping tasks. One type of quality metric that has received the most attention focuses on the ability of the grasp to reject disturbance forces. Other metrics provide a measure of the dexterity and ability of a grasp to manipulate objects.

In order to assess the degree of immobility that a grasp imparts on an object, several metrics have been proposed. The most popular metric in this regard is the Worst-Case Quality metric also known as the Largest-Ball criterion. This quality measure has been defined as being equal to the radius of the largest hyper-sphere centered at the origin such that it can fit in its entirety inside the wrench space [37]. In the context of grasping, the Largest – Ball metric

represents the minimum required magnitude that an arbitrary disturbance wrench needs to break the grasp. The Largest-Ball criterion is independent of task requirements and provides a very general means of grading a grasp.

The value of the Largest-Ball metric may vary in relation to the frame of reference that is used to compute the moment vectors. There have been attempts to eliminate this duality as shown in the work by Teichmann [45]. In [46], Miller and Allen used the volume of the wrench space since its value does not depend on the choice of reference. However, this metric may fail to provide insight about the largest disturbance force that can be resisted.

Other metrics provide a quality measure in relation to a task that has been properly characterized. Hence, it is necessary to provide a suitable description of the task such that it can be put into common terms with the Wrench Space. In general, the task is expressed as a group of generalized forces that should be applied on the object in order to achieve a particular goal. Furthermore, this set of task oriented forces can be approximated with a discrete polytope that bounds them as done by Li and Sastry [47] and Zhu [48]. Hence, the grasp quality is given as the scaling factor required for expanding the task polytope until it becomes largest and fully contained in the wrench space. An interesting method that relied on the Ray-Shooting technique to compute the task-oriented grasp quality was provided by Zheng [35].

In addition, several computational methods have been formulated for calculating the grasp quality that corresponds to the Largest-Ball metric. In [46], Miller and Allen used the "Qhull" [49] program to compute the radius of the largest inscribed hypersphere. Borst relied on an incremental algorithm to compute the quality [50]. Zhu and Wang [51] used a polytope substitution and computed the quality by solving linear programming problems. Zheng proposed a method that did not rely on linearizing the friction cones [52]. In [53], a very extensive survey

on grasp quality was provided by Roa. The computational library "Qhull" [49] is used in this thesis to compute grasp quality value as it was done in [46].

## 2.2 Grasp Synthesis

The concepts that constitute the foundation of grasp mechanics were studied and developed in order to provide analytical tools to be used in formulating sophisticated methodologies for prehensile tasks. The core of robotic grasping lies in the elaboration of efficient algorithms that can synthesize grasps for complex tasks. Consequently, grasp synthesis has been the focus of several research efforts that has been conducted in robotic grasping.

Grasp synthesis consists in the selection of contact points such that they form a Force-Closure grasp that presents optimality in relation to some designated criteria. These criteria are most commonly expressed in terms of how well a grasps can fully restrain an object affected by external disturbances. Given the vast number of possible grasp configurations, it becomes extremely difficult to determine a unique solution that fulfills the desired requirements. Therefore, grasp synthesis remains an open-ended problem and has been one of the most difficult in the field of robotic grasping.

In addition, the study of grasp synthesis can be categorized in three well distinguished groups: Physiological, hand-based synthesis and contact-based synthesis. Physiological studies attempt to mimic the human hand and so they are very empirical in nature. On the contrary, hand-based synthesis and contact-based synthesis rely on a more analytical approach.

### 2.2.1 Human Hand

The influence of the human hand in the study of robotic grasping is undeniably revealed in many of the works that have become part of the literature. Consequently, one of the pioneering works by Napier [54] established the first taxonomy of prehensile posture that

categorized grasps in relation to power, precision, dexterity and object compliance as observed in the human hand. Eventually, Cutkosky [55] extended the taxonomy in order to consider additional postures that were applicable to manufacturing tasks.

A more flexible classification was provided by Iberall [56] in order account for the fact that the human hand relied on multiple types of grasp simultaneously during practical tasks. The concept of virtual fingers was used by Lyons [57] to select grasp in relation to certain object characteristics such as shape and size as well as degree of firmness and precision. However, his classification was too general and could not be used to provide a grasp tailored to a specific object.

Another classificatory approach had been presented by Stansfield [58]. An existing classification of grasps was built into a rule based system. This system could select a set of appropriate hand pre-shapes and approach directions once a number of parameters that characterized an object were specified. However, no attempt was made to select a unique configuration that presented sizeable advantages over the rest.

In [59], a mapping between poses of human hands and a robot hand was carried out by Pao and Speeter. A DataGlove allow recording joint angle values that were then used to reproduce corresponding poses with the robotic hand. The work presented in [60] by Biggs attempted to estimate muscle excursion based on finger pose. In other study, Valero-Cuevas [61] conducted thorough studies on the influence of tendon networks in the torque production of fingers. Cadaveric experiments permitted exposing the propagation of tension in the tendon network.

### 2.2.2 Hand Based Grasp Synthesis

This methodology takes into consideration the geometric features, kinematic configuration and torque production capabilities that characterize a particular hand mechanism. By considering these elements it is possible to produce grasps configurations that do not exceed the physical limitations of the hand. In addition, the placement of the palm is commonly considered a challenging task that demands to be treated as a separate problem.

One of the first contributions in this topic was made by Jameson [62], who used a physical hand to generate a three contact grasp. His method was based on minimizing the amount of friction that was necessary to lift an object. In the work of Wren and Fisher [63], it was proposed to use predefined postures or hand pre-shapes and then drive the hand to a close configuration. Smith [64] computed parallel-jaw grips on n-sided polygons by relying on a polynomial time algorithm.

A grasp planner to find precision-grasp for the DLR hand was implemented by Borst [50]. The system required choosing a starting point on the object and then a reliable grasp was found with a meticulous heuristic. In [65], Hester reduced the grasp configuration space by fixing the configuration of the palm, thus eliminating 6 parameters. Then, the contribution of the fingers and the palm towards the formation of a grasp were measured separately with different performance criteria. This method was then used to operate the Robonaut hand developed by NASA.

The work done by Miller [66] consisted in the utilization of hand preshapes to generate starting hand configurations as well as approach directions for the palm. In addition, an auto-grasp feature closed the finger joints at preset velocities until the finger links registered a

collision with the object. Several experiments were conducted on the Barret hand using a simulator of his own invention known as "GraspIt!" [67].

In more recent years, Xue, Zoellner and Dillmann [68] took on an innovative approach by using sophisticated continuous collision detection (CCD) technique and swept volumes to find a large number of reachable contact points. They reported to have found over 10000 reachable contact points between the Shunk hand and a sphere. Furthermore, the same authors used a Kd-tree to index grasp configuration according to their grasp quality [69]. Their work made an important contribution towards breaching the gap between hand-oriented and contact-oriented synthesis. Consequently, the work in this thesis uses a heuristic, similar to [68], which relies very strongly on continuous collision detection to generate valid finger configuration candidates.

### 2.2.3 Contact Based Grasp Synthesis

At times it is convenient to decouple the grasp synthesis problem from the limiting physical aspects inherent to robotic hands. Such separation alleviates the burden of formulating generalized solutions that must inevitably account for these physical constraints. Thus, this allows prioritizing on other aspects that can lead to grasps, which provide a more proper fit on an object. This approach yields grasp configurations that could have been omitted if the search for solutions was influenced by the physical limits of robotic hands.

Therefore, contact based grasp synthesis proposes a methodology that places greater focus on conforming to the characteristics of the object rather that the limiting features of a particular hand. This strategy attempts to reduce the grasp configuration space to a smaller set of grasps whose contact points accommodate more appropriately to the object. This smaller set of

grasps can then be examined in terms of the hand characteristics such that it becomes easier to select a reliable grasp from a more manageable solution space.

Consequently, there have been many efforts that opted for contact-based synthesis approach in order to device adequate grasp synthesis methods. For instance, the work by Markenscoff and Papadimitriou [70] showed an algorithm for computing optimal grasps with three contacts. Their method was applicable to polygons whose center of gravity was unknown. In [71], Chen and Burdick presented a system that could find antipodal grasp for 2 and 3-dimensional objects with irregular shapes. A system that computed Form-Closure grasps with 2 contacts was described by Ponce in [72]. Such system could process curved 2-Dimensional objects represented as groups of polynomial parametric curves. Its use was demonstrated by driving a PUMA robot equipped with a parallel jaw gripper.

Furthermore, the introduction of triangular meshes used for approximating 3-Dimensional objects [73] gave light to a new category of methods for grasp synthesis. In relation to that, Borst stated that it was quick and efficient to generate random grasps from which adequate grasps could be chosen [74]. An algorithm for fixture synthesis of discrete objects was proposed by Wang [75]. His algorithm minimized positioning errors such that a fixture could be found. In [76], Liu developed a method that could find a Force-Closure grasp if the discrete sampling of the object's surface permitted it.

The work done by Roa [77] presents a method that finds locally optimum FC grasps for contacts with any type of friction model. His approach sampled an object in order to produce a dense collection of contact points and normal pairs that approximated the surface of the object. This cloud of points was then passed to a two stage procedure. The first stage found an initial FC grasp with no optimality in the shortest amount of time. The second stage attempted to

optimize the FC grasp found in the first stage by substituting contact points for new points that had a likelihood of increasing the grasp quality. The Force-Closure test as well as the grasp quality had to be computed after every substitution. The search for an optimum grasp ended when no new combination of points could continue to increase the quality.

In [78], Roa refined his technique in order generate independent contact regions (ICRs) that provided some flexibility to finger positioning errors. The concept of ICRs had been explored previously by other authors in [79] and [80]. The idea was that if at least one contact was located inside each ICR then a Force-Closure grasp could be guaranteed. In addition, Roa correlated the size of the ICRs to the value of the grasp quality. His work involves a very geometric-oriented treatment of the 6-Dimensional wrench space. The methods he presented make extensive use of hyperplane half-spaces for determining regions of the wrench space where feasible contact candidates could be found.

The method for effective grasp synthesis presented in this thesis shares some similarities with the work done by Roa. In particular, hyperplanes are utilized as decision boundaries here as well. However, a large reduction on the number of Force-Closure and Grasp-Quality computations is demonstrated by the grasp synthesis technique in this thesis. It is important to point out that the work presented here unifies a number of hand-based and contact-based grasp synthesis methods with the aim or producing reliable grasps for a given robotic hand.

## 2.3 Collision Detection

There is clear evidence that collision detection (CD) has had a great deal of influence in some of the most recent works in the literature of robotic grasping; this fact is demonstrated in a multitude of research efforts [66], [68]. In particular, the use of collision detection is very

predominant in the methods for grasp synthesis proposed by Xue [68], [69]. Similarly, the work in this thesis makes extensive use of collision detection techniques at various levels. Hence, with the aim of facilitating the understanding of later chapters, a brief summary on the literature of collision detection is given in this section.

Collision detection involves the use of algorithms, stated in terms of geometry, that allow checking for intersections between two objects in space. Furthermore, collision detection has been studied thoroughly for the last two decades. It's considered a solid and mature science with robust implementations that perform well under very rigorous conditions. The majority of CD methods treat the interference problem in a discrete sense. This implies that at a given time, interferences are only detected between objects with fixed spatial configurations. However, the need to focus on dynamic scenarios has lead to the development of methods that can anticipate collisions between objects whose motions are known. A very general classification of the branches of collision detection results in two distinct categories: Discrete and continuous collision detection (CCD).

### 2.3.1 Discrete Collision Detection

Discrete CD methods check for interferences at fixed time intervals. In most cases, it is common practice to use specialized algorithms that are compatible with unique features presented by some objects. A very significant contribution by Larsen [81] sampled 3-Dimensinal objects into hierarchical tree structures where each relevant geometric feature was bounded by a conforming type of swept sphere volume. A quick traversal of the tree allowed detecting intersections very rapidly. In addition, distance computations for non-intersecting objects could be performed as well. These algorithms were implemented in the open source computational package known as PQP (Proximity Query Package).

Most real time applications demand performing interference computations for various objects very quickly. This necessity led to the development of collision detection systems that could process numerous objects in a very short amount of time. Today, most collision detection systems incorporate at least two stages of interference detection. The broad-phase stage relies on simple approximations of the objects for detecting potential colliding pairs. The narrow-phase performs more rigorous collision test on the potential colliding pairs found during broad-phase culling.

In [11], Gino Van der Bergen provided several algorithms and methodologies that he had used in the design and development of his collision detection package named Solid. He also described spatial data structures and how they were used to reduce the number of intersection tests at the broad-phase. In more recent years, there has been an emergence of several physics libraries that allow creating very rich and sound simulation environments [7], [8], [9] and [10]. The majority of them include very sophisticated collision detection systems and dynamics solvers. In addition, they may provide advanced features such as convex decomposition, soft body support and parallel processing capabilities.

One of the major downfalls of discrete collision detection is found in the fact that at certain extreme conditions some collisions may be missed. This typically occurs when bodies undergo very high translational and rotational speeds. This condition is known as the tunneling effect and it takes place whenever a body changes its spatial configuration very drastically between consecutive time steps. As a result, this weakness has motivated the investigation and further development of continuous collision detection since it prevents missed collisions by taking the object's motion into account.

### 2.3.2 Continuous Collision Detection

The ability to report the first time of contact (TOC) between two moving objects is the highlight of CCD algorithms. CCD has received a great deal of attention because it handles non-penetration constraints effectively. Furthermore, it has been used in various types of applications that involve impulse based dynamics [83], constraint-based dynamics [84], avatar interaction in virtual environments [85], god-object computations for 6-DOF haptic rendering [86] and robot path planning [87].

In [88], Zhang applied a generalization of conservative advancement (CA) to general polyhedron. The method was applicable to articulated bodies that could contain non-convex links. The algorithm generated continuous motions for each link and reported TOC if a collision were to occur at some point during the motion. They reported achieving a computational time of 1.22 ms for an articulated body with 15 links that consisted of 20K triangles each.

The work done by Tang [2] featured a CCD algorithm that could be used on general polygonal models undergoing rigid motions. Their technique made no assumptions about the underlying geometry and topology of the models. This generalization was possible by incorporating swept sphere volume hierarchies that bounded relevant geometric feature of the models. Their work is also available as an open source computational package for the C++ programming language. This tool has been incorporated as a part of the Robotic Grasping toolkit presented here and it plays a major role in the grasp synthesis method featured in this thesis as well.

## 2.4 Simulation Software applied to the study of grasping

Simulation software has been applied in a variety of fields because it allows conducting thorough studies and evaluations of very involved processes at a low cost and short time. The current sophistication of simulation software makes it possible to create accurate and reliable representations of real physical phenomena. Consequently, this technology has been incorporated in a number of research efforts that aim to device adequate solutions for several grasping problems.

In 2000 [89], a graphical robot hand with a simplistic bone structure was developed with the use of the graphical software 3D Studio Max and Maya 3D. Some material properties and lightning effects were added to this model as well. Several results produced from kinematic and dynamic analysis were reported. The work done by Sancho-Bru showed a 3D inverse kinematic model of the human finger that was used to simulate the influence of muscular forces in free finger movements. A Hand Programming Language (HPL) was created by Speeter [90]. It attempted to provide a set of commands whose use became intuitive in the context of robotic grasping. In particular, common hand motion primitives were encapsulated into high-level programmatic functions.

In [67], Miller developed a simulation environment for robotic hand models undergoing grasping tasks. Real time collision detection operations are handled by a custom version of the Proximity Query Package (PQP) [82]. It also provided a simplistic mechanism to visualize 3-Dimensional projections of the wrench space. Rigid body dynamics were enforced by means of the ODE physics library (Open Dynamics Engine) [7]. A GUI (Graphical User Interface) was provided as the main form of interaction between the program and the user. Through this GUI it

was possible to jog the hand joints to a desired position and look at some results. In addition, the package included a number of robotic hand models that could be loaded into the simulation.

Additionally, other investigations in robotic grasping such as [91], [92] and [93] were carried out with the use of the commercial software package called ADAMS (MSC Software™). This software is a modeling and simulation tool that allows studying the performance of highly complex mechanical systems from multiple perspectives. However, the majority of its capabilities are tailored to automotive and aeronautics applications. Specialized tools for the analysis of grasping and robotic hands are not available with this software.

The work by Miller [67] remains as one of the most notable efforts to use simulation software towards the benefit of robotic grasping. It was very novel at the time and gave access to various innovative features. However, current software applications are required to provide access to new functionality that becomes available as technologies evolve. The toolkit introduced in this thesis tackles this necessity by making new functionality available through scripting. In relation to GUI's, scripting provides a superior level of control and favors software expandability.

# CHAPTER 3: THE MECHANICS OF GRASPING

The mechanics of grasping comprise the body of knowledge that constitutes the theoretical foundation of robotic grasping. Grasp mechanics provide number of fundamental concepts that aid in the conception of effective methodologies for complex grasping problems. This chapter aims to describe these concepts in great depth with the intention of providing insight on their relevance and applicability in the development of grasp synthesis methods.

## 3.1 Friction Cone Models

A hand interacts with an object through contacts. A contact permits applying a set of forces and moments whose directions and magnitudes depend largely on friction and the type of surfaces that come into contact. Hence, each type of contact model bounds the set of allowable forces that are applicable through the contact. A friction cone is an entity that represents the set forces that are defined by the contact model. The work in this thesis is concerned with four fundamental types of contact models and these are: Frictionless point contact (FPC), point contact with friction (PF), soft-finger elliptical contact (SFCe) and soft-finger linear contact (SFCl).

The forces and moments produced by the contacts define the characteristics of a grasp. Furthermore, a separate analysis of forces and moments fails to provide a general insight of the grasp. Hence, it is necessary to disolve the differing notions of forces and moments and treat them in a broad and generalized sense. A wrench is the representation of a force and moment in the space of generalized forces. Equivalently, a wrench provides a convenient means to encapsulate a force and moment pair as a single entity. A wrench vector can be created simply by concatenating a force and moment vector together as follows:

$$w = \begin{bmatrix} f \\ \tau \end{bmatrix} \tag{3.1}$$

Where $w \in R^3$ for 2-dimensional problems while for 3-dimensional problems, $w \in R^6$. Consequently, every force that exists in a friction cone has a corresponding wrench in the space of generalized forces. Furthermore, a common frame of reference must be designated in order to provide an equivalent context for all forces. Hence, it is required to map each force according to the object coordinate frame and contact constraints. Such mapping can be carried out with the use of the grasp map.

### 3.1.1 The Grasp Map

The grasp map is a convenient mechanism that encapsulates two important mapping operations. The first operation consists of determining the force components that are transmissible though the contact as defined by the corresponding friction cone. Such operation can be made with a wrench basis. Thus, each friction cone defines its own wrench basis in order to represent the set of allowable forces applied at the contact. The table 3.1 shows the wrench basis assigned to each contact type.

In general, a wrench basis $B_{c_i} \in R^{p \times mi}$ maintains a correspondance to the friction cone $FC_{c_i}$ of the $i^{th}$ contact $c_i$. For the 3-dimensional case $p$ = 6 and $m$ is the dimension of the wrench basis and indicates the number of independent forces that can be applied at the contact. Furthermore, a wrench $F_{c_i}$ is the generalized representation of the force $f_{c_i}$ at the contact $c_i$ with friction cone $FC_{c_i}$. Such relation is equivalent to the next statement:

$$F_{c_i} = B_{c_i} f_{c_i} \quad , \quad f_{c_i} \in FC_{c_i} \tag{3.2}$$

**Table 3.1: Wrench Basis.**

| Contact Type | Wrench Basis | Friction Cone |
|---|---|---|
| Frictionless Point Contact | $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ | $f_{c_i} > 0$ |
| Point Contact with Friction | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ | $\sqrt{{f_{o_i}}^2 + {f_{t_i}}^2} \leq \mu f_{n_i}$<br>$f_{n_i} \geq 0$ |
| Soft Finger Contact Linear | $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ | $\sqrt{\frac{{f_{o_i}}^2 + {f_{t_i}}^2}{\mu}} + \frac{\lvert f_{s_i} \rvert}{\gamma} \leq f_{n_i}$<br>$f_{n_i} \geq 0$ |
| Soft Finger Contact Elliptical | $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ | $\sqrt{\frac{{f_{o_i}}^2 + {f_{t_i}}^2}{\mu} + \frac{{f_{s_i}}^2}{\acute{\gamma}}} \leq f_{n_i}$<br>$f_{n_i} \geq 0$ |

In addition, each force $f_{c_i}$ in the friction cone $FC_{c_i}$ must be expressed in terms of the object's frame of reference. Then, it is convenient to select the contact coordinate frame $c_i$ in such way that the z-axis coincides with the direction of the inward contact normal and its origin lies at the location of the contact point expressed relative to the object frame $O$. The x and y axis of the contact frame may be selected arbitrarily so as to form a right-handed coordinate frame system. Fig. 3.1 illustrates this configuration.

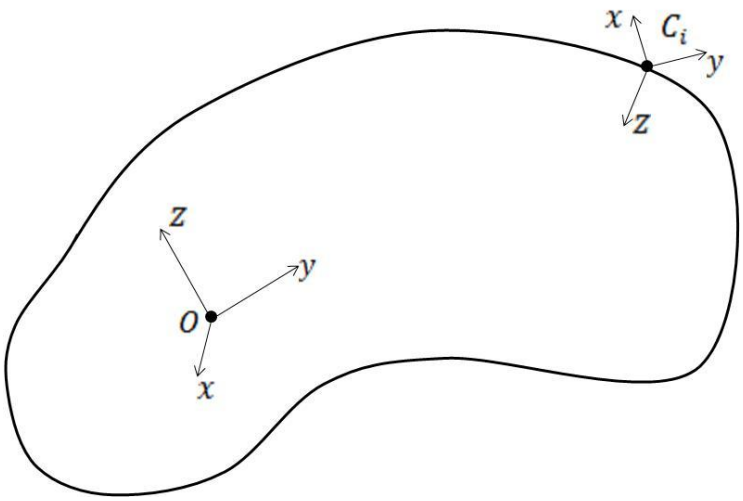


**Figure 3.1: Contact and Object Coordinate Frames.**

We can define the relative configuration of the contact coordinate frame as $(p_{oc_i},\ R_{oc_i})$, where $p_{oc_i}$ and $R_{oc_i}$ indicate position and orientation respectively. Then, the wrench produced by a contact expressed in terms of the object coordinates is given in the following statement:

$$w_{oc_i} = \begin{bmatrix} R_{oc_i} & 0 \\ \hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{bmatrix} B_{c_i} f_{c_i} \tag{3.3}$$

Where $\hat{p}_{oc_i}$ indicates a cross product matrix operation with the orientation matrix $R_{oc_i}$. Therefore, $w_{oc_i}$ is the corresponding contact wrench defined in object coordinates. Furthermore,

this entire transformation can be performed in a single step by joining both mapping operations into one. Such operation is defined by a matrix known as the grasp map:

$$G_i = \begin{bmatrix} R_{oc_i} & 0 \\ \hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{bmatrix} B_{c_i} \tag{3.4}$$

The grasp map allows making a simple and direct conversion of a contact force $f_{c_i} \in FC_{c_i}$ to the corresponding wrench $w_{oc_i}$in the space of generalized forces. For the sake of simplicity and clarity the $o$ subscript is disregarded and $w_{c_i}$ is used from here on to refer to the wrench in object coordinates generated by the $i^{th}$ contact. In addition, mapping forces to wrenches allows conducting a more independent analysis of the grasp. This implies that the properties of the grasp can be studied in terms of the forces that it can generate. Hence, it is not necessary to consider other aspects that present no influence on the grasp.

As described previously, each friction cone defines a set of forces that can be transmitted through the contact. Consequently, the set of forces in the friction cone can be correspondingly represented in the space of generalized forces by means of the grasp map. Therefore, the following sections describe each friction cone and outline the procedure to create an adequate approximation.

### 3.1.2 Frictionless Point Contact (FPC)

This is the simplest type of contact and can only exert a force in the direction of the inward normal vector at the contact. These types of contact do not occur in practice because friction is always present. But, they are useful to model and analyze grasps that should not rely on friction. By combining the wrench basis in table 3.1 with the contact's spatial configuration we can obtain the corresponding grasp map and wrench:

$$w_{c_i} = \begin{bmatrix} R_{c_i} & 0 \\ \hat{p}_{c_i} R_{c_i} & R_{c_i} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} f_{c_i} = \begin{bmatrix} n_i \\ p_{c_i} \times n_i \end{bmatrix} f_{c_i} = G_i f_{c_i} \quad f_{c_i} > 0 \tag{3.5}$$

where $n_i$ is the inward unit normal at the contact expressed in the object coordinate frame. The wrench $w_{c_i}$ corresponds to a single vector for the FPC model. Other friction models define an entire set rather that a single wrench vector. The geometry of the frictionless point contact model is given in Fig 3.2.

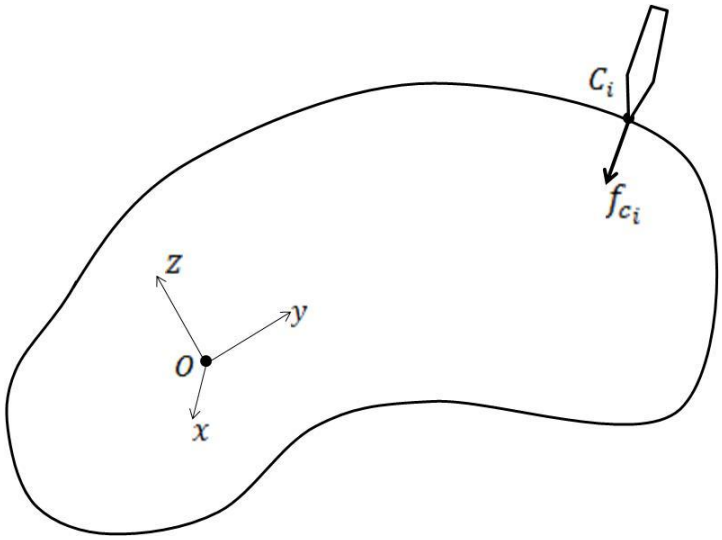


**Figure 3.2: Frictionless Point Contact.**

### 3.1.3 Point Contact with Friction (PF):

A point contact that supports friction can exert forces that are tangential to the surfaces in contact. The PF contact model may prevent slippage as long as frictional forces are not exceeded. The coulomb's friction model bounds the set of forces that lie within the friction cone. Therefore, the maximum attainable friction force $f_\tau$ depends on the normal force $f_n$ and the coefficient of friction $\mu$ as given by the following relation:

$$f_\tau \leq f_n \mu \tag{3.6}$$

In the 3-dimensional case the tangential force can be further decomposed as the sum of two perpendicular vectors $f_o$ and $f_t$ that lie in the tangent plane. Then the friction cone becomes:

$$FC_{c_i} = \{ f \in R^3 : \sqrt{{f_{o_i}}^2 + {f_{t_i}}^2} \leq \mu f_{n_i} , f_{n_i} \geq 0 \} \tag{3.7}$$

Hence all forces $f$ lie in the non-empty space defined by $FC_{c_i}$. In order to create the wrench $w_{c_i}$ that corresponds to the force $f_{c_i}$ other additional formalizations shall be made. Given a right-handed frame defined by the basis vectors $\{ n_i, o_i, t_i \}$, a force $f_{c_i}$ that lies in the friction cone $FC_{c_i}$ can be expressed as:

$$f_{c_i} = [f_{n_i} \; f_{o_i} \; f_{t_i}]^T \tag{3.8}$$

Each element in the vector $f_{c_i}$ represents the magnitude of the component along each basis vector of the contact coordinate frame $c_i$. Given this relation and the wrench basis in table 3.1, the wrench produced by a point contact with friction can be obtained as follows:

$$w_{c_i} = \begin{bmatrix} R_{c_i} & 0 \\ \hat{p}_{c_i} R_{c_i} & R_{c_i} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} f_{c_i} = G_i f_{c_i} \quad f_{c_i} \in FC_{c_i} \tag{3.9}$$

$$G_i = \begin{bmatrix} n_i & o_i & t_i \\ p_{c_i} \times n_i & p_{c_i} \times o_i & p_{c_i} \times t_i \end{bmatrix}, \quad G_i \in R^{6\times 3}$$

A geometric representation of the friction cone produced by the point contact with friction model is given in Fig. 3.3.

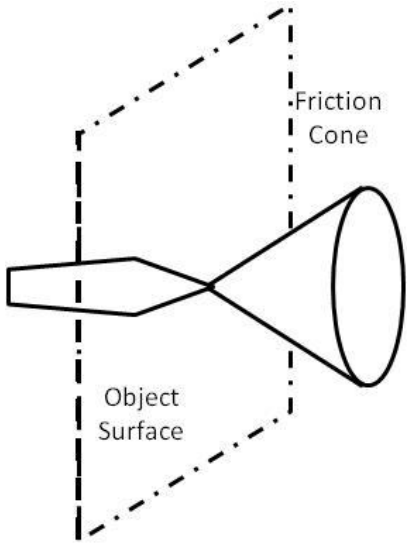


**Figure 3.3: Point Contact with Friction**

The nonlinear expression in equation 3.7 defines an infinite set of forces that reside within the bounds of the friction cone. Since it is impractical to deal with infinite numbers, a discrete approximation of the friction cone must be produced. Such approximation should have sufficient elements so as to preserve the properties of the original nonlinear model. In the case of the PF model, the friction cone is approximated with an m-sided pyramid [67] as follows:

$$f_{c_{ij}} = |f_{n_i}| \, [\, 1 \quad \mu_s \cos\left(\frac{2\pi j}{m}\right) \quad \mu_s \sin\left(\frac{2\pi j}{m}\right) \,]^T, \; j = 1 \ldots m \tag{3.10}$$

Therefore, the discrete counterpart of the friction cone contains m forces $f_{c_{ij}}$ distributed along the edges of an m-sided pyramid. A larger value of m produces a more accurate approximation but induces a larger computational cost. Each force in the cone can be used to produce the corresponding wrench. This results in a finite set of wrench vectors that represent contribution of the contact in the space of generalized forces. Such conversion may be carried out with the grasp map in equation 3.9 as well and it is shown below:

$$w_{c_{ij}} = G_i f_{c_{ij}}, \qquad f_{c_{ij}} \in FC_{c_i} \tag{3.11}$$

### 3.1.4 Soft Finger Contact Linear (SFCl)

In general, soft finger contacts provide a more realistic representation of contacts that induce slight deformations on the objects. These deformations generate a surface of contact that allows applying spin moments about the surface normal. The magnitude of the torques that can be applied without causing slippage is bounded by the coefficient of spin moment. In fact, two types of soft finger contact were suggested by Howe in [94]. This section discusses the linear variety (SFCl) and the next will describe the elliptical soft finger contact (SFCe).

The friction cone of the linear soft finger contact bounds tangential forces and spin moments according to coulomb's friction and the coefficient of spin moment. Hence, all forces that lie inside the friction cone abide to the following statement:

$$FC_{c_i} = \{ f \in R^4 : \sqrt{\frac{{f_{o_i}}^2 + {f_{t_i}}^2}{\mu}} + \frac{|f_{s_i}|}{\gamma} \leq f_{n_i},\ f_{n_i} \geq 0 \} \tag{3.12}$$

The coefficient of torsional friction is denoted by $\gamma$ while $f_{s_i}$ represents the spin moment about the surface normal $n_i$. By taking the notion of a force to a 4-dimensional space, it is possible to include the spin moment as an additional component of the force vector [32]. The resulting force vector has the following form:

$$f_{c_i} = [f_{n_i}\ f_{o_i}\ f_{t_i}\ f_{s_i}]^T \tag{3.13}$$

By combining the wrench basis in table 3.1 with the corresponding spatial transforms, the wrench vector can be found in the following manner:

$$w_{c_i} = \begin{bmatrix} R_{c_i} & 0 \\ \hat{p}_{c_i} R_{c_i} & R_{c_i} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i} = G_i f_{c_i},\quad f_{c_i} \in FC_{c_i} \tag{3.14}$$

$$G_i = \begin{bmatrix} n_i & o_i & t_i & 0 \\ p_{c_i} \times n_i & p_{c_i} \times o_i & p_{c_i} \times t_i & n_i \end{bmatrix}, \qquad G_i \in R^{6 \times 4}$$

The 3-dimensional projection of the friction cone results in a bicone as shown in Fig. 3.4.

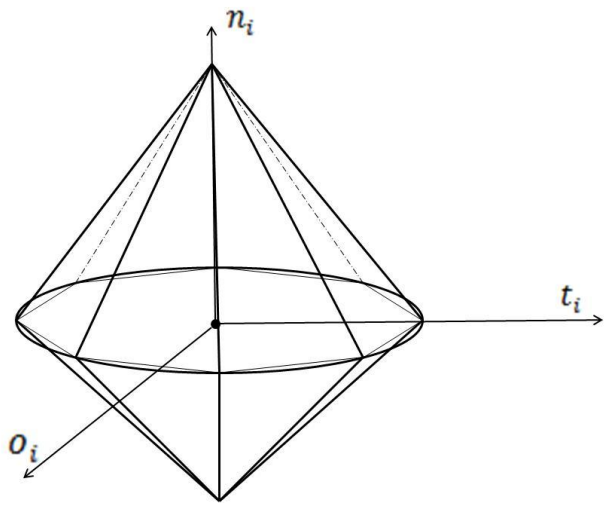


**Figure 3.4: Soft Finger Contact Linear.**

For many years, much of the analysis in robotic grasping only considered the FPC and PF models because they were relatively simpler to study as shown in [29], [37] and [71]. Some research efforts have tried to study grasps that were composed of soft finger contacts using very involved techniques [29, 30]. However, these techniques were applicable to a limited number of cases and did not provide a general treatment of the contact models. Eventually, the use of polyhedral approximation made it possible to generalize solutions to a number of grasping problems. However, polyhedral approximations were not available for soft finger contacts.

The spin moments that are induced by soft finger contacts gave the impression that a linearization for this contact model was unattainable. However, the work by Zheng [32] accounted for this limitation by expanding the notion of a force to a 4-dimensional space. Hence, the fourth component in the 4-dimensional vector was the magnitude of the spin moment about the contact normal. From that point on, the procedure for linearizing the soft finger friction cone was very similar to the one that had been used for the PF model. This procedure is demonstrated in the following equation:

$$f_{c_{ij}} = \|f_{n_i}\| \left[ 1 \quad \mu_s \cos\left(\frac{2\pi k}{K}\right) cos\left(\frac{q\pi}{2Q}\right) \quad \mu_s \sin\left(\frac{2\pi k}{K}\right) cos\left(\frac{q\pi}{2Q}\right) \quad \gamma sin\left(\frac{q\pi}{2Q}\right) \right]^T \quad (3.15)$$

$$k = 1, 2, \ldots, K \ \ (K \geq 3); \quad q = -1, 0, 1; \quad if\ q = \pm 1\ then\ k = 1$$

$$j = 1, 2, \ldots m \ \ (m = K + 2)$$

The approximation is improved by increasing the value of K but the computational cost is increased as well. Once again, the wrench vectors can be obtained with the use of the grasp map as it was shown in equation 3.11.

### 3.1.5 Soft Finger Contact Elliptical (SFCe)

This type soft finger contact is only slightly different that the linear case, but in general it imposes larger bounds on the allowable tangential forces and spin moments. The differences between the two models account for the degree of softness in the material that covers the finger. The SFCe contact is considerably softer than the SFCl contact and can apply uniform pressure distribution over a constant contact area [94]. The friction cone for this contact model is given by the next equation:

$$FC_{c_i} = \{ f \in R^4 : \sqrt{\frac{{f_{o_i}}^2 + {f_{t_i}}^2}{\mu} + \frac{{f_{s_i}}^2}{\acute{\gamma}}} \leq f_{n_i},\ f_{n_i} \geq 0 \} \quad (3.16)$$

In this case, $\acute{\gamma}$ indicates the coefficient of spin moment for the elliptical case. Similarly, $f_{s_i}$ corresponds to the magnitude of the spin moment about the normal vector $n_i$. The resulting contact force is a 4-dimensional vector that contains the normal, tangential and spin moment components as indicated by equation 3.13. Furthermore, the grasp map $G_i \in R^{6\times 4}$ is identical to the one presented in equation 3.14. The projection of the elliptical soft finger contact in the 3-dimensional space produces an ellipsoid as portrayed in Fig. 3.5.

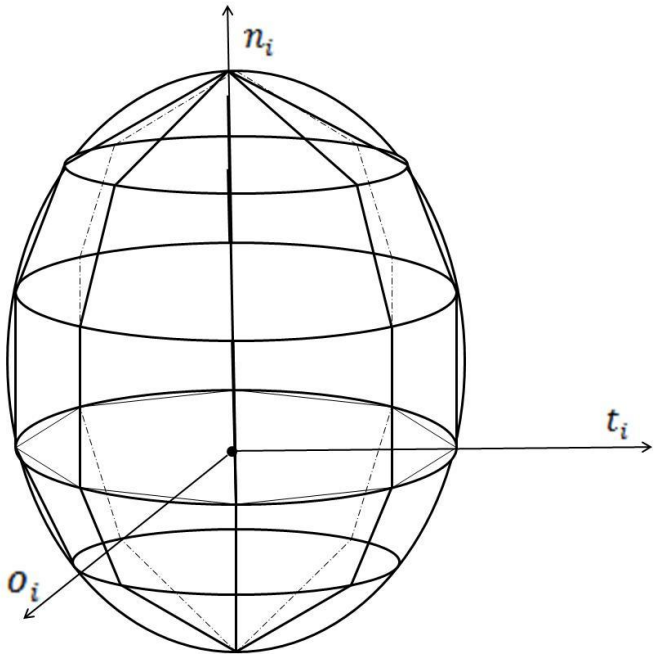


**Figure 3.5: Soft Finger Contact Elliptical.**

The friction cone approximation relies on the same expression that was used with the SFCl contact, but spans a larger set of indices. For the sake of clarity, the same equation with a few modifications is shown next:

$$f_{c_{ij}} = \|f_{n_i}\| [\, 1 \quad \mu_s \cos\left(\frac{2\pi k}{K}\right) cos\left(\frac{q\pi}{2Q}\right) \quad \mu_s \sin\left(\frac{2\pi k}{K}\right) cos\left(\frac{q\pi}{2Q}\right) \quad \acute{\gamma} sin\left(\frac{q\pi}{2Q}\right) ]^T \quad (3.17)$$

$$q = -Q, \dots, -1,0,1, \dots, Q \;\; (Q > 1); if\ q = \pm 1\ then\ k = 1$$

$$k = 1, 2, \dots, K \;\; (K \geq 3);$$

$$j = 1, 2, \dots m \;\; (m = 2KQ - K + 2)$$

In this case, a better approximation can be obtained by increasing the values of K and Q. Then, each force $f_{c_{ij}}$ can be mapped into its corresponding wrench $w_{c_{ij}}$ with the grasp map $G_i \in R^{6\times 4}$ from equation 3.14.

## 3.2 Wrench Space

The Wrench Space comprises the set of all wrenches that are generated by the contacts of a grasp. Forces expressed in 3-dimensional space yield a 6-dimensional wrench space. The wrench space is a useful construct that allows assessing the properties presented by a grasp. Hence, the properties of a grasp inferred from the wrench space are a measure of the force generation capabilities in relation to a given requirement.

### 3.2.1 Wrench Space Construction

The construction of this space typically involves creating a convex hull that bounds all the wrenches. However, two methods were suggested by Canny and Ferrari [37] in order to account for different properties of the grasp. The difference lies in the way that the magnitude of the normal contact forces is bounded. For convenience, the grasp force vector is used to represent the force that a grasp exerts on an object:

$$g = \begin{bmatrix} f_{n_1} \\ \vdots \\ f_{n_{1p}} \end{bmatrix} \tag{3.18}$$

The first modality for constructing the wrench space involves finding the grasp configuration whose contacts present a maximum individual resistance to external disturbances. In this case, the upper bound on $g$ is given by the maximum applied contact force. In order to consider the ratio of applied forces, it is necessary to scale the forces so as to make the maximum applied force equal to one. Then, the grasp force is set to 1 over a $L_\infty$ metric as follows:

$$\|g\| = \max\left( f_{n_1}, \dots, f_{n_{1p}} \right) = 1 \tag{3.19}$$

This results in a wrench space that contains all possible combinations of wrenches acting on the object. The Minkowski sum allows finding all possible wrenches by combining every

element of a set with every other element from other sets. Then, the convex hull operation forms a convex set that bounds all the wrenches, as demonstrated by the next equation:

$$W_{L_\infty} = ConvexHull(\bigoplus_{i=1}^{p} \{ w_{c_{i1}}, \dots, w_{c_{im}} \}) \quad (3.20)$$

A total of $m^p$ 6-dimensional points are produced by the Minkowsky operation. Hence, the convex hull operation must process that amount of points in order to compute the boundary of the convex set that bounds all points.

The second alternative imposes an upper bound on the sum of the magnitude of the forces at the contact points. This bound aims to consider the torque production capabilities of the hand. Hence, the grasp force $g$ is bounded over a $L_1$ metric so as to obtain the wrench space as indicated in the next expression:

$$W_{L_1} = ConvexHull(\bigcup_{i=1}^{p} \{ w_{c_{i1}}, \dots, w_{c_{i1}} \}) \quad (3.21)$$

This computation only needs to process $mn$ 6-dimensional points. In addition, it's a fact that $W_{L_\infty} \supseteq W_{L_1}$ because the set of points computed by the Minkowsky sum is a large enough to accommodate all the points that obtained by means of the convex hull operation of equation 3.21. The procedure for constructing the wrench space with a $L_1$ metric yields a more manageable computation since fewer points are processed. The implementation of the first method for $W_{L_\infty}$ would induce very large computational demands. Hence, this thesis uses the second method as it is done in most documented works.

### 3.2.2 Wrench Space Properties

By means of the wrench space, it is possible to make quantitative assessments of grasp properties that can be measured in terms of the force production capabilities. In addition, the geometric interpretation of the wrench space permits relating grasp properties to relevant

geometric characteristics that are highly influenced by the contact wrenches. Thus, the analysis of these characteristics leads to adequate measures of the grasp properties.

One of those properties is that of Force-Closure (FC), which indicates whether the grasp is capable of rejecting disturbance forces that affect the object. Such condition is achieved whenever the contact wrenches span the 6-dimensional space of generalized forces. In addition, a FC grasp can be assigned a measure of its ability to reject disturbances. This measure in known as the grasp quality and its definition may vary in relation to the tasks requirements.

Furthermore, the grasp equilibrium property entails that the wrenches acting on the object add up to zero. An associated problem is that of optimal force distribution. Also, dexterity relates to the ability of the hand to manipulate the object as prescribed by the tasks objectives. In the absence of a task, a grasp is considered dexterous when it can move the object in any direction. This thesis is mainly concerned with Force-Closure and grasp quality; hence they are described in more depth in the following sections.

## 3.3 Force-Closure Test

A grasp that has the ability to balance disturbance forces is said to be Force-Closure [38]. These disturbance forces may originate by a number of different factors such as gravity or collisions with obstacles. The presence of internal forces plays a critical role in dissipating disturbance forces. The set of internal forces is a subset of the contact forces that yields no net force on the object. Hence, a FC grasp may nullify the disrupting effect of external disturbances by redistributing the internal forces.

An equivalent definition is that the origin resides in the interior of the wrench space of a FC grasp [38]. Consequently, this definition has been used in the formulation of various

methods that determine when a grasp is Force-Closure. A number of these methods rely on optimization techniques that may incur in expensive computations due to the non-linearity of the contact models. However, an efficient Force-Closure test technique that relies on the ray-shooting test was presented by Zheng [35]. In addition, its implementation is simple and it has been incorporated in this thesis to perform FC tests. The main idea of this technique is described in the remaining paragraphs of this section.

For a given a grasp with $p_0$ FPC contacts, $p_f$ PF contacts and $p_s$ SFC contacts, the total number of contacts is $p = p_0 + p_f + p_s$. Each contact $c_i$ $(i = 1,2 \dots, p)$ generates a wrench set $w_i = \bigcup_{j=1}^{m} w_{ij}$ in accordance to the designated friction cone. Thus, the corresponding wrench space $W$ can be constructed as shown in equation 3.21. The objective is to determine that the origin $O$ lies in the interior of the convex set $W$ [35].

The convexity of set $W$ assures that its centroid $w_c$ lies in the interior of $W$. The centroid can be obtained as shown in the next statement:

$$w_c = \frac{1}{p+1} \sum_{i=1}^{p} \left[ n_i^T \quad \left(p_{c_i} \times n_i\right)^T \right]^T = \frac{1}{p+1} \sum_{i=1}^{p} w_{n_i} \tag{3.22}$$

Where $w_{n_i} = \left[ n_i^T \quad \left(p_{c_i} \times n_i\right)^T \right]^T$ and it represents the normal contact wrench generated by the contact. A point $z(-w_c,\ W)$ represents the intersection of the boundary $BD(W)$ with the ray $R(-w_c)$. The ray $R(-w_c)$ is a line segment that starts at $w_c$ and moves towards $O$. Thus, Force-Closure is achieved when the following condition is met:

$$\overline{w_c O} < \overline{w_c z} \tag{3.23}$$

The intersection test between the ray $R(-w_c)$ and $BD(W)$ is conducted with the ray-shooting algorithm [33]. Furthermore, the ray-shooting algorithm relies heavily on the GJK distance algorithm [11]. The explanations of these algorithms are omitted so as to avoid

deviating from the objectives of this chapter. However, they will be described in chapter 4. An illustration of the FC test in a hypothetical 2-Dimensional space is shown in Fig. 3.6.

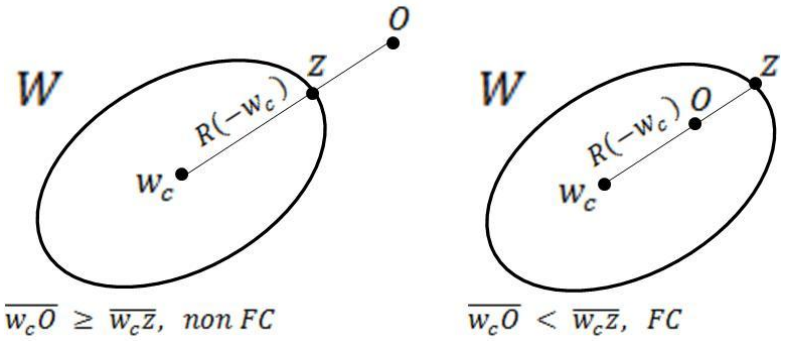


**Figure 3.6: Force-Closure (FC) Test.**

As a remark, the method for Force-Closure test presented here differs from [35] in certain aspects. In [35], $W$ is created by performing the convex hull operation on the union of the origin $O$ and the wrench sets $w_i$. Thus, that approach requires an additional convex hull computation; these computations can be rather costly when there are too many elements. The procedure shown here only needs a simple segment length comparison and so it overcomes this inconvenience. An implementation of this FC test has been incorporated as part of the Robotic Grasp Toolkit, which will be discussed in chapter 6.

## 3.4 Grasp Quality Evaluation

The grasp quality is a measure of the effectiveness of the grasp in relation to some designated criteria. These criteria can be expressed in terms of a required task. In that case, the grasp quality is said to be task-oriented. In the absence of a task, the grasp quality reflects the general ability to reject a disturbance force that may come from any direction. Such metric is

known as the Worst-Case Quality metric. In the space of generalized forces, this metric is equivalent to the radius of the largest hyper-sphere, centered at the origin, which is contained in its entirety within the grasp wrench space [37]. Consequently, this metric is also known as the Largest-Ball criterion.

The intuitiveness of the Largest-Ball metric definition allows computing the corresponding grasp quality by means of a procedure that relies on the computational package Qhull [49]. This package can perform convex hull operations on a cloud of points for any number of dimensions up to nine. The output is a faceted representation of the boundary of the convex hull. Hence, the actual result produced by Qhull is an integer array that indexes vertices and indicates facet connectivity. This array is sufficient to recreate the convex hull in its entirety. Additionally, the normal vectors and offset values that define a facet's hyperplane are returned in the output as well. So, the radius of the largest ball $e$ centered at the origin $O$ is equal to the offset value of the closest hyper-plane $H_c$ [46]. This idea is illustrated in the context of a hypothetical 2-dimensional wrench space in Fig. 3.7.

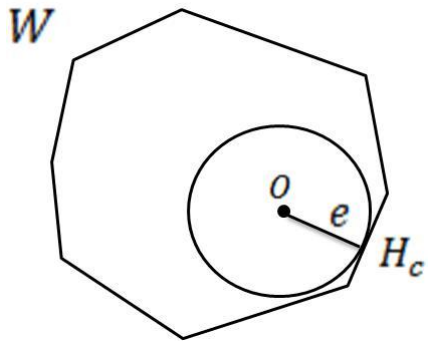


**Figure 3.7: Largest-Ball metric.**

This grasp quality algorithm as well as the Qhull library have been implemented and made part of the toolkit presented in this thesis.

# CHAPTER 4: COMPUTATIONAL GEOMETRY

The field of Computational Geometry is the study of algorithmic solutions to problems that can be stated in terms of geometry. The growth of this field was motivated by the need to utilize the full potential of emerging technologies in the areas of computer graphics and computer hardware. This field of study has spawned a multitude of sophisticated algorithms and data structures for problems such as convex hull computation, distance queries, interference detection and linear programming. In general, these algorithms rely very strongly on mathematical approaches so as to provide computationally efficient solutions for geometric problems.

Computational Geometry methods are applicable to a number of areas that include robotic motion path planning, computer-aided engineering, geometric modeling and collision detection. Furthermore, there is evidence of the use of Computational Geometry methods in formulating solutions to grasping problems [67], [69, [78]. In most cases, they are used to handle the geometric representation of the 6-dimensional wrench space. Additionally, some of them have been used to determine contacts between hands and object models that come into contact [67].

The work in this thesis employs computational geometry methods to perform Force-Closure and grasp quality calculations as indicated in chapter 3. Also, hyper-planes are frequently utilized by the grasp synthesis technique which is the focus of this thesis and is presented in chapter 5. Thus, the aim of this chapter is to provide sufficient understanding of the computational geometry techniques that have been incorporated in the work presented here. Moreover, a discussion of continuous collision detection (CCD) is given with the aim of exhibiting its relevance in finding valid finger configuration candidates. Our own

implementation of a recursive CCD technique for articulated bodies is presented in this chapter as well.

## 4.1 Gilbert Johnson Keerthi (GJK) Distance Algorithm

This algorithm allows computing the minimum separating distance between two disjoint convex objects. Furthermore, it can accommodate any convex shape and its implementation is relatively simple. Its generalized definition makes it applicable to objects expressed in an N-dimensional space. In the work presented here, it is an essential part of the ray-shooting algorithm that is used in the Force-Closure test method described in chapter 3. The version of GJK depicted here is based on the work presented in [11]. The following formalizations and remarks need to be made before GJK is described in full depth.

### 4.1.1 Configuration Space Obstacle

The relative spatial configuration of two convex objects can be expressed in more implicit terms by means of the configuration space obstacle (CSO). Solutions to the majority of proximity queries can be obtained by directly interrogating the CSO rather than each individual object. For two non-intersecting convex objects A and B, the CSO is a convex set which does not contain the origin in its interior. The mathematical definition of the CSO is the Minkowsky sum $A + (-B)$ of the convex sets A and B [11]. This operation sweeps object B through every point in object A and the results is a convex set as well.

### 4.1.2 Support Mapping

The GJK algorithm's versatility is due to its reliance on support mapping. Support mapping provides an implicit description of the underlying geometry of an object. It does not require that the geometry be organized according to a particular topologic arrangement. The

result of a support mapping operation on an object is a support point. Hence, it is formally defined as function $s_A$ that maps a vector $v$ to a point of $A$ in the following way:

$$s_A(v) \in A \mid \ v \cdot s_A(v) = \ max\{v \cdot x : x \ \in A\} \tag{4.1}$$

Equivalently, the point $s_A(v)$ equals the largest projection of vector $v$ onto set $A$. The support point may not be unique since several points $x \ \in A$ could yield an identical value for the dot product with $v$. Our implementation of support mapping relies on a simple search for the largest dot product.

### 4.1.3 Johnson Distance Algorithm

Furthermore, GJK resorts on the Johnson's Distance algorithm at various instances. The Johnson's Distance algorithm computes the distance from the origin 0 to the closest point v of a simplex. A simplex is the generalization of a triangle or tetrahedron for the 2-dimensional and 3-dimensional space respectively. In the context of the Johnson Distance algorithm, Y refers to the set of vertices of the simplex $\{y_1, \dots, y_m\}$. Hence, the point v of the simplex Y that is closest to the origin can be expressed as a convex combination of the elements in Y in the following way:

$$v = \ \sum_{i=1}^{n} \lambda_i y_i, \quad \sum_{i=1}^{n} \lambda_i = 1, \quad \lambda_i \geq 0 \tag{4.2}$$

Given the expression above, the objective of the Johnson's Distance algorithm can be restated as finding the smallest set $X \subseteq Y$ such that v is a convex combination of $X = \{y_i : \lambda_i > 0\}$. A further observation indicates that v is closest to the origin if and only if it is perpendicular to $X = \{x_1, \dots, x_m\}$. This remark implies that $(x_i - x_1) \cdot v = 0$ for $i = 2, \dots, m$. Hence, these conditions are used to form the following system of equations:

$$A[\lambda_1, \dots, \lambda_m]^T = b \tag{4.3}$$

$$A = \begin{bmatrix} 1 & \cdots & 1 \\ (x_2 - x_1) \cdot x_1 & & (x_2 - x_1) \cdot x_m \\ \vdots & \ddots & \vdots \\ (x_m - x_1) \cdot x_1 & \cdots & (x_m - x_1) \cdot x_m \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Nevertheless, this description of the Johnson's Distance only works well for cases when we need to find the smallest distance between the origin O and the simplex. It does not resolve situations in which the distance from an arbitrary point needs to be found. Such situation is encountered during the ray-shooting computations performed in every Force-Closure test as shown in chapter 3. Our version of the distance algorithm accounts for this downfall by making necessary adjustments to equation 4.3. Hence, the point v in simplex $X \subseteq Y$ is closest to an arbitrary point p when $(x_i - x_1) \cdot (v - p) = 0$ for $i = 2, \dots, m$. This condition along with expression 4.2 yields the system of equations presented below:

$$A[\lambda_1, \dots, \lambda_m]^T = b \tag{4.4}$$

$$A = \begin{bmatrix} 1 & \cdots & 1 \\ (x_2 - x_1) \cdot x_1 & & (x_2 - x_1) \cdot x_m \\ \vdots & \ddots & \vdots \\ (x_m - x_1) \cdot x_1 & \cdots & (x_m - x_1) \cdot x_m \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ (x_2 - x_1) \cdot p \\ \vdots \\ (x_m - x_1) \cdot p \end{bmatrix}$$

Since the set $X = \{y_i : \lambda_i > 0\}$ is not known a priori, our implementation first solves equation 4.4 using the elements of set Y. Then, each element $y_i$ whose corresponding $\lambda_i \leq 0$ is eliminated and a new set is formed with the remaining elements. The system of equations in 4.4 is solved once again for the new set. This procedure is repeated until all conditions in expression 4.2 are met. Thus, the Johnson's Distance algorithm returns the smallest set $X \subseteq Y$ and the closest point v.

### 4.1.4 GJK Algorithm Description

The GJK is approximates the closest point $v(C)$, in the set $C = A + (-B)$, to the origin by means of a descent search; where A and B are convex objects [11]. The set $C$ is the CSO of the convex sets A and B, which implies that $C$ is also a convex set. The objective of the GJK algorithm can be formally declared as in the next statement:

$$d(C) = \|v(C)\|, \ \|v(C)\| = \min\{\|x\| : x \in C\} \tag{4.5}$$

However, this statement can be redefined so as to consider the closest point $v(p, C)$ in set $C$ to an arbitrary point $p$ that lies outside set $C$. It is important to point out that this particular case may not have any significance in the context two objects A and B that are disjoint in space. But, this adjustment makes GJK applicable to the ray-shooting algorithm. As mentioned previously, it is in our interest to carry out ray-shooting computations; thus this particular variety of GJK will be discussed here. Furthermore, the details of GJK are described in terms of set $C$ so as to omit its association with the CSO of two objects A and B. Hence, the corresponding modifications to equation 4.4 lead to the following expression:

$$d(p, C) = \|p - v(p, C)\|, \ \|p - v(p, C)\| = \min\{\|p - x\| : x \in C\} \tag{4.6}$$

The term $d(p, C)$ in equation 4.6 represents the shortest distance from set $C$ to point $p$. Having set forth the objectives, we can proceed to describe GJK.

During each iteration, the procedure constructs a simplex that is fully contained in $C$. The simplex can have one to $N + 1$ vertices and may represent a point, a line, a hyper-plane or an N-simplex. Then, a new set $W_k$ is defined as the set of vertices corresponding to the simplex constructed during the $k^{th}$ iteration. In addition, the point $v_k \in W_k$ refers to the point closest to p. It can also be observed that $\|p - v_k\| \geq d(p, C) = \|p - v(p, C)\|$, hence $v_k$ imposes an upper bound on the closest point $v(C)$.

The next step computes a support point $w_k = s_C(p - v_k)$ that belongs to $C$. The search for the closest point finishes when a termination criterion is met. Such criterion is given next:

$$\|p - v_k\| - \frac{(p - v_k) \cdot w_k}{\|p - v_k\|} \leq \varepsilon_{abs} \tag{4.7}$$

Where the tolerance $\varepsilon_{abs}$ is a user defined value that accounts for numeric errors which arise during floating point arithmetic operations. If this condition cannot be fulfilled then the point $w_k$ is added to the simplex $W_k$. The simplex $W_k$ is passed to the Johnson's Distance algorithm in order to compute a new simplex and closest point. Thus, $W_k$ and $v_k$ are set equal to the corresponding point and simplex that were returned by the Johnson's Distance technique. The value of k is increased by one and the GJK algorithm proceeds to the next iteration.

The key of this process lies in creating a simplex that is closer to the point p than the simplex from the previous iteration. The support point $w_k = s_C(p - v_k)$ represents the closest vertex to the point p in the direction of $p - v_k$. Thus, the algorithm continues as long as $w_k$ is not the closest vertex in $C$. The condition in 4.5 is met when $w_k$ becomes the closest vertex, at which point the condition in 4.7 is met and the search concludes. Fig. 4.1 depicts a simplistic version of the GJK in the context of a 2-dimensional space. Furthermore, the main steps of the GJK algorithm are presented in Alg. 4.1.

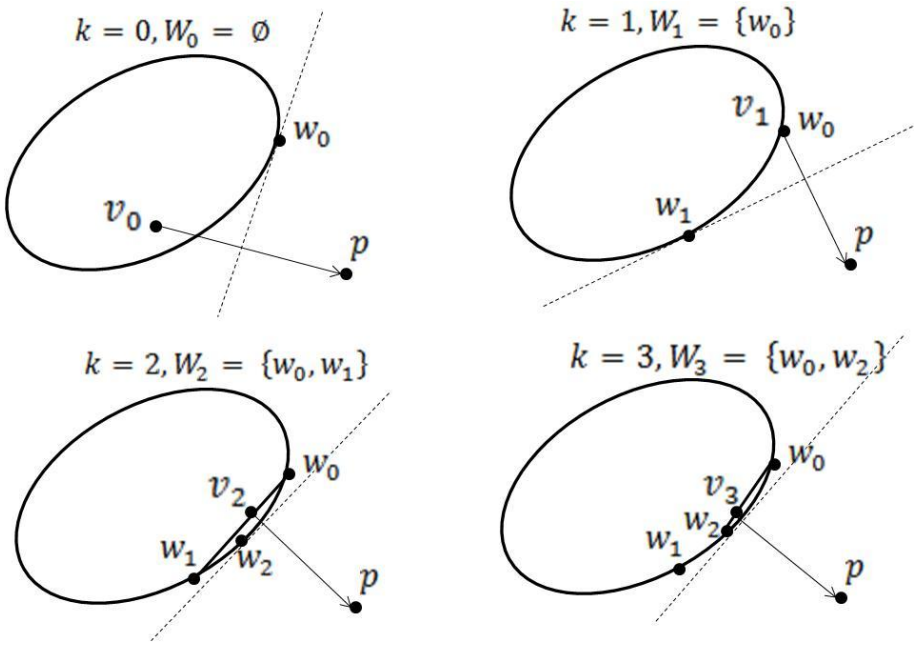


**Figure 4.1: GJK Algorithm.**

Algorithm 4.1 *GJKDistance*

**Inputs:**

$C$: Convex Set

$p$: Point outside of set $C$.

**Outputs:**

$v_k$: Closest point in set $C$ to point $p$.

**Begin:**

$k \equiv 0$;

$v_k \equiv$ Arbitrary point in $C$;

$W_k \equiv \emptyset$;

$w_k \equiv s_C(p - v_k)$;

**While** $\|p - v_k\| - \frac{(p - v_k) \cdot w_k}{\|p - v_k\|} > \varepsilon_{abs}$ **iterate**

$k \equiv k + 1$;

$W_k \equiv W_k \cup w_k$ ;

$v_k \equiv$ *JohnsonDistance*$(p, W_k)$;

$w_k \equiv s_C(p - v_k)$;

**End while**

**Return** $\|p - v_k\|, v_k$

As a final remark, the implementation of GJK presented here is based on [11] but it is general enough to accommodate objects described in N-dimensional space. The documented implementations of GJK presented in [7], [8], [9], [10] and [11] can process objects in 2 and 3 dimensions only. Instead, they contain a number of optimizations to handle very specific cases and perform considerably faster. However, the flexibility of our implementation allows handling the 6-dimensional wrench space, thus making it appropriate for our purposes.

## 4.2 Ray-Shooting Algorithm

The ray-shooting method finds the point $z(u, C)$ at which a ray $R(u)$ intersects the boundary of a convex set $C$ [35]. The ray $R(u)$ originates from an arbitrary point $v_c$ in the interior of $C$ and moves along the direction of a unit vector $u$. Also, the GJK algorithm is relied upon multiple times in order to compute the closets point between set $C$ and an arbitrary point outside $C$. In regards to robotic grasping, this method allows evaluating the Force-Closure property of a grasp very efficiently. As explained in chapter 3, the ray-shooting algorithm is used to detect that the origin lies in the interior of the wrench space. Thus, a grasp attains Force-Closure whenever its wrench space contains the origin in its interior.

At a given iteration of the ray-shooting algorithm, the closest point $v(b_k, C)$, in the boundary of $C$, to a point $b_k \notin C$ is found with the GJK algorithm. Moreover, the point $b_k$ lies in the region of the ray $R(u)$ that escapes the interior of the set $C$. Then, a hyper-plane $H_{k+1}$, with normal $b_k - v(b_k, C)$ and containing $v(b_k, C)$, intersects ray $R(u)$ at the point $b_{k+1}$. Hence, the relationship between $\mathrm{b_k}$ and $\mathrm{b_{k+1}}$ is established by the following equation:

$$\mathrm{b_{k+1}} = \mathrm{b_k} - \frac{\mathrm{d(b_k, C)^2}}{\mathrm{(b_k - v(b_k, C))\cdot u}}\mathrm{u} \tag{4.8}$$

In this equation, $d(b_k, A)$ refers to the shortest distance between $b_k$ and set $C$. Thus, the point $b_{k+1}$ is closer to $C$ than $b_k$ and the distance $d(b_k, A)$ is reduced with each iteration. The algorithm finishes when the distance $d(b_k, A)$ is below a user defined threshold; at which point $z(u, C) \cong v(b_k, C)$. A suitable choice for the initial value $b_0$ is $\|s_C(u)\|u$; which is the intersection point between ray $R(u)$ and the hyper-plane with normal $u$ that passes through the support point $s_C(u)$. An illustration of the ray-shooting algorithm for the 2-dimensional case is given in Fig. 4.2. Also, the corresponding pseudo code is shown in Alg. 4.2.

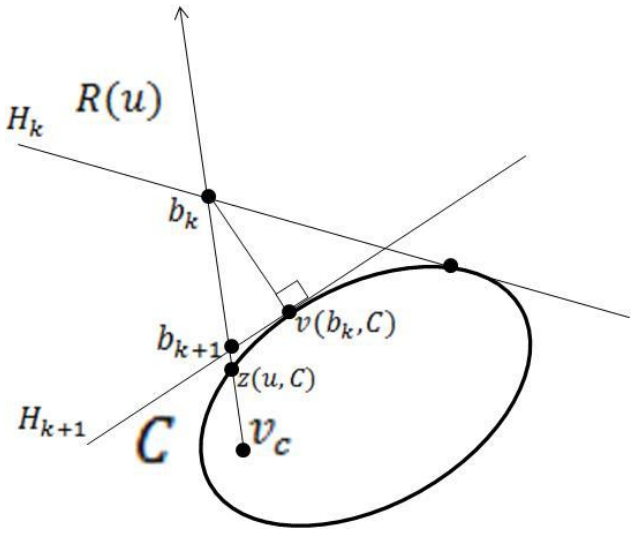


**Figure 4.2: Ray-Shooting Algorithm.**

In summary, the ray-shooting algorithm returns the point $z(u, C)$ at which the ray $R(u)$ and the boundary of set $C$ intersect. In the Force-Closure test, the set $C$ is analogous to the wrench space $W$ and the ray $R(-w_c)$ emanates from the centroid $w_c$ of the wrench space to the origin $O$. Thus, the resulting point $z(-w_c, W)$ corresponds to the location where the ray $R(-w_c)$ intersects the boundary of the wrench space $W$.

Algorithm 4.2 *Ray-Shooting Algorithm*

**Inputs:**

| | |
|---|---|
| $C$: | Convex Set |
| $v_c$: | Point in the interior of $C$ from which $R(u)$ emanates. |
| $u$: | Unit vector in the direction of $R(u)$. |

**Outputs:**

| | |
|---|---|
| $z(u, C)$: | Point at which ray $R(u)$ intersects set $C$. |

**Begin:**

$k \equiv 0$;

$b_0 \equiv \|s_C(u)\|u$;

$d(b_0, C) \equiv GJKDistance(b_0, C)$;

**While** $d(b_k, C) > \varepsilon_{abs}$ **iterate**

$b_{k+1} = b_k - \frac{d(b_k,C)^2}{(b_k - v(b_k,C))\cdot u} u$

$k \equiv k + 1$;

$d(b_k, C) \equiv GJKDistance(b_k, C)$;

**End while**

$z(u, C) \equiv v(b_k, C)$;

**Return** $z(u, C)$;

## 4.3 Hyperplanes

A hyperplane is the generalization of a line and flat plane in two and three dimensions respectively. The main way to define a hyperplane is with an offset and a normal vector; however there exists other alternatives for their construction. Furthermore, hyperplanes can divide a space into two half spaces. This particular trait allows them to be used as decision boundaries. The positive half space is given by the direction of the normal vector while the negative half space lies in the opposite side of the plane; the dividing plane region is considered neutral.

In this thesis, hyperplanes are employed to select the wrench sets that can increase the size of the wrench space in the weakest region. The idea of using hyperplanes in grasp synthesis

has been explored by Roa in [77] and [78]. However, the work in here uses hyperplanes in conjunction with support mapping so as to perform a single classification test for multiple points within a set. The details of this procedure along with the description of our grasp synthesis method are provided in chapter 5.

The mechanism to determine the side of the hyperplane associated with the location of a point is relatively simple. A hyperplane H with normal vector $a = [a_1, a_2, \dots, a_n]$ and offset d can be defined by a single linear expression in the following form:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n + b \tag{4.9}$$

A point $x = [x_1, x_2, \dots, x_n]$ that makes equation 4.9 evaluate to zero lies on the hyperplane H. Consequently, a point x belongs to the positive half space $H^+$ whenever the hyperplane equation yields a positive value. If a negative value is obtained then x resides in the negative half space $H^-$ of the hyperplane.

## 4.4 Convex Hull

The computation of the convex hull is a very common problem in computational geometry. It consists of finding the minimal convex set that envelops all the points of a given set [49]. In general, the convex hull refers to the convex polytope that constitutes the boundary of the hull. Hence, it is common practice to describe the convex hull as a collection of facets arranged in an adjacency list. Also, a facet is bounded by ridges; where each ridge indicates adjacency with another facet.

The work in [49] presents the Quick Hull algorithm, which is used to obtain the convex hull of geometries described in N-dimensional space. Thus, the Quick Hull algorithm has been implemented into a computational package known as Qhull. The Qhull package performs

convex hull computations very robustly and it has been employed in various research efforts that investigate robotic grasping as shown in [67], [77] and [78].

The work in this thesis utilizes Qhull in order to compute the convex hull of the wrench space. Consequently, the wrench space is represented by a list of facets, normal vectors, offset distance, neighboring facets and vertices. This representation benefits grasping methods that observe geometric characteristics of the wrench space such as the Force-Closure test. In addition, the grasp quality value can be easily obtained by extracting the smallest value of the offset list as it was done in [67].

## 4.5 Continuous Collision Detection

Continuous collision detection (CCD) focuses on determining interferences between two objects in motion. CCD can report the time of contact (TOC) whenever a collision takes place during the motion. Some recent works in robotic grasping explore the use of CCD towards the development of grasp planning and grasp synthesis methods [68], [69]. The work in this thesis also relies on CCD in order to determine finger configuration candidates that may lead to reliable grasps. Here, the term finger configuration candidate refers to a finger pose that makes contact with the object in one or more locations.

In recent years, CCD has received a great deal of attention due to its ability to prevent collision during high speed movement of bodies as shown in [83], [84], [85] and [86]. The work in this thesis focuses on the CCD technique presented in [2] due to its robustness and because it is available as a computational library for the C++ programming language. The CCD technique in [2] performs very fast TOC computations on non-convex triangulated models composed of thousands of triangles. Furthermore, it can process triangulated models without requiring a

specific geometric arrangement or shape. These features have been found to be very compatible with the objectives outlined throughout this thesis and so it has been selected for conducting CCD computations.

### 4.5.1 Recursive CCD Search Technique for articulated bodies

As pointed out earlier, the work in this thesis incorporates CCD into a procedure that detects contacts between a finger and an object. This procedure conducts a recursive search for collisions between individual links and the object. It begins the search with the first link from the base and continues until the last link is reached. A result is reported in the form of a finger configuration candidate whenever all link configurations are valid. Each finger configuration candidate is associated with a unique set of contacts and joint values that define the configuration. The greatest advantage of this recursive search procedure is that it takes the kinematic constraints of the fingers into consideration. Thus, every contact that was found is reachable by a given finger in the hand.

In addition, this recursive CCD search method has been influenced by the work presented in [68]. One of the most significant differences is that the method presented here does not rely on swept volumes. The inclusion of swept volumes in [68] allows exploring the entire work space of a finger. Hence, the results obtained with the method described here are a subset of those that can be computed with the method in [68]. Nevertheless, the recursive CCD search method produces a sufficiently large solution set from which reliable grasps can be found. In addition, it can accommodate fingers composed of links with multiple degrees of freedom (DOF) while the method in [68] has only been shown to process links with a single DOF. Thus, finger models with any kinematic configuration can be searched by this method thoroughly.

The recursive CCD search method consists of applying CCD to each link of the finger through a predefined motion range. Then, CCD returns one of three possible results: Interference at Start, Collision Free and Time of Contact (TOC). The Interference at Start condition is undesired and it stops the search from continuing to look for collisions of further links. The TOC condition indicates that at least one contact was found between the current link and the object. So, the current motion range and the contacts are saved whenever TOC is reported. Moreover, Collision Free indicates that no contact points were found. In this case, the current motion range is also saved in the event that further links report TOC. Either TOC or Collision Free allows the search to proceed onto the next link. Thus all links are tested though the entire range of motion. In addition, results are discarded if their corresponding contacts are deemed unreachable due to the presence of obstacles in the environment. Each set of reachable contacts and their associated joint values at TOC are saved to a finger configuration candidate. Thus, a single recursive CCD search may yield several finger configuration candidates that contain one or several reachable contacts.

In this work, the implementation of the recursive CCD search algorithm requires the following user inputs: The finger model, the environment, the object and the joint configuration range. The joint configuration range is provided as an array of initial and final values corresponding to every joint's range of motion. The environment is an optional input and may contain one or more obstacle that could interfere with the links. Hence, the recursive CCD search initiates by calling a subroutine that has been denominated as *findContactPoints*.

The *findContactPoints* subroutine represents the core of the CCD search and it can call another instance of itself when a number of conditions are met. Thus, *findContactPoints* will continue to make recursive calls until there are no more reachable contacts to be found. A

simple but effective flagging mechanism resolves situations in which the search must decide whether it continues or stops. This flagging mechanism relies on three flags that correspond to the three possible collision situations described earlier. These flags are: *InterferenceAtStart*, *CollisionFree* and *TOCFound*.

Each individual CCD test between a single link and the object is handled by the *CCDTest* routine. Thus, *CCDTest* takes the motion range of the link and the object as inputs. The object is assumed to be stationary which means that its spatial configuration is the same throughout the test. If a collision is encounter, it returns the time of contact (TOC) that corresponds to a value between 0 and 1. A collision flag, indicating one of the three collision situations, is also returned. Furthermore, the *CCDTest* routine calls the CCD methods for polyhedral models provided in computational library presented in [2]. In this manner, it is possible to isolate the intrinsic details of continuous collision detection from the recursive search performed here. The pseudo code corresponding to the implementation of the full recursive CCD search method is described in Alg. 4.3.

All the finger configuration candidates found by the recursive CCD search will be examined in order to assess their individual contribution to the formation of a grasp. This analysis examines each of their contacts and will be explained in detail in chapter 5; where we introduce our novel technique for grasps synthesis.

Algorithm 4.3 *RecursiveCCDSearch*

**Inputs:**

F: Finger Model with $I$ links $l_i$ where $i = 1, \dots, I$

$[\theta_{ij,start}, \theta_{ij,end}]$: Joint Motion Range for each joint $j = 1, \dots, DOF_i$ at the link $l_i$

$ENV$: Obstacle Environment with spatial configuration ${}^{W}_{Env}T$ in world coordinates.

$OBJ$: Object with spatial configuration ${}^{W}_{Obj}T$ in world coordinates.

**Outputs:**

$FConf_r$: Array of Finger Configuration Candidates where $r = 1, \dots, R$

**Begin:**

$i = 1;\ j = 1;\ r = 0;$ Initialize variables

*findContactPoints*(1,1) Call *findContactPoints* subroutine

**Subroutine** *findContactPoints*(i, j)

$\theta_{ij,start} \implies {}^{W}_{l_i}T_{start}$ Compute spatial configuration for link $l_i$ at start

$\theta_{ij,end} \implies {}^{W}_{l_i}T_{end}$ Compute spatial configuration for link $l_i$ at end

[CollisionFlag, $\text{TOC}_{env}$] $\Leftarrow$ *CCDTest*($l_i, ENV, {}^{W}_{l_i}T_{start}, {}^{W}_{l_i}T_{end}, {}^{W}_{Env}T$) Perform CCD test between $l_i$ and $ENV$

**If** CollisionFlag is *InterferenceAtStart* **then**

**Return** CollisionFlag;

**Else if** CollisionFlag is *CollisionFree* **then**

[CollisionFlag, $\text{TOC}_{obj}$] $\Leftarrow$ *CCDTest*($l_i, OBJ, {}^{W}_{l_i}T_{start}, {}^{W}_{l_i}T_{end}, {}^{W}_{Obj}T$) CCD test between $l_i$ and $OBJ$

**Else if** CollisionFlag is *TOCFound* **then**

[CollisionFlag, $\text{TOC}_{obj}$] $\Leftarrow$ *CCDTest*($l_i, OBJ, {}^{W}_{l_i}T_{start}, {}^{W}_{l_i}T_{end}, {}^{W}_{Obj}T$)

**If** $\text{TOC}_{obj} > \text{TOC}_{env}$ **then** link collides with $ENV$ first

CollisioFlag = *CollisionFree*; further links may register collisions

$\theta_{ij,end} = \theta_{ij,TOC} \Leftarrow TOC_{Env}$ Compute joint value at $TOC_{Env}$

**End if**

**End if**

This stage can only be reached if the environment does not interfere

**If** CollisionFlag is *InterferenceAtStart* **then** $l_i$ and $OBJ$ are at interference

**Return** CollisionFlag;

**Else if** CollisionFlag is *CollisionFree* **then** $l_i$ and $OBJ$ do not collide

Do nothing and proceed;

**Else if** CollisionFlag is *TOCFound* **then**

$\theta_{ij,end} = \theta_{ij,TOC} \Leftarrow TOC_{Obj}$ Compute joint value at $TOC_{Obj}$

**End if**

Iterate through the entire motion range

**For all** $\theta_{ij,k} = \{\theta_{ij,end}, \dots, \theta_{ij,start}\}$, $k = 1, \dots, Steps_j$

$\theta_{ij,k} \Rightarrow {}^{W}_{l_i}T_k \implies l_i$ Apply spatial configuration ${}^{W}_{l_i}T_k$ to current link $l_i$

**If** $j < DOF_j$ **then**

CollisionFlagNext = *findContactPoints*$(i, j+1)$

**Else**

**If** $i < I$ **then**

CollisionFlagNext = *findContactPoints*$(i+1, 1)$

**Else**

Break loop when the last joint in the last link is reached

**End if**

**End if**

**End for**

**If** CollisionFlag is *TOCFond* & CollisionFlagNext is *CollisionFree* **then**

$r = r + 1;$ Increase number of results found

Store all $\theta_{ij,TOC}$ and contacts into $FConf_r$

**End if**

**End** *findContactPoints*

## CHAPTER 5: A NOVEL TECHNIQUE FOR EFFECTIVE GRASP SYNTHESIS

In the context of robotic grasping, the problem of grasp synthesis constitutes the search for an adequate grasp that restrains an object in relation to a desired requirement. Furthermore, the superior dexterity of robotic hands is also one their greatest downfalls since their numerous degrees of freedom induce an unmanageable number of grasp possibilities. Thus, the main difficulty in grasp synthesis lies in selecting a fitting grasp from an extremely large configuration space. Consequently, the grasp synthesis problem is at the core of robotic grasping research, and as such it has hoarded a great deal of attention from many investigators in the field; as demonstrated in [50], [55], [65], [67], [68], and [78].

The aim of this chapter is to introduce a novel technique for synthesizing reliable Force-Closure grasps applicable to robotic hand models. This technique represents the most important contribution of the work presented in this thesis and it puts together two methodologies that are commonly applied towards the development of grasp synthesis methods. These methodologies were discussed in chapter 2 and they are: Hand-Based and Contact-Based grasps synthesis. Hand-Based grasp synthesis considers the physical characteristics of the hand models while Contact-Based grasp synthesis focuses on conforming to the characteristics of the object. Only a few documented efforts have considered a combination of Hand-Based and Contact-Based approaches in their investigation as shown in [67], [68], and [69].

The technique presented in this thesis relies on an iterative approach that finds combinations of finger postures such that their corresponding contacts contribute in the formation of a Force-Closure grasp whose quality is then incrementally increased. At first, it resorts to a recursive continuous collision detection (CCD) search in order to obtain finger configuration candidates that have one or more contacts with the object. Then, it creates an

initial Force-Closure (FC) grasp in the quickest time by selecting the finger configuration candidates that contains the most number of contacts. In the next stage, additional finger configuration candidates, with one or more contacts, are generated for the fingers that had not been searched up until this point. Each contact is then converted to a set of wrenches that represent a discrete approximation of the corresponding friction cone. Then, the grasp is incrementally improved by adding the wrenches that provide the most increase in size of the weakest region in the wrench space. An extension of the weakest region improves the grasp since the grasp quality is a direct measure of the weakest part of the wrench space [37]. Thus, the procedure selects the finger configuration candidates that correspond to the wrenches that were added into the wrench space. In addition, the obstacle is grown after each search by uniting it with the links of the finger that was just processed. Thus, the next recursive CCD search discards results where contacts are obstructed by fingers that were previously searched. At the end of the search, the technique produces a grasp that assures immobility of the object and is free of any interference between the fingers. This technique is applicable to hand models with any kinematic configuration and that is composed of triangulated links of any shape and size.

In this technique, the concepts of Force-Closure and grasp quality become fundamental tools for seeking a hand configuration that produces a reliable grasp. In particular, they facilitate selecting the finger configuration candidates whose contacts best favor the formation of an adequate grasp. The grasp quality and Force-Closure computations employed here are carried out with the methods described in chapter 3 and 4. Also, the recursive CCD search method, described in chapter 4, is used in the generation of finger configuration candidates that correspond to poses in which the finger can interact with the object through accessible contacts. Lastly, the grasp synthesis technique given in this thesis takes advantage of the faceted

representation of the wrench space as well as the idea of hyperplane half spaces; both of them were introduced in chapter 3 and 4 respectively.

## 5.1 Overview of Related Works

The development of the grasp synthesis method exhibited here has been mainly influenced by the works of Xue [69] and Roa [78] in the area of grasp synthesis. Thus, this section aims to describe the particularities of each approach so as to indicate their influence, and to put the innovative features and advantages of the novel grasp synthesis method into context.

The work by Xue et al. [69] describes a grasp synthesis method that uses iterative continuous collision detection tests and swept volumes in order to find all possible contact points within the allowable range of motion of the fingers in the hand. Then, all found contacts along with their corresponding joint values are organized into a hierarchical structure known as a Kd-tree. Their claim is that the Kd-tree provides an efficient storing and extraction mechanism for the contacts that are found. At a later stage, their procedure seeks to optimize an initial grasp by generating as many grasp as possible with the contacts that were found for a single finger; the other fingers are kept immobile. All the new grasps are then sorted into a list that indexes them in descending order according to their grasp quality. Thus, the list is examined until the first grasp with no collisions amongst the fingers is observed. The selected grasp is set as the current best grasp and the search for a better grasp proceeds onto the next finger. They perform the entire search offline and then store the result such that it can be used later on a real time session.

The most influential feature of the method proposed by Xue et al., in relation to the technique described in this thesis, is found in the innovative utilization of CCD for finding contact points. However, their method involves finding every possible contact for every finger. Hence, it is very likely that this approach yields contacts that can never be reached due to finger

interference; so computing these contacts is unnecessary. On the contrary, the technique presented here embeds the recursive collision detection search within the grasp improvement procedure and converts each finger into obstacles after they have been searched. This implies that obstructed contacts are not computed and so the computational burden is greatly reduced.

Another important remark about the work of Xue is that a grasp quality evaluation is performed for every grasp candidate that is produced. Their grasp quality computation consists of generating a convex hull approximation of the wrench space and extracting the offset distance corresponding to the facet closest to the origin. They rely on the programmatic library Qhull for carrying out convex hull computation, just as it is done in the work presented here. Furthermore, the convex hull operation can be rather expensive for 6-dimensional sets like the wrench space. Hence, it is very impractical to evaluate the grasp quality for every combination of contacts that may result in a grasp. On the other hand, the grasp synthesis method in this thesis performs the grasp quality evaluation only once after a finger is searched and so the number of grasp quality evaluations is greatly reduced.

Nevertheless, Xue's work examines the problems of palm placement and initial hand posture that occur during the pre-grasp stage. They rely on the method suggested by [67] which consists of decomposing the object into geometric primitives so as to generate feasible approach directions for the palm. Also, initial hand postures or hand pre-shapes are obtained from a database that categorizes human hand pre-shapes in relation to object shape and task requirements. In the work presented in this thesis, the problems associated with the pre-grasp stage are not explored but could be considered as a viable direction for future research.

In regards to the work by Roa [78], the problem of grasp synthesis is explored by exploiting the geometric notion of the wrench space. Roa's method begins by discretizing the

surface of the object so as to produce a collection of contact points and normal vectors; these are then converted to 6-dimensional wrenches according to the designated friction cone. In the next step, the set of wrenches that were previously generated are further processed by a two stage procedure. The first stage creates a temporary wrench space from random points and then defines a region in the wrench space where the next point should be located so as to form a Force-Closure grasp. He searches for points located in this region by means of hyperplane half-spaces; the search continuous until a point that belongs in this region is found. Similarly, the second stage attempts to improve the FC grasp by finding regions of the wrench space where new points can increase the grasp quality. Then, points in the wrench space are iteratively substituted by new points that are found inside the designated region; the grasp quality and Force-Closure evaluations are performed after each substitution. The end result of this method is claimed to be a Force-Closure grasp with optimum quality.

The most novel aspect of Roa's work is found in his use of hyperplanes to explore the discretized geometry of the wrench space. In the context of his work, hyperplanes provide an efficient mechanism to discard points that do not improve the grasp and so a great reduction of the configuration space is achieved. But, it is also reported that his technique must compute the grasp quality for several combinations of points that may lead to a better grasp. This approach could become inconvenient since a better grasp may be attained only after conducting very many grasp quality computations; the simplest reported example required conducting 48 grasp quality evaluations before finding the best grasp. In a similar manner, the grasp synthesis technique described here relies on hyperplanes to eliminate points that do not offer any improvements. However, it does not perform any point substitutions and thus avoids making unnecessary evaluations of grasps that do not have a chance at improving the quality.

Another aspect Roa's method is that each contact is examined, with the hyperplanes, one at the time during the search for a better grasp. This approach works well if it is assumed that a finger and the object only share a single contact, but it may be insufficient to characterize more realistic grasps where the finger is in contact with the object at multiple locations. Examining each associated contact separately prevents making and adequate assessment of the collective contribution of the associated contacts as a single candidate. Hence, the method proposed in this paper takes this insufficiency into consideration by examining a group of associated contacts as a single candidate with the aim of maintaining their unique correspondence with a finger configuration. For instance, a finger configuration, with multiple contacts at its links, reports the group of contacts as a single candidate that may improve the grasp.

In addition, Roa's technique relies on friction cone linearizations; this means that each contact is represented by a finite set of points that approximate the friction cone. Hence, the examination of a contact implies that each point in the friction cone must be checked and so numerous computations may be necessary in order to process high resolution friction cones. The support mapping operation is used by the technique presented here so as to select a single point from the friction cone that best represents the contact during a hyperplane test. Consequently, only one point of the contact's friction cone is checked whenever that particular contact is examined with the hyperplane test.

The works of Xue and Roa have proposed very innovative solutions for the grasp synthesis problem. Each of these works present unique features that have influenced and inspired the development of the technique for effective grasp synthesis presented in this paper. Therefore, the work in this thesis honors their contribution to the field of robotic grasping by taking their ideas a step further.

## 5.2 Grasp Synthesis Method Description

Before proceeding with the full description of the algorithm, a number of assumptions and formalizations need to be established:

a- The hand models are composed of a palm object $P$ and $N$ finger objects $F_n$ which in turn are composed of 3-dimensional links $l_m$. The geometry of each link $l$ is represented by a list of triangles that are arranged according to an adjacency list. Furthermore, each finger defines its own kinematic configuration, which is expressed according to the Denavit-Hartemberg (DH) standard convention [96]. This convention is inclusive of the link lengths and the axis of rotation and so it facilitates resolving the spatial configuration of a link, which depends on the joint angles and the relative configuration of every other link. The palm is a triangulated mesh as well and it carries the relative position and orientation of each finger base described in palm coordinates.

b- A finger configuration candidate $FConf_i$ is a convenient entity that represents a set of contacts $c_{ij}$ that maintain a unique correspondence with a finger pose. Hence, a finger configuration $FConf_i$ candidate allows assessing the collective contribution of a group of associated contacts $c_{ij}$ as if they were a single candidate; the contact group $G_i$ is used here to represent a group of associated contacts. In addition, the recursive CCD search reports each result as an array of finger configuration candidates; where each $FConf_i$ stores joint values $\theta_{TOC}$ at time of collision (TOC) and their contact group $G_i$.

c- A contact $c_{ij}$ is described by a position vector $p_{ij}$, a normal vector $n_{ij}$, in the direction of the inward surface normal, and a friction cone model $FC_{ij}$; the position vector and normal vector are expressed in object coordinates. The friction cone $FC_{ij}$ of the contact $c_{ij}$ is approximated by a finite set of forces $f_{ijk}$, which in turn are mapped into wrench vectors

$w_{ijk}$ in the 6-dimensional space of generalized forces. Thus, the wrench set $s_{ij}$ holds all the wrench vectors $w_{ijk}$ that lie in the boundary of the space defined by the friction cone $FC_{ij}$. Moreover, the wrench sets $s_{ij}$ from the same contact group $G_i$ are placed within a wrench group $\psi_i$ and so a contact group $G_i$ is represented by a wrench group $\psi_i$ in the 6-dimensional space of generalized forces.

d- The palm's position and orientation must be set by the user in such way that the object $OBJ$ is within reach of the fingers $F_n$. If the palm and the object touch then it's possible to use the corresponding contacts $c_0$ to initialize the wrench space $W$. This permits taking into account the contribution of the palm in the formation of the grasp.

e- The search order of the fingers $F_n$ is user-defined, although it is set equal to the order in which they are listed in the hand model whenever a user-defined search order is not provided. In most cases, a different search order yields a different grasp.

f- A joint profile $J_n$ defines the allowable range of motion of the finger $F_n$ that is to be used in the search for a grasp. Therefore, the start and end joint values, $\theta_{nm,start}$ and $\theta_{nm,end}$ respectively, as well as the number of steps $\theta_{nm,steps}$ dictate the joint profile $J_n$ for the finger $F_n$.

The grasp synthesis technique is divided in two main stages: A search for Force-Closure stage and a grasp quality improvement stage. So, the first stage forms a Force-Closure grasp by searching the least number of fingers possible. The second stage improves the grasp by selecting finger configuration candidates that provide the most reinforcement of the wrench space. The description of each stage is provided in full detail in the following sections.

### 5.2.1 Stage 1: Search for a Force-Closure Grasp

This stage is an iterative procedure that begins by initializing the wrench space $W$ with the wrenches that correspond to the contacts between the palm $P$ and the object $OBJ$. When there aren't any initial contacts, the wrench space must be initialized to an empty set. Then, the *RecursiveCCDSearch* routine, described in chapter 4, is called in order to generate finger configuration candidates $FConf_i$ for the first finger $F_i$. At this point one of two possible actions is taken depending on the current stage of the wrench space $W$. If the wrench space is empty, $W$ is initialized to the elements of the wrench group $\psi_i$ from the finger configuration $FConf_i$ that reports the most contacts $c_{ij}$. Whenever the wrench space already has elements, then a temporary wrench space $T$ is created by uniting the wrench space $W$ and the origin $O$ in the following way:

$$T = ConvexHull(W \cup O) \tag{5.1}$$

The formation of $T$ is followed by the identification all the hyperplanes $H_t \in T$ that contain the origin $O$. Thus, the intersection of all positive half-spaces $H_t^+$ corresponds to the region $R$ where the next point must be located so as to achieve Force-Closure (this statement is valid according to the definition of Force-Closure discussed in chapter 3). Thus, the region $R$ is obtained according to the statement below:

$$R = \bigcap_{t=1}^{K} H_t^+ \tag{5.2}$$

Next, each wrench group $\psi_i$, extracted from the candidates $FConf_i$ obtained previously, is checked with the aim of finding the wrench sets $s_{ij} \in \psi_i$ such that there is at least one point $w_{ijk} \in s_{ij}$ located inside $R$; this statement is equivalent to the following expression:

$$w_{ijk} \in R = \bigcap_{t=1}^{K} H_t^+ \tag{5.3}$$

Whenever such point $w_{ijk}$ is found, the origin $O$ can be confined to the interior of the space defined by the convex hull of the union between the wrench space $W$ and all the elements in the wrench group $\psi_i$; such is the condition for Force-Closure, as given in [38]. Consequently, the fulfillment of expression 5.3 implies that the following statement is valid as well:

$$O \in ConvexHull(W \cup \psi_i) \tag{5.4}$$

In practice, it is not necessary to check each point $w_{ijk}$ in the wrench group $\psi_i$ in order to determine if a least one point lies inside the region $R$. Instead, we can resort to the support mapping operation so as to obtain a single point $w_{ij,support} \in s_{ij}$ from each wrench set in $\psi_i$ such that the support points can be used in a series of hyperplane tests. In this case, the support mapping operation $S_{s_{ij}}(n_{H_t})$, applied to set $s_{ij}$, yields the point $w_{ij,support}$; where $n_{H_t}$ corresponds to the normal vector of hyperplane $H_t$ (support mapping and hyperplanes tests computations were discussed in chapter 4). Therefore, it is observed that if at least one element of the wrench set $s_{ij}$ is located in the positive half space $H_t^+$, then the support point $w_{ij,support} \in s_{ij}$ is farthest from $H_t$ in the direction of $n_{H_t}$ and it belongs to the positive half-space $H_t^+$. In the opposite case, where no element of set $s_{ij}$ is in $H_t^+$, the point $w_{ij,support}$ is closest to $H_t$ and it is located either on the plane or in the negative half-space $H_t^-$.

By using the discussed support mapping and hyperplane tests operation, it's possible to determine the group $\psi_i$ that contains at least one support point $w_{ij,support}$ located in every positive half-space $H_t^+$; thus the condition in equation 5.3 is met. At this instance, Force-Closure has been achieved; so the new wrench space $W$ is created and the iteration ends. Whenever there are multiple wrench groups that obey $w_{ijk} \in R$ , then the wrench group $\psi_i$ with the most wrenches $w_{ijk}$ is chosen. If a wrench group $\psi_i$ that fulfills equation 5.3 could not be found,

then the wrench group with the most elements is added to $W$ and the algorithm proceeds with the next iteration. At the end of every iteration, the finger $F_n$ is incorporated into the environment $ENV$ so that it can be treated as an obstacle during the following iteration. Once again, the procedure calls the *RecursiveCCDSearch* routine in order to generate a new set of candidates $FConf_i$ for the next finger $F_n$. Thus, the entire procedure is repeated until Force-Closure is achieved. Alg. 5.1 shows the pseudo code for the entire method.

The quickness of the FC search procedure depends heavily on the motion range at the joints and the order at which the fingers are searched. Hence, a different search order can lead to a quicker resolution of a Force-Closure grasp. Also, initial contacts between the palm and the object produce restraint forces at locations that are difficult for the fingers to reach; thus supplying these contacts can lead to a quicker formation of a Force-Closure grasp. A graphical depiction of the Force-Closure search in the context of a 2-dimensional wrench space is portrayed in Fig 5.1.

Algorithm 5.1 *FindForceClosureGrasp*

**Inputs:**

$H$: Hand model with palm $P$ and $N$ fingers $F_n$, $n = 1, \dots, N$.

$OBJ$: Object with spatial configuration ${}^{W}_{Obj}T$ in world coordinates.

$c_0$: Optional initial contacts between palm $P$ and object $OBJ$.

$J_n$: Joint Profiles for each finger $F_n$.

**Outputs:**

$FConf_n$: Finger configuration for each finger $F_n$ corresponding to the FC grasp.

$W$: Wrench space of the Force-Closure grasp.

**Begin:**

$c_0 \implies \psi_0$ Obtain wrench group from initial contacts.

$W \equiv ConvexHull(W \cup \psi_0)$ Compute new wrench space.

$ENV \equiv \emptyset$ Environment is set to empty.

**While** $W$ is not FC **iterate**

$FConf_i \Leftarrow RecursiveCCDSearch(F_n, OBJ, J_n, ENV)$; Find finger configuration candidates.

$\psi_i \Leftarrow FConf_i$; Extract each wrench group.

$s_{ij} \Leftarrow \psi_i$; Extract each wrench set from wrench groups.

**If** $W$ is $\emptyset$ **then**

Select $\psi_i$ with the most elements

$W \equiv ConvexHull(\psi_i)$; Initiate wrench space

**Else**

$T \equiv ConvexHull(W \cup O)$; Create temporary wrench space with $W$ and $O$.

Find all $H_t \in T$ that pass through $O$;

$R \equiv \bigcap_{t=1}^{K} H_t^+$; Create region where the next point must be located.

**End if**

$w_{ij,support} \equiv S_{s_{ij}}(n_{H_t})$; Compute support points.

**If** $w_{ij,support} \in R$ **then**

Force-Closure is achieved, last iteration

Select $\psi_i$ that contains $w_{ij,support}$ ;

**Else**

Select $\psi_i$ that contains the most elements;

**End if**

$W \equiv ConvexHull(W \cup \psi_i)$; Create new wrench space.

Store $FConf_i$;

$ENV \equiv ENV \cup F_n$; Add finger into the obstacle environment

$n \equiv n + 1$; Iteration counter increases by one

**End while**

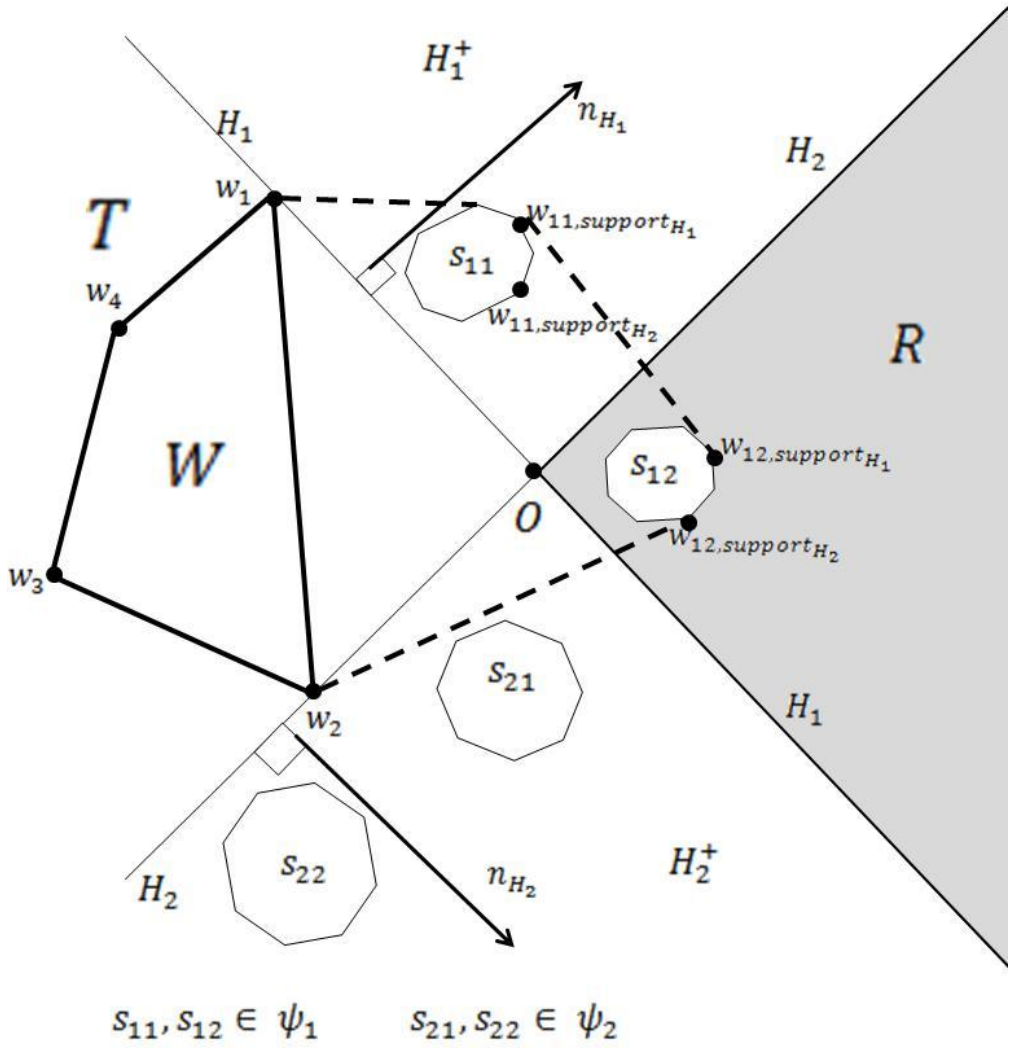


**Figure 5.1: Search for Force-Closure using Hyperplanes and Support Mapping**. The sets $s_{11}, s_{12} \in \psi_1$ contain a point(s) in the region $R$ and so the wrench group $\psi_1$ is selected. $W$ is the current wrench space and $T$ is the temporary wrench space equal to $ConvexHull(W \cup O)$.

### 5.2.2 Stage 2: Grasp Quality Improvement

This stage consists of an iterative search that complements the results obtained by the Force-Closure search procedure in the first stage. It begins by conducting a search for finger configuration candidates $FConf_i$ for the next finger $F_n$; the *RecursiveCCDSearch* routine is used for this purpose too. The search order from the first stage designates which finger is to be searched next. As in the first stage, each finger configuration candidate holds a wrench group $\psi_i$

that contains one or several wrench sets $s_{ij}$. In the next step, the grasp quality $e_1$ and the closest hyperplane $H_{e_1}$ are computed from the wrench space $W$; where the hyperplane $H_{e_1}$ corresponds to the closest facet in $W$ and it is located at a distance $e_1$ from the origin $O$. By examining each wrench set $s_{ij}$ of every group $\psi_i$, it can be determined if there is at least one support point $w_{ij,support} \in s_{ij}$ that rests in the positive half-space $H_{e_1}^+$. Each group that does not contain one or more points in $H_{e_1}^+$ is discarded and rejected as a selectable candidate. In the event that several wrench groups $\psi_i$ contain a point $w_{ij,support} \in H_{e_1}^+$, the second closest hyperplane $H_{e_2}$ is retrieved so as to find the wrench groups $\psi_i$ that hold at least one support point $w_{ij,support}$ in $H_{e_2}^+$. The purpose is to continue retrieving the next closest hyperplane $H_{e_{l+1}}$ from the wrench space $W$ as long as there are wrench groups $\psi_i$ with a support point $w_{ij,support}$ that resides in $H_{e_l}^+$. This series of hyperplane tests ends once no points $w_{ij,support}$ are found to reside in the positive half-space of the current hyperplane. At this point, we select the wrench group that reported a support point in the positive half-space of the previous hyperplane $H_{e_l}$. Then, the corresponding elements of the selected wrench groups $\psi_i$ are incorporated into the wrench space $W$, and so a new $W$ is formed. Once again, the algorithm proceeds to the next iteration unless there aren't any more fingers to search. Also, the current finger is converted into an obstacle at the end of each iteration so as to discard unreachable contacts in subsequent searches. The summary of the pseudo code for the grasp quality improvement stage is provided in Alg. 5.2.

Algorithm 5.2 *GraspImprovement*

**Inputs:**

| | |
|---|---|
| $H$: | Hand model with palm $P$ and $N$ fingers $F_n$, $n = 1, \dots, N$. |
| $OBJ$: | Object with spatial configuration ${}^{W}_{Obj}T$ in world coordinates. |
| $J_n$: | Joint Profiles for each finger $F_n$. |
| $W$: | Wrench space of FC grasp obtained with the *FindForceClosureGrasp* method. |

**Outputs:**

| | |
|---|---|
| $FConf_n$: | Finger configuration for each finger $F_n$ corresponding to the improved grasp. |
| $W$: | Wrench space of the improved Force-Closure grasp. |

**Begin:**

**While** there are fingers $F_n$ to search **iterate**

- $FConf_i \Leftarrow RecursiveCCDSearch(F_n, OBJ, J_n, ENV)$; Find finger configuration candidates.
- $\psi_i \Leftarrow FConf_i$; Extract each wrench group.
- $s_{ij} \Leftarrow \psi_i$; Extract each wrench set from wrench groups.
- $H_{e_l} \leftarrow W$ Find closest hyperplane in the wrench space;
- $l = 1$; Initiate counter.
- **While** $w_{ij,support} \in H_{e_l}^{+}$ **iterate**
  - Remove all $\psi_i$ such that $w_{ij,support} \notin H_{e_l}^{+}$;
  - $H_{e_{l+1}} \leftarrow W$ Find next closest hyperplane from the wrench space.
  - $l = l + 1$;
- **End while**
- Select $\psi_i$ such that $w_{ij,support} \in H_{e_{l-1}}^{+}$ Keep wrench group that reported a support point in positive half-space of previous hyperplane.
- $W \equiv ConvexHull(W \cup \psi_i)$; Create new wrench space.
- Store $FConf_i$;
- $ENV \equiv ENV \cup F_n$; Add finger into the obstacle environment
- Search next finger $F_{n+1}$;

**End while**

The main idea in this procedure is to grow the weakest sectors of the wrench space $W$ by incorporating new points at key locations. In here, the weakest sectors correspond to the facets of $W$ that lie considerably closer to the origin than other facets. By adding the group $\psi_i$ with the most reported support points $w_{ij,support}$ in $H_{e_l}^{+}$, it is ensured that the majority of the weakest sectors are expanded outwards so as to induce a larger average resistance of the grasp to external disturbances. The procedure for grasp improvement is shown in Fig. 5.2.

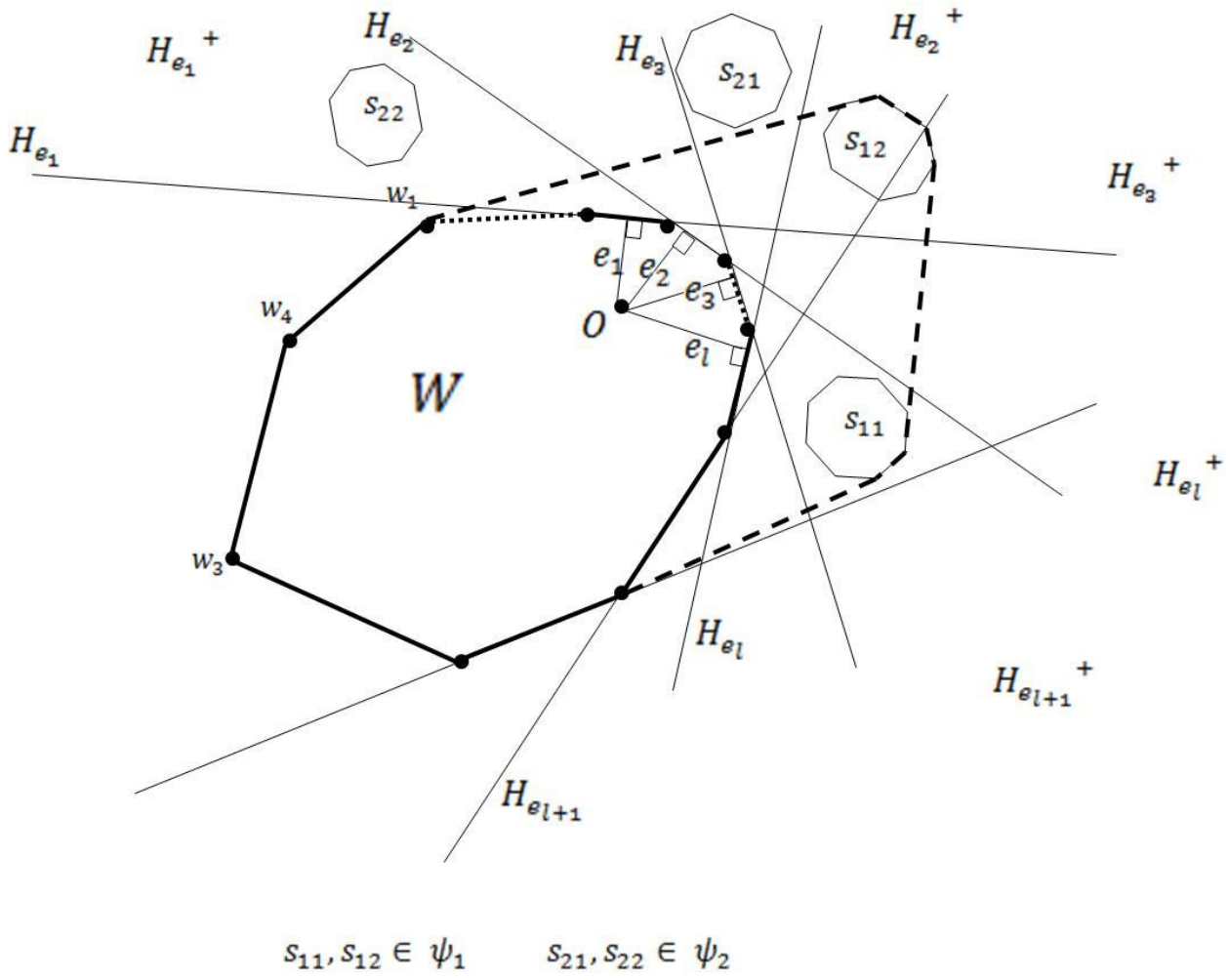


**Figure 5.2: Grasp Quality Improvement.** The wrench sets $s_{11}, s_{12} \in \psi_1$ are selected because they occupy more half-spaces $H_{e_l}^+$ than the wrench sets $s_{21}, s_{22} \in \psi_2$.

## 5.3 Implementation Overview

The two stages in the grasp synthesis method have been incorporated as computational routines in the Robotic Grasping Toolkit. This facilitates applying these methods to representations of finger and hand models that also exists within the database of this toolkit. Hence, the experiments presented in the next section were carried out by means of the computational routines provided in it. A thorough description of the Robotic grasping toolkit is given in chapter 6.

## 5.4 Grasp Synthesis Experiments

In order to demonstrate the efficacy and flexibility of the grasp synthesis method, three experiments involving three different hand models and objects are presented. There are three tests in each experiment, where each test consists of synthesizing a grasp for a hand model and an object. Only one object is used per tests and these objects are a sphere, a rook chess piece and a rabbit toy. Thus, each test increases the difficulty of synthesizing a grasp since it uses an object of greater geometric complexity than the one in the previous test. In the context of these experiments, each object is a triangulated mesh and their geometric complexity is dictated by the number of triangles and the shape that the triangles constitute. Consequently, the computational cost required to process an object goes in hand with the amount of triangles and irregularities in the shape. The table 5.1 shows the details of each object used for the experiments.

**Table 5.1: Object details.**

| | Triangles | Vertices | Length | Width | Height |
|---|---|---|---|---|---|
| Sphere | 3590 | 1791 | 8 | 8 | 8 |
| Rook | 6590 | 19770 | 5.47 | 5.45 | 10.5 |
| Rabbit Toy | 31937 | 95811 | 13.32 | 6.91 | 22.46 |

Furthermore, the hand model utilized by each experiment contains more degrees of freedom and geometrically complex link objects than the hand model in the preceding experiment. Thus, the order in which the experiments are presented is meant to escalate the difficulty that the grasp synthesis method must overcome in order to find a grasp. In each experiment, the location of the palm has been predetermined in order to allow it coming into contact with the object. Then, the resulting contact points between the palm and the object are used to initialize the wrench space so as to speed up the resolution of a grasp. To speed up the

formation of an FC grasp in the first stage of the grasp synthesis method, the finger search order has been predefined as well. Lastly, all of the experiments presented here rely on a point contact friction model (PF) with a friction value of 0.5 and a resolution of 8 elements for every friction cone.

### 5.4.1 Three Finger Hand Experiment

This experiment relies on the Three Finger hand model which has a total three fingers with three degrees of freedom each. All links are identical and their length has been scaled to 5 units. The entire hand is composed of a total of 5984 triangles including the palm; the table 5.2 gives the kinematic configuration in terms of the Denavit-Hartemberg (DH) convention. In the first test, the grasp synthesis method finds a grasp between the hand and a sphere with a radius of 5 units. The results revealed that a FC grasp is computed by the Force-Closure search stage in over 4 seconds with a grasp quality of 0.44. Also, this stage only needed to use the palm and one finger that make contact with the top and bottom of the sphere respectively. The second stage (grasp improvement stage) improved the quality of the grasp found in the first stage to a value of 0.690 in a total of 30 .5 seconds by incorporating contacts that involved the rest of the fingers. The images corresponding to this test are shown in Fig 5.3.

**Table 5.2: Three Finger hand DH parameters.**

| Three Finger Hand | Link Length $a_i$ | Link twist $\alpha_i$ | Link offset $d_i$ | Joint variable $\theta_i$ |
|---|---|---|---|---|
| Index Finger | 5 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 |

A peculiar result is given by the second test between the hand and the rook piece. The initial grasp was computed in 4.138 seconds and it shows a quality of $1.93 \times 10^{-4}$. Then, the grasp quality is increased to a much larger value of 0.197 in just 7.163 seconds during the second grasp synthesis stage. The low grasp quality value corresponding to the initial grasp is mostly due to the low frictional restriction that the hand exerts at the base of the rook. Eventually, the incorporation of the other two fingers helps reinforce the resistive forces at the base and sides of the rook and so a much large grasp quality value is obtained. The Fig. 5.4 shows the formation of the grasp corresponding to this test.

The last test involving the rabbit toy object also shows a drastic increase in grasp quality that goes from 0.037 to 0.148; the first and second stages were completed in 7.35 and 22.9 seconds respectively. This test was particularly difficult because the object contains 31937 triangles, which is approximately 5 and 10 times more triangles than objects used for the first and second tests. Also, the shape of the rabbit toy object presents the most geometric irregularities in comparison to the sphere and rook objects. Consequently, this test registered the lowest grasp quality value at the end of the second stage; although it achieved a greater grasp quality increase than the first test. The images in Fig 5.5 correlate to this result because it is seen that the grasp only restrains the head portion of the object while the rest remains untouched. This implies that a disturbance force applied somewhere in the body area has a greater chance at breaking the grasp that if it were to be applied in the head region.

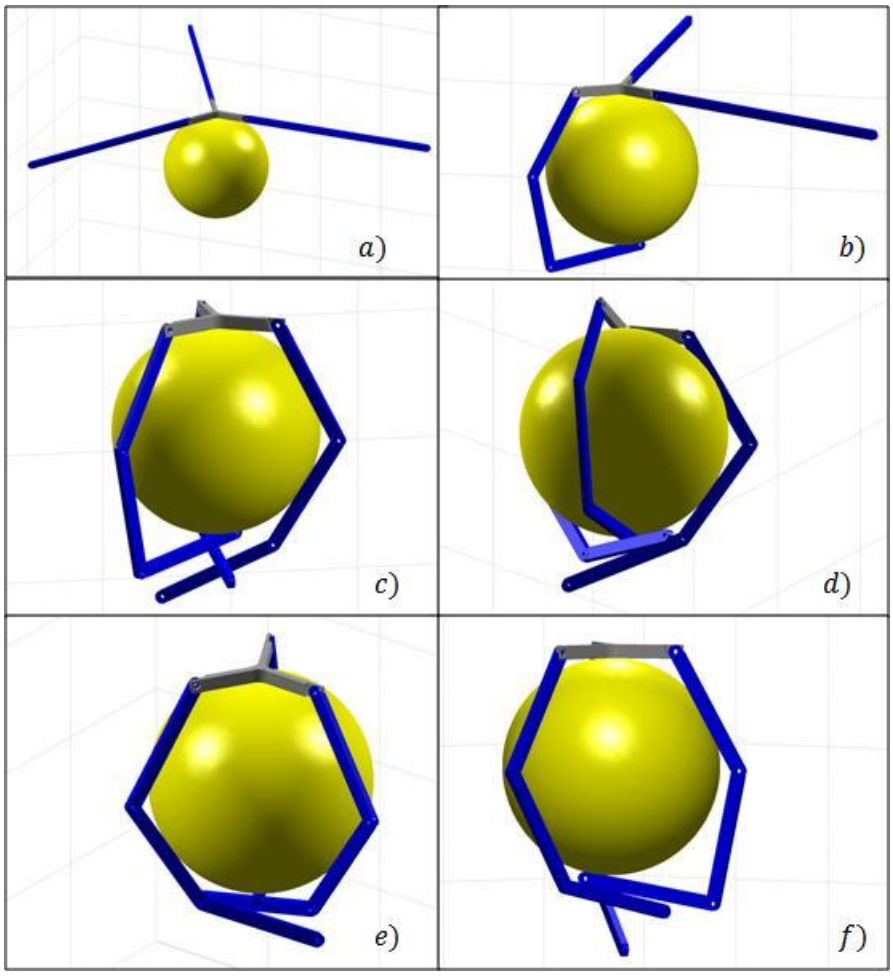


**Figure 5.3: Three Finger hand vs. Sphere.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

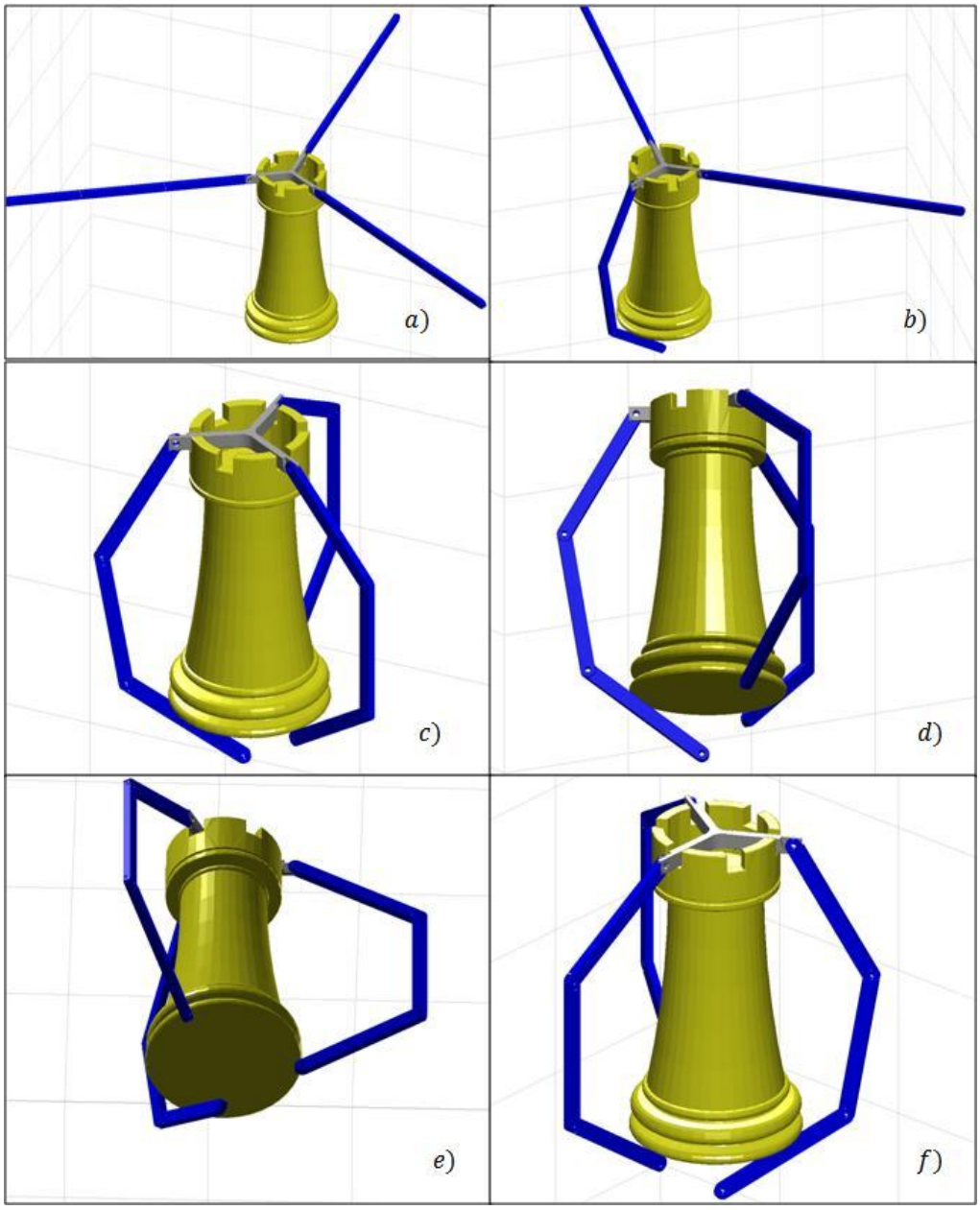


**Figure 5.4: Three Finger hand vs. rook chess piece.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

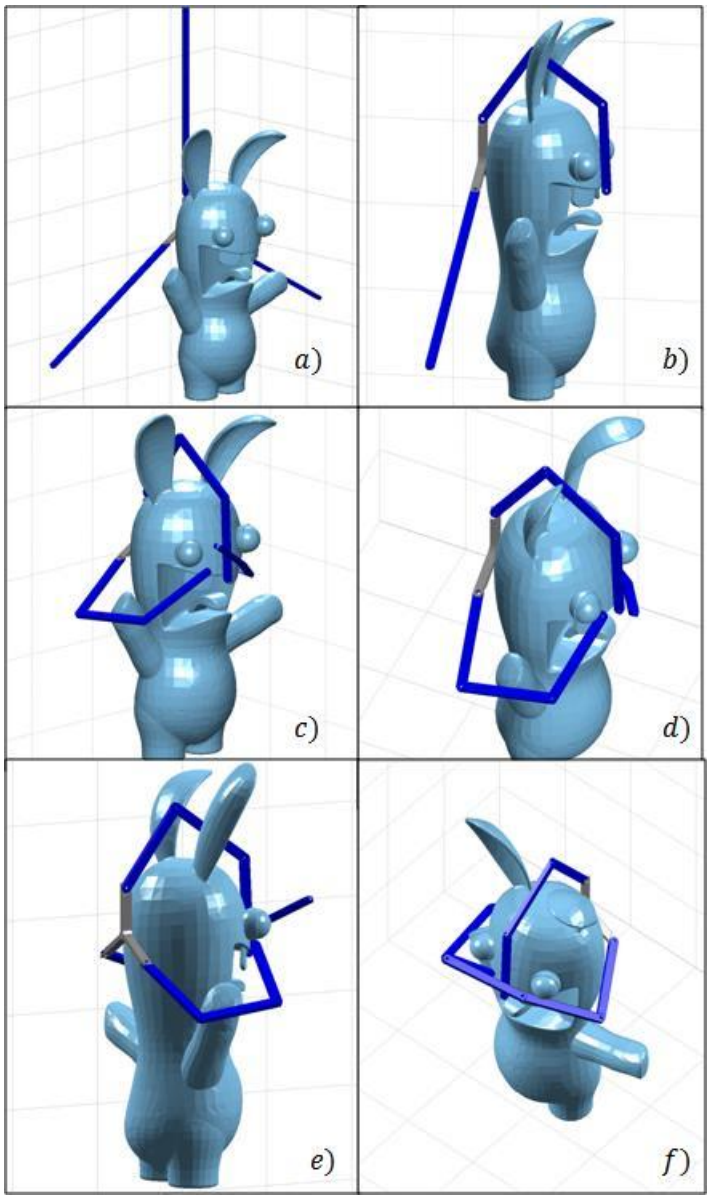


**Figure 5.5: Three Finger hand vs. Rabbit toy.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

**Table 5.3: Three Finger hand results.**

| Object | Triangles | Search Order | Grasp Synthesis Stage | Grasp Quality | Time (s) | Fingers searched |
|---|---|---|---|---|---|---|
| sphere | 3590 | [2 1 3] | 1 - Find FC | 0.4392 | 4.16 | 1 |
| | | | 2 - Improve Grasp | 0.6905 | 30.48 | 3 |
| Rook | 6590 | [2 1 3] | 1 - Find FC | 0.0002 | 4.14 | 1 |
| | | | 2 - Improve Grasp | 0.1975 | 7.16 | 3 |
| Rabbit Toy | 31937 | [2 1 3] | 1 - Find FC | 0.0367 | 7.35 | 1 |
| | | | 2 - Improve Grasp | 0.1484 | 22.99 | 3 |

In summary, it is observed that largest grasp quality value recorded on the first test reflects the fact that the fingers envelop the entire sphere very evenly. This is not the case for the rook object since the hand cannot reach the bottom of the rook's base and so it must rely on frictional forces to support it. Furthermore, only one finger was necessary to achieve Force-Closure during the first stage for each test. The entire set of results for all the test in this experiment is shown in table 5.3.

### 5.4.2 Rowdy Hand Experiment

The hand model used experiment is composed of 4 fingers and a larger palm than in the other two hands. The first three finger connected at the top of the palm are identical and posses a total of 4 DOF. The thumb is relatively larger than the rest of the fingers so that it may exert forces in the opposite direction of the palm surface. Furthermore, the entire hand model is composed by a total of 28066 triangles. The DH kinematic parameters for the regular fingers and the thumb are given in table 5.4.

In the first test against the sphere, the hand achieves Force-Closure during the first stage by placing the farthest link of the thumb at bottom of the sphere; this configuration of the thumb induces forces in the opposite direction to those produced by the palm. Thus, it takes 7.35 seconds to determine the initial grasp with a quality value of 0.140. The second stage improves this grasp to a quality value of 0.578 in 15.1 seconds. The final grasp produced by the grasp improvement stage locates the links of the three top fingers at regions of the sphere that are unreachable by the thumb. The Fig. 5.6 contains the images corresponding to this test.

**Table 5.4: Rowdy hand DH parameters.**

| Rowdy Hand | Link Length $a_i$ | Link twist $\alpha_i$ | Link offset $d_i$ | Joint variable $\theta_i$ |
|---|---|---|---|---|
| Index Finger | 0 | $\pi/2$ | 0 | 0 |
| | 5 | $-\pi/2$ | 0 | 0 |
| | 4 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 |
| Thumb Finger | 0 | $\pi/2$ | 0 | 0 |
| | 6 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 |

The second test presents more of a challenge than the first because the shape of the rook object barely fits within the reaches of the hand's workspace. In a similar manner as in the first test, the thumb finger allows forming a FC grasp by extending its joints until the last link comes into contact with the base of the rook. The corresponding grasp quality value of 0.210 is found within 16.5 seconds. Then, the tips of the remaining fingers are placed in the opposite end of the rook's base so as to prevent slipping along the direction of the palms surface. The recorded grasp quality for this stage is 0.306 and the corresponding grasp was obtained after 20.8 seconds. Thus, this test yields a smaller grasp quality than the first test because the length of the rook impedes caging the object with the fingers as it was done with the sphere. The Fig. 5.7 allows seeing this grasp configuration.

The third test also relies on the thumb to resolve an initial FC grasp with a quality of 0.100, which was computed in 23.7 seconds. In this case, the thumb is placed around the rabbit's head so as to place the last link in an opposite location relative to the palm. Furthermore, this test experiences a smaller grasp quality increase than the other tests since only a quality value of 0.172 is computed during the grasp improvement stage after 42.7 seconds. This is due to the fact that the irregularities in the object prevent the three remaining fingers from reaching various regions of contact. Thus, the fingers can only touch around the face region where the thumb is already placed and so their contribution is considerably reduced. The stages related to the formation of this grasp are shown in Fig 5.8.

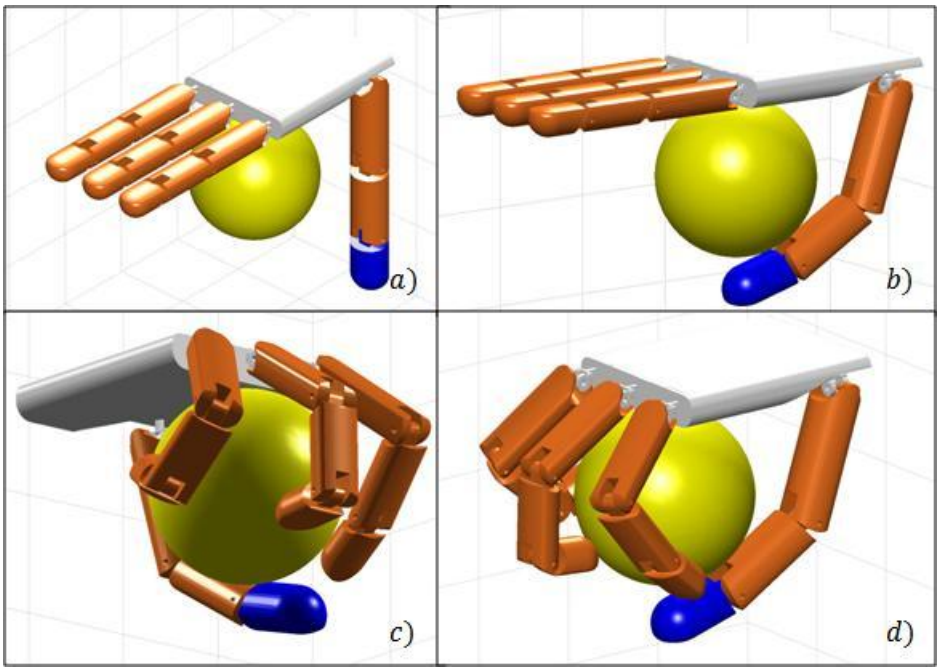


**Figure 5.6: Rowdy hand vs. Sphere**. The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c and d show different views of the improved grasp.

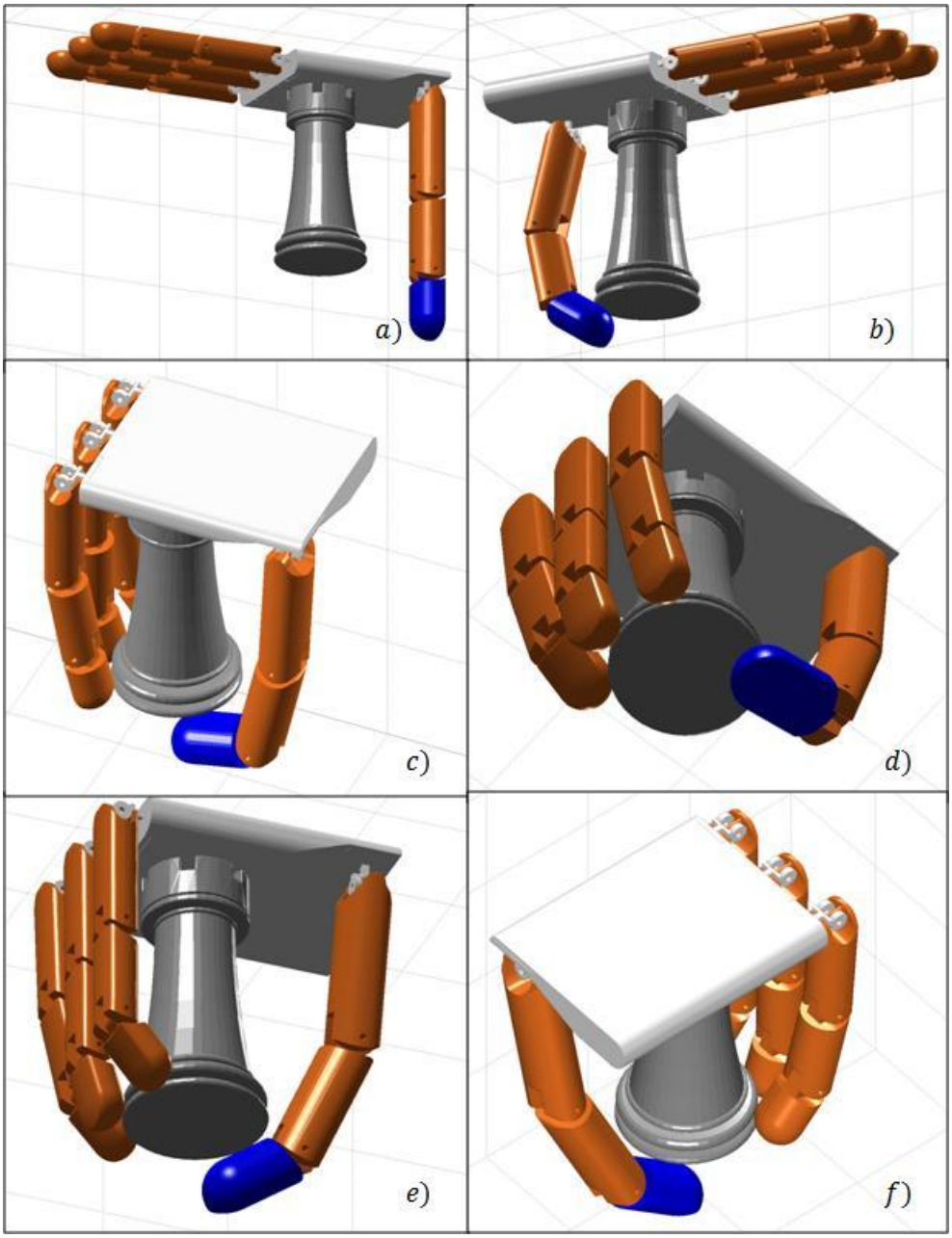


**Figure 5.7: Rowdy hand vs. Rook Chess Piece.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

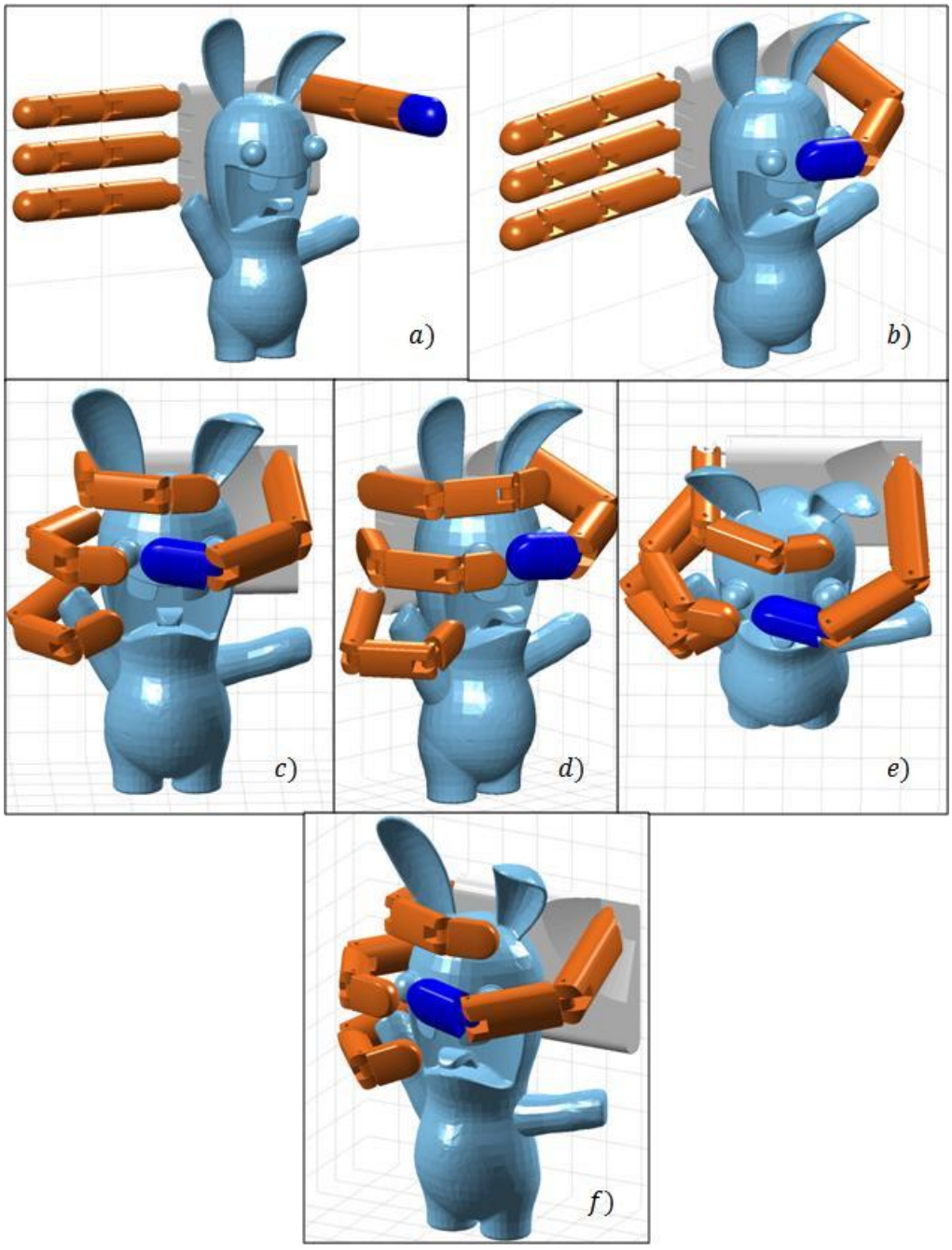


**Figure 5.8: Rowdy hand vs. Rabbit toy.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

In conclusion, the hand model used in this experiment has shown to rely on the thumb finger very often. This is confirmed by the fact that the grasp quality was not greatly favored by the contribution of the remaining fingers in the same degree as it was observed for the first experiment. However, this hand achieved larger grasp quality values for the rabbit and rook toy objects in relation to the first experiment. Coincidentally, these two cases correspond to the tests where the Three Finger hand registered the lowest qualities. The table 5.5 contains the results pertaining to these tests.

**Table 5.5: Rowdy hand results.**

| Object | Triangles | Search Order | Grasp Synthesis Stage | Grasp Quality | Time (s) | Fingers searched |
|---|---|---|---|---|---|---|
| Sphere | 3590 | [4 2 3 1] | 1 - Find FC | 0.139688131 | 7.346436 | 1 |
| | | | 2 - Improve Grasp | 0.577532726 | 15.0815 | 4 |
| Rook | 6590 | [4 2 3 1] | 1 - Find FC | 0.210036925 | 16.54367 | 1 |
| | | | 2 - Improve Grasp | 0.305938205 | 20.8084 | 4 |
| Rabbit Toy | 31937 | [4 2 3 1] | 1 - Find FC | 0.100525552 | 23.6711 | 1 |
| | | | 2 - Improve Grasp | 0.17160677 | 42.73118 | 4 |

### 5.4.3 Prototype1 Hand experiment

The Prototype1 hand is composed of 30908 triangles and possesses 20 degrees of freedom and so it is the most complex articulated mechanism used in these three experiments. All five fingers including the thumb contain 4 links and palm presents several gaps instead of a flat surface of contact as seen in the second hand model. The kinematic configuration for each finger in this hand is given in table 5.6.

**Table 5.6: Prototype1 hand DH parameters.**

| Prototype1 Hand | Link Length $a_i$ | Link twist $\alpha_i$ | Link offset $d_i$ | Joint variable $\theta_i$ |
|---|---|---|---|---|
| Index Finger | 0 | -π/2 | 0 | 0 |
| | 3.81 | 0 | 0 | 0 |
| | 2.7 | 0 | 0 | 0 |
| | 2.26 | 0 | 0 | 0 |
| Middle Finger | 0 | -π/2 | 0 | 0 |
| | 4.45 | 0 | 0 | 0 |
| | 2.86 | 0 | 0 | 0 |
| | 2.26 | 0 | 0 | 0 |
| Ring Finger | 0 | -π/2 | 0 | 0 |
| | 3.81 | 0 | 0 | 0 |
| | 2.86 | 0 | 0 | 0 |
| | 2.26 | 0 | 0 | 0 |
| Pinky Finger | 0 | -π/2 | 0 | 0 |
| | 3.18 | 0 | 0 | 0 |
| | 2.22 | 0 | 0 | 0 |
| | 2.26 | 0 | 0 | 0 |
| Thumb Finger | 0 | -π/2 | 0 | 0 |
| | 5.08 | 0 | 0 | 0 |
| | 3.33 | 0 | 0 | 0 |
| | 2.26 | 0 | 0 | 0 |

The first test produces an initial grasp for the sphere with a quality of 0.100 by wrapping the thumb over the surface of the sphere so that each link comes into contact with it. Next, the grasp improvement stage synthesizes a grasp where the sphere has been enveloped by the rest of the fingers. Thus, the increased quality is of 0.349 and the grasp was resolved in a time of 24.5 seconds. It is observed that the contribution of the fingers used during the second stage allows increasing the quality to about 3 times of its original value. The images for this test are shown in Fig. 5.9.

The second test corresponds to the rook object and it required using three fingers in order to find an initial FC grasp with a quality of 0.005 in a time of 16.6 seconds. This is the only test in all of the experiments in which it was necessary to process this many fingers during the first stage. Consequently, a shorter processing time of 8.96 seconds was necessary during the second stage so as to obtain a grasp with an improved quality of 0.08. The smaller length of the fingers in this hand did not allow generating contacts in adequate location and so this test presented the smallest grasp quality value obtained in all of the experiments. The images in Fig. 5.10 correlate the results that were just described.

The third and last test needed 38.2 seconds to obtain a grasp with a quality of 0.078 for the rabbit toy object. Additionally, it only needed two fingers in order to achieve Force-Closure as seen in Fig. 5.11. In the next stage, the addition of the three remaining fingers allowed incrementing the quality to a value of 0.200 in about 37.64 seconds. Thus, it is concluded that this hand model does not demonstrate the same degree of dependence on the thumb finger as it was observed for the hand used in the second experiment. The table 5.7 shows all the results found in this experiment.

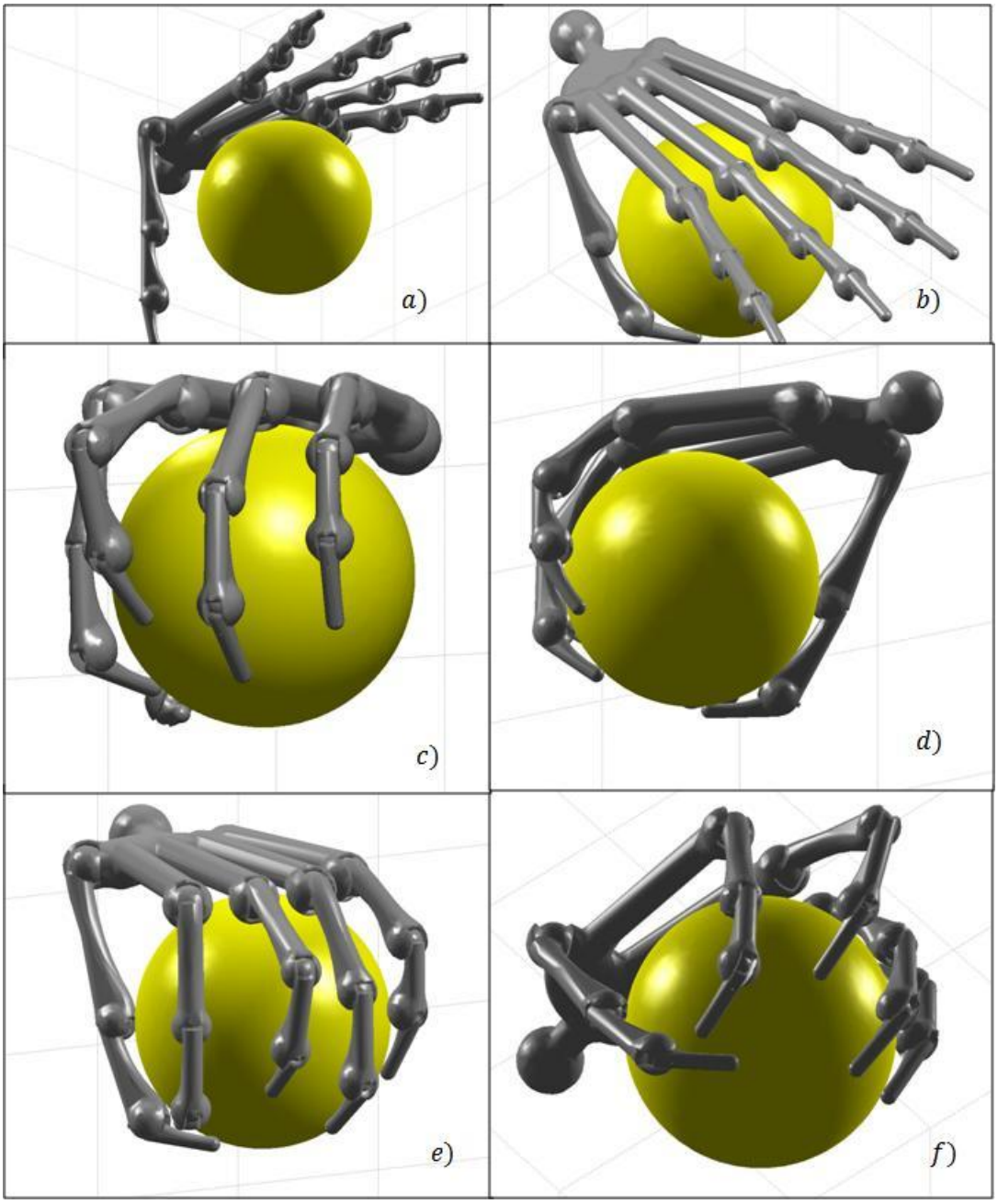


**Figure 5.9: Prototype1 hand vs. Sphere.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

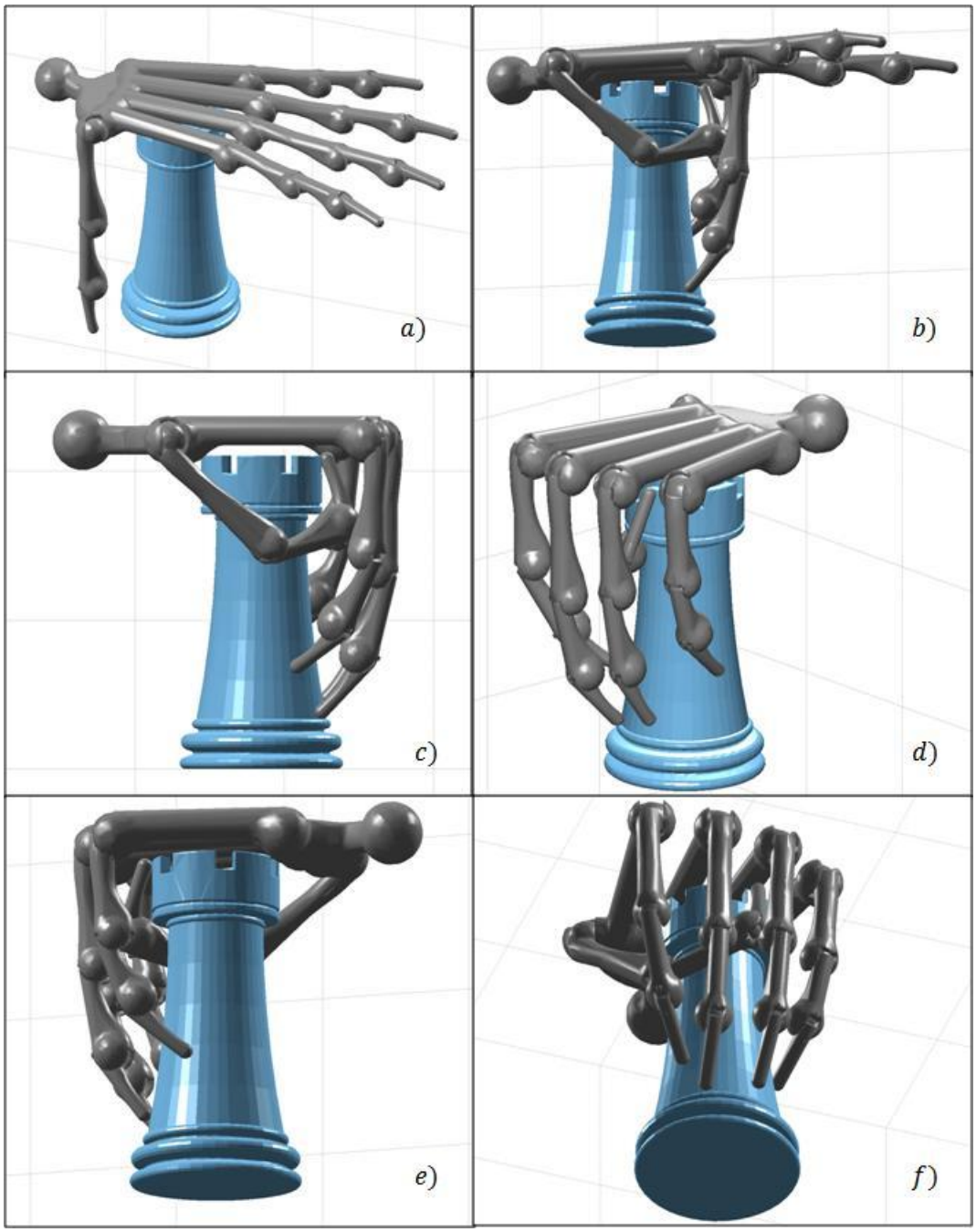


**Figure 5.10: Prototype1 hand vs. Rook chess piece.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

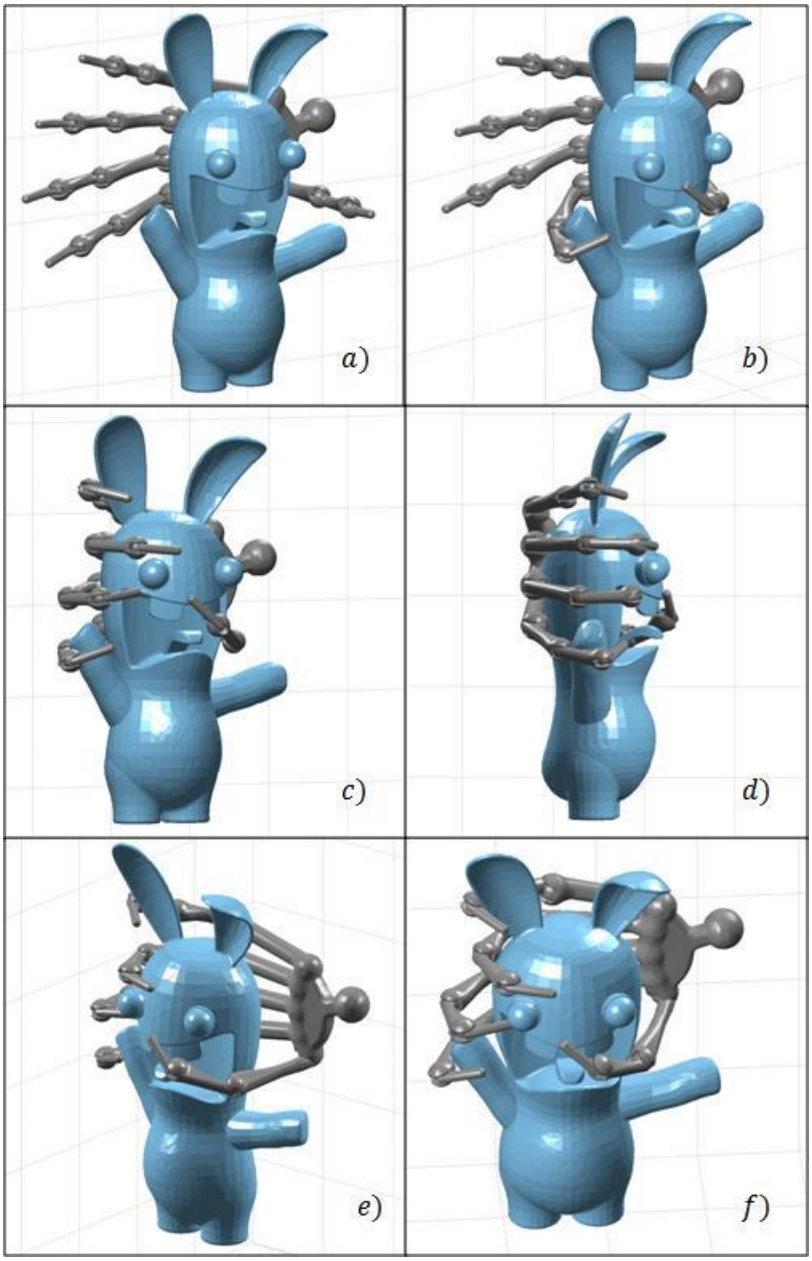


**Figure 5.11: Prototype1 hand vs. Rabbit toy.** The image a) shows the start posture and the image in b) corresponds to the initial FC grasp. The images c, d, e and f show different views of the improved grasp.

**Table 5.7: Prototype1 hand results.**

| Object | Triangles | Search Order | Grasp Synthesis Stage | Grasp Quality | Time (s) | Fingers searched |
|---|---|---|---|---|---|---|
| sphere | 3590 | [5 2 1 3 4] | 1 - Find FC | 0.100 | 7.75 | 1 |
| | | | 2 - Improve Grasp | 0.349 | 24.52 | 5 |
| Rook | 6590 | [5 4 2 3 1] | 1 - Find FC | 0.005 | 16.57 | 3 |
| | | | 2 - Improve Grasp | 0.082 | 8.96 | 5 |
| Rabbit Toy | 31937 | [1 5 2 3 4] | 1 - Find FC | 0.078 | 38.21 | 2 |
| | | | 2 - Improve Grasp | 0.200 | 37.64 | 5 |

In conclusion, these three experiments demonstrate the applicability of the grasp synthesis technique to any combination of hand model and object. Each test showed that the method was able to increase the grasp quality at the end of the second stage, although the increment varied depending on the hand model and the object that were employed. Furthermore, the largest computational time reported by the experiments barely exceeded one minute; which is a drastic improvement in relation to the several minutes or hours consumed by the grasp synthesis methods of [66], [69] and [78]. Thus, this method is a step closer to providing robotic hands with the real time adaptation capability that is necessary for coping with uncertain environments during real time tasks.

## 5.5 Future Improvements

One of the problems that were omitted by this method is that of initial palm placement. A poor selection for the location the palm makes it difficult for the grasp synthesis algorithm to find and improve a grasp. In the worst case, a Force-Closure grasp may not be found at all. Thus, autonomous palm placement is a challenging problem that will be explored in the future.

Furthermore, the finger search order in this grasp synthesis technique has to be selected by the user. This could be rather inconvenient since the finger search order is not always intuitive and needs to be adjusted according to the hand and object models that are being processed. Hence, a promising study is that of guiding the finger search in relation to some parameters that could favor the formation of a grasp.

Lastly, the *RecursiveCCDSearch* technique used for detecting finger configuration candidates omits the use of swept volumes; this was not the case for the work in [68]. The inclusion of swept volumes allows sampling a much larger portion of the finger work space and so more finger configuration candidates could be obtained. Thus, this would allow the grasp synthesis method to form other types of grasp that do not rely on the palm; such as precision grasp that are used for manipulation tasks.

# CHAPTER 6: ROBOTIC GRASPING TOOLKIT

The problem of robotic grasping has been investigated by conducting practical experiments in a variety of ways such as cadaveric studies [61], robotic test bed [50, 59] and simulation software [67, 89]. The study of cadaveric specimens has focused on understanding the biomechanical functionality of the human hand and its effect on prehensile human grasp [61]. Consequently, the results and conclusions obtained from this approach may not be directly applicable to the development of robotic grasping methods. On the other hand, robotic hand prototypes have allowed exploring a number of topics such as robotic hand mechanism design, grasp planning and grasp execution [50, 59]. But, physical robot hands are difficult and expensive to make and troubleshooting these devices demands a great deal of time and effort.

As a consequence, the inconveniences found in robotic hand prototypes may obstruct the progress in robotic grasping and so many investigators opt to rely on simulation software for conducting their own studies. The use of simulation software has proven to be an efficient and flexible approach for the investigation and development of adequate robotic grasping methodologies [67, 89]. Hence, the innate characteristics of simulation software make it a very viable alternative for conducting preliminary grasp studies on hand devices.

The aim of this chapter is to introduce a versatile computational toolkit for investigating robotic hands carrying out grasping tasks. This toolkit features a scripting interface that allows describing an entire grasping task in the form of programmatic functions and computational routines. In addition, it incorporates a number of modules that facilitate making a variety of computations that are applicable towards the development of effective grasp methods. Hence, these modules allow conducting several tasks such as graphical object manipulation, forward kinematics, collision detection, computational geometry and grasp mechanics calculations. Also,

embedded graphical routines enable users to visualize results very intuitively and require minimum programming effort. The inclusion of the physics simulation module makes it possible to create virtual work environment with several constraints and populated by hundreds of objects. Furthermore, this toolkit relies on an object-oriented design approach in order to make each module as self-contained as possible, thus achieving a great level of modularity. The Robotic Grasping Toolkit represents the second most important contribution addressed by this thesis and each one of its components will be described in the following sections of this chapter.

## 6.1 Interacting with Software through Scripting

The use of scripting for interacting with software offers several advantages in relation graphical user interfaces (GUI). Hence, a script is a type of text file that contains a number of word commands corresponding to functions that are executed in a desired sequence within a single run. Quick edits to the script permit introducing new variables that lead to different results from which a broad range of observations can be made. On the contrary, GUIs may need to include a confusing series of clickable buttons and menus in order to provide a similar level of control as scripting.

The toolkit presented here relies on the MatLab™ (MathWorks™) [95] software as the scripting environment for interacting with the toolkit functions. The MatLab environment was selected due to its powerful data manipulation routines and because it can access specialized computational libraries written in compiled languages such as C++, Java and FORTRAN. In addition, it supports object-oriented programming (OOP) paradigms such as classes, inheritance, polymorphism and scoping through modular packages. The example shown in Fig 6.1 shows how a set of simple function calls in a MatLab script is used to load, control and visualize a finger object.

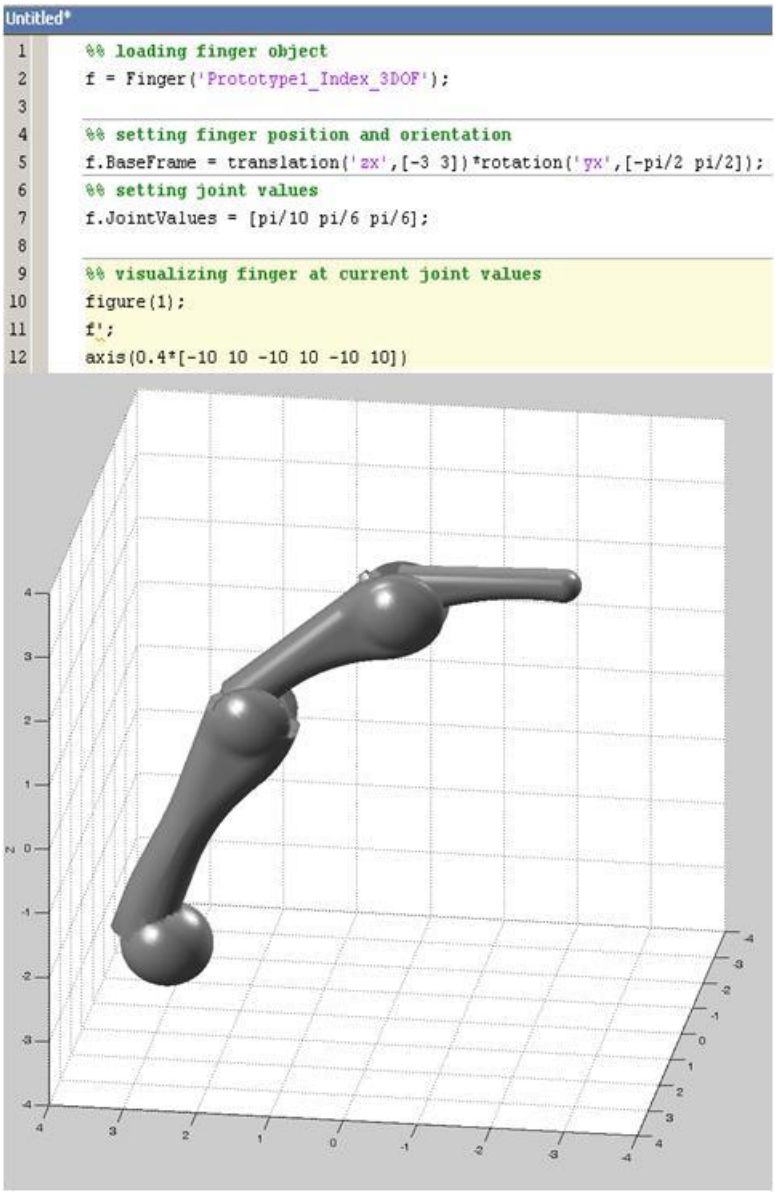


**Figure 6.1: Script that controls and visualizes a finger object.**

In particular, the toolkit obtains the most benefit out of scripting during the use of the physics simulator program; which has been incorporated as an additional component of the Robotic Grasping Toolkit. Through a script, a user can immerse an object within a virtual environment that runs inside the simulation software in order to observe how the object behaves in the presence of physical phenomena such as gravity, friction and inertia. Furthermore, data

can be sent back and forth between the scripting environment and the physics simulator so as to provide a very minute level of interaction between both ends. The details of the physics simulation program will be described further in this chapter.

## 6.2 Object Oriented Programming towards Modularity

The Robotic Grasping Toolkit exploits MatLab's object-oriented programming capabilities in order to achieve a great deal of flexibility and modularity. OOP is a programming methodology in which a computer object or data structure defines a set of data fields and methods that pertain to a particular task. In this toolkit, each category of functions is encapsulated inside an object class so as to make their access easy and intuitive; thus each class can be thought of as an independent module that performs a unique set of computations.

Furthermore, some of the classes implemented in the toolkit contain associated databases that are used to save and load objects (instances of the class) with a particular set of characteristics. For instance, the rigid body classes use their corresponding database to store data such as triangulated meshes, color, mass, inertia and many other distinct attributes. In order to facilitate the access to class databases, the toolkit implements a simple but efficient data management mechanism that is handled by a single class which has been named "Toolkit". Relegating all data management tasks to a single class alleviates other classes in the toolkit from the need to implement their own data administration functions. Hence, all classes access their data by making calls to functions implemented in the Toolkit class. The Toolkit class also favors extendibility by allowing new classes to administer their own data with a minimum amount of code.

## 6.3 Graphical Object Representation

Each rigid body object in the toolkit is represented by geometry, mass parameters and spatial configuration data. Hence, the "Solid" class has been implemented in order to encapsulate all the details necessary to describe a rigid body object. However, the main role of the Solid class is to perform the majority of low level graphical functions in order to minimize user intervention during graphics manipulation. All other classes that present some form of graphical representation inherit most of the graphical functionality directly from the Solid class. Thus, inheriting makes it possible to reuse existing graphical routines instead of rewriting the same graphical functions for each class.

The Solid class can manipulate geometries that are provided as a list of vertices and triangular facets; just as presented by "STL" files. Once the geometry is available as a triangulated facet list, Solid can perform a multitude of operations such as scaling, animations and spatial transformations. In addition, the Solid class can combine objects together so as to form new ones that contain composite geometries. This capability to combine objects was found most useful during the incremental growth of the obstacle environment used by the grasp synthesis method described in chapter 5.

Furthermore, objects of the Solid class are used as a fundamental building block for others classes that represent more complex and specialized entities. For instance, the rigid body classes, that are associated with a geometric primitive such as spheres or boxes, are considered specialized cases of the Solid class. Moreover, the "Link" and "Palm" class extend the characteristic of the Solid class in order to acquire additional capabilities. As an example, a Link object is just another Solid object with a few more properties such as joint parameters and kinematic configuration. In addition, there exists other classes that contain multiple instances of

the Solid class so as to define articulated bodies; such is the case of the “Finger” and “Hand” classes. The image in Fig.2 shows a script that loads and manipulates “Solid” objects.

### 6.4 Articulated Bodies

The Robotic Grasping Toolkit supports the use of mechanisms composed of several rigid bodies that are connected by joints. Hence, the articulated body classes in the toolkit are responsible for controlling and managing the connectivity between several “Link” objects that form a mechanism. The toolkit has three articulated body classes and these are: “Articulated Body”, “Finger” and “Hand”. The Articulated Body class implements the necessary functionality associated with mechanical joints, so that other classes, like the Finger class, can borrow this functionality. In these classes, the relative spatial relationship between the links of an articulated body is enforced in accordance to the kinematic configuration defined by the Denavit-Hartenberg (DH) convention [96]. Some Finger objects are shown in figure 6.3.

```
%% creating and moving "Solid" object
s(1)=Solid('flange');
s(1).scale(50);
s(1).Position=[0 4 0]';
s(1).Quaternion=Solid.transform2quatern(Solid.rotation('x',-pi/3));

%% instantiating second "Solid" object
s(2)=Solid('Sprocket.STL');
s(2).Position=[0 -4 0]';
s(2).Quaternion=Algorithms.transform2quatern(Solid.rotation('x',-pi/3));

%% creating Box
s(3) = Box([2 2 2]);
s(3).Position = [0 4 4]';

%% creating sphere
radius = 2.5;
s(4) = Sphere(radius);
s(4).Position = [4 3 4]';
s(4).Color = [0 1 1];

%% displaying objects on drawing panel
figure(1);
s';
grid on
```

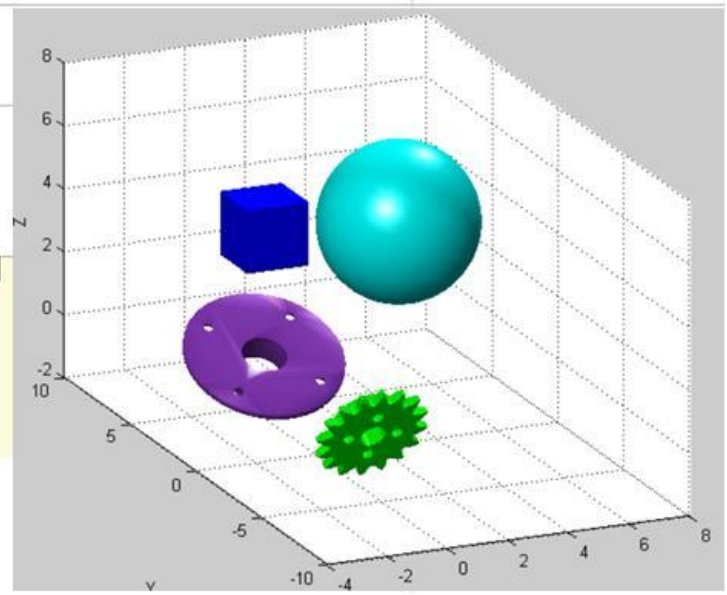


**Figure 6.2: Creating and manipulating "Solid" objects.**

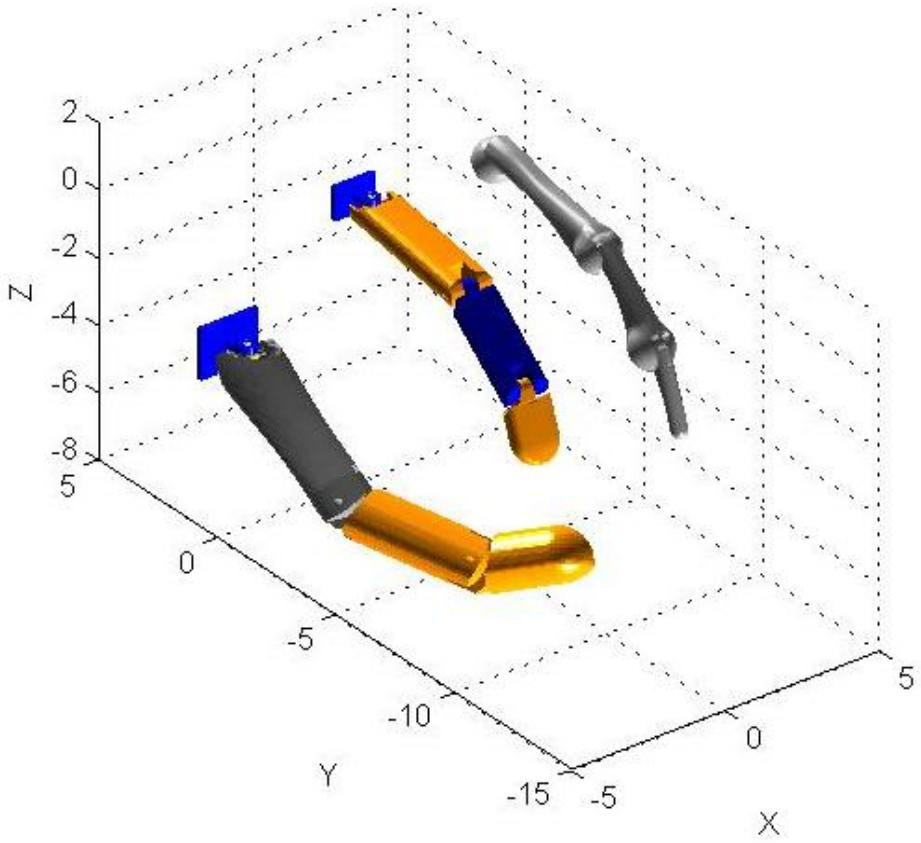


**Figure 6.3: Several Finger Objects.**

The "Hand" class is used to represent mechanisms that are composed of multiple articulated bodies; such as the majority of robotic hand devices where a base is connected to multiple fingers. Thus, a Hand object contains a single Palm object that connects to one or several Finger objects; each finger may have any number of links and kinematic configuration. The particular design of the Hand class is flexible enough to accommodate virtually any hand model; there are currently three hand models stored in the toolkit database and they will be described further in this chapter. Once a hand model is loaded, the user may control the position and orientation of the hand as well as each individual joint in the links.

## 6.5 Forward Kinematics

The use of forward kinematics allows determining the relative spatial placement of the links that belong to an articulated mechanism as dictated by the joints [96]. The Robotic Grasping Toolkit relies on the Denavit-Hartenberg (DH) convention in order to describe the kinematic relations that govern a particular articulated body. Hence, the DH notation correlates the relative position and orientation of a link to a set of parameters that specify the kinematic configuration of the link. The DH parameters of the $i^{th}$ link are as follows: Link length $a_i$, link twist $\alpha_i$, joint angle $\theta_i$ and link offset $d_i$. In general, revolute or prismatic joints are supported by the DH notation. Whenever a revolute joint is selected, the joint angle $\theta_i$ is made into a variable and the other parameters are fixed; this implies that the relative position and orientation is a function of the joint value. If a prismatic joint is used, the link offset $d_i$ is turned into the joint variable and so the other parameters remain at a fixed value. The following equation demonstrates the mapping between the DH parameters and the spatial configuration of the link:

$$F_i = Rot_x(\theta_i) \times Transl_z(d_i) \times Transl_x(a_i) \times Rot_x(\alpha_i) \tag{6.1}$$

$$F_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i c\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Furthermore, an additional "universal" joint has been incorporated in the toolkit presented here; this joint type is not directly supported by the DH notation. The universal joint allows two degrees of freedom and it is best described as two coupled revolute joints with different rotational axes. This type of joint has been found convenient for describing the abduction-adduction (AA) and flexion-extension (FE) movements observed in human fingers [97].

In this toolkit, forward kinematics routines have not been incorporated as a separate module as the rest of the utilities. Instead, they've been embedded into the code that defines the behavior of each articulated body class so as to give them the ability to resolve their own kinematic configuration. Thus, there is no need to perform separate kinematic computations, although the forward kinematic results may be retrieved from the corresponding articulated object at any time. In addition, a number of support classes such as "Joint Configuration" and "Joint Profile" implement convenient mechanisms for generating path generation data that may be used by an articulated body object at any time. The image in Fig 6.3 shows how a Finger object computes its own kinematic results.

```
>>  f = Finger('Prototype1_Index_3DOF');
>> f.DH % display DH parameters

ans =

    3.8100         0         0         0
    2.7000         0         0         0
    2.2600         0         0         0

>> f.JointValues = [pi/10 pi/6 pi/6]; % set joint values
>> f.Links.Frame % obtain the transformation matrix of each link

ans =

    0.9511   -0.3090         0    3.6235
    0.3090    0.9511         0    1.1774
         0         0    1.0000         0
         0         0         0    1.0000


ans =

    0.6691   -0.7431         0    5.4302
    0.7431    0.6691         0    3.1838
         0         0    1.0000         0
         0         0         0    1.0000


ans =

    0.2079   -0.9781         0    5.9001
    0.9781    0.2079         0    5.3945
         0         0    1.0000         0
         0         0         0    1.0000

>>
```

**Figure 6.4: Forward Kinematics Computations performed by Finger Object.**

## 6.6 Current Hand Models

The three hand models that currently exist in the toolkit are original designs that belong to the Robotics and Intelligent Machines (RIM) laboratory at the University of Texas in San Antonio. Neither one of these hand models has been made into a prototype yet but their CAD models can be use to perform numerous experiments such as those that were conducted in chapter 5.

The "Three Finger" hand model features a very simplistic design and contains three fingers that connect to the same base at equally separated locations. All the fingers are identical and contain three links of 10 units long each. Each finger possesses three degrees of freedom (DOF), which makes up a total of 9 degrees of freedom for the entire hand. This hand model resembles some commercially available three-jaw grippers, which are commonly used to perform pick and place operations. An image of the Three Finger hand is shown in figure 6.5.

The "Rowdy" hand is the second hand model of the list and features a quasi-anthropomorphic design. It contains a total of four fingers, where the first three fingers connect to the top part of the palm and the last finger is placed at the opposite end of the palm. The three fingers on top are identical and consist of three links and 4 DOFs; where the second joint allows abduction and adduction (AA) motions while the other three joint are used for flexion and extension (FE) of the links. The last finger serves as an opposable thumb and it is of larger size than the other three fingers. Its first revolute joint performs AA movement while the other three joints allow FE motions. The first articulation of each finger uses a universal joint in order to model the two coupled revolute joints of the first link. The Rowdy hand is shown in Fig. 6.6.

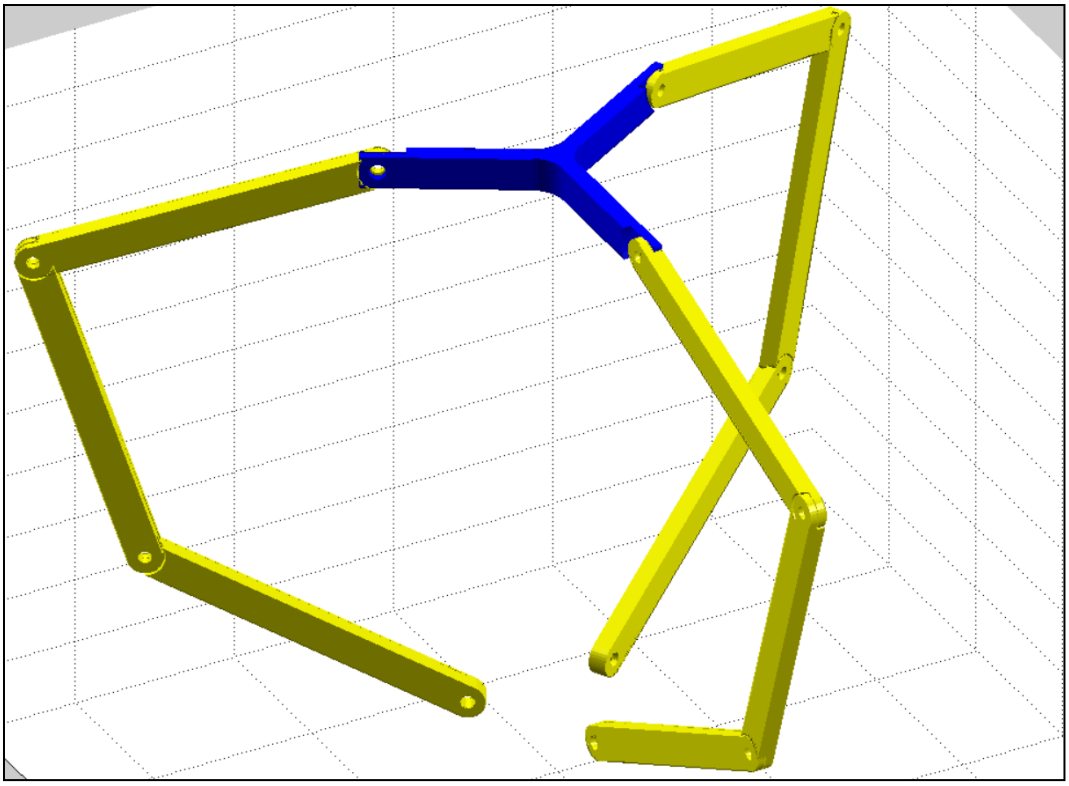

**Figure 6.5: The Three Finger hand model.**

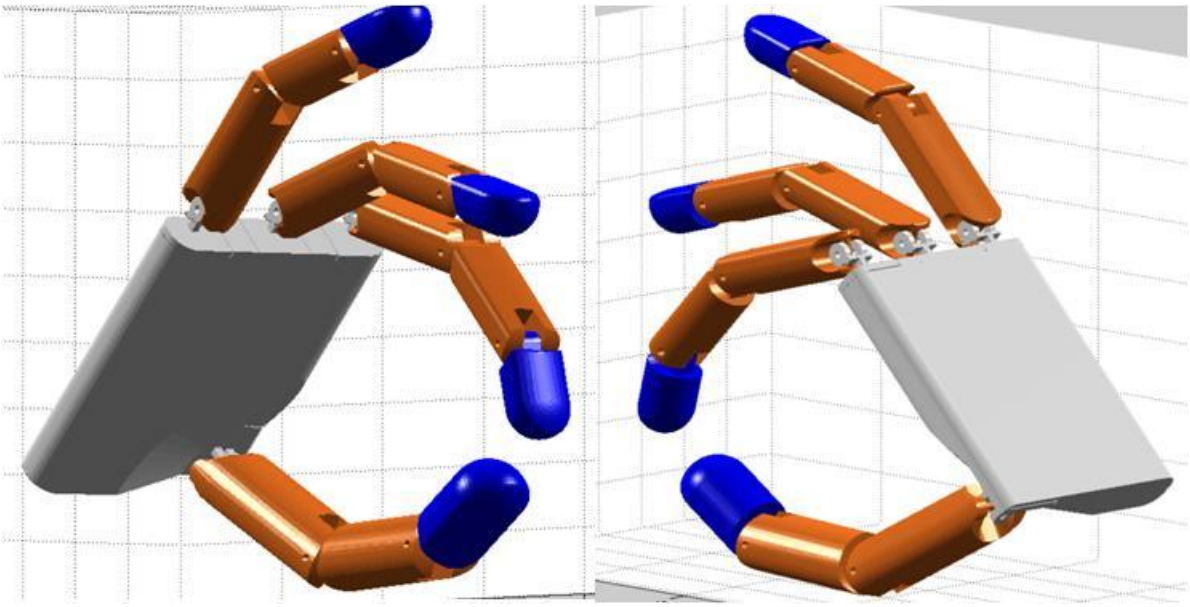

**Figure 6.6: The Rowdy hand.**

Lastly, the Prototype1 hand shares many similarities with the bone structure that makes up an average human hand. It has four regular fingers and an opposable thumb that is located at the bottom cavity of the palm. There are 4 DOF in each finger including the thumb; where the first joint permits AA motions and the last three are used for extending or flexing (EE movements) the finger. The palm contains spherical cavities that hold the base of each finger and it presents large gaps in many regions. The Fig 6.7 displays the image of the Prototype1 hand.

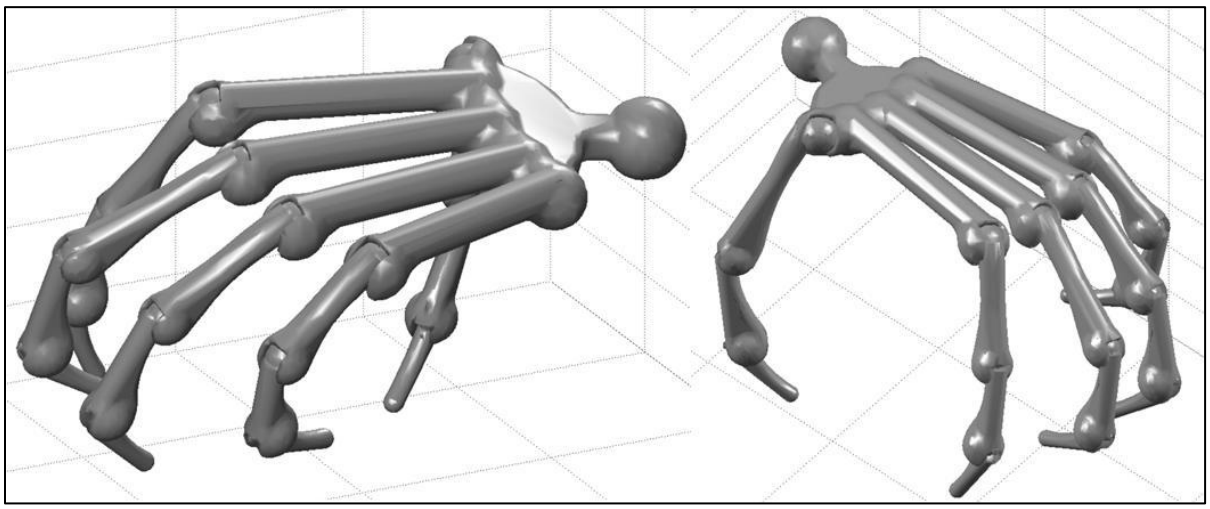

**Figure 6.7: The "Prototype1" hand.**

## 6.7 Collision Detection

The detection of possible collisions between the fingers and the object is a key aspect of the grasp synthesis method presented here. The need to provide this capability led to the creation of the Collision Detection module, which main component is the "Continuous Collision Detection" class. Objects of this class can perform continuous collision detection computations (CCD) for any two rigid bodies with predefined motion. Thus, a time of collision (TOC) value along with contact features are determined if a collision is reported at some point during the motion. The actual CCD computations are performed by routines in the Conservative Controlled Advancement (C2A) programmatic library for the C++ language that was presented in [2]. In order to make the C2A routines available to MatLab, a dynamically linked library (DLL) was created by combining the MatLab's application programming interface (API) for C++ with the corresponding functions in the C2A library. In addition, the collision detection system from the Bullet Dynamics Engine [8] was integrated into the same DLL file so as to detect and report all the contact features associated with a collision. In general, C++ programs are orders of magnitude faster than MatLab programs and so the overall performance of the CCD routines in this toolkit is very fast.

Objects from the Continuous Collision Detection class accept Solid, Link and Palm objects implemented in the toolkit and so this made it possible to apply *RecursiveCCDSearch* routine to Finger objects; this routine is used by the grasp synthesis method described in chapter 5. Consequently, the *RecursiveCCDSearch* routine has been made available as part of the Continuous Collision Detection class. In addition, the incorporation of several support classes allows encapsulating collision data in a convenient manner. For instance, the "Contact Point" class stores contact location, contact normal, friction values and penetration depth data within a

single object. The “Contact Pair Result” class is useful for associating multiple Contact Point objects from a single collision between two Solid objects. Furthermore, collision results associated with an articulated body are managed by objects from the “Collision Group Result” class. The Fig 6.8 shows how CCD is used to resolve a collision situation between two objects.

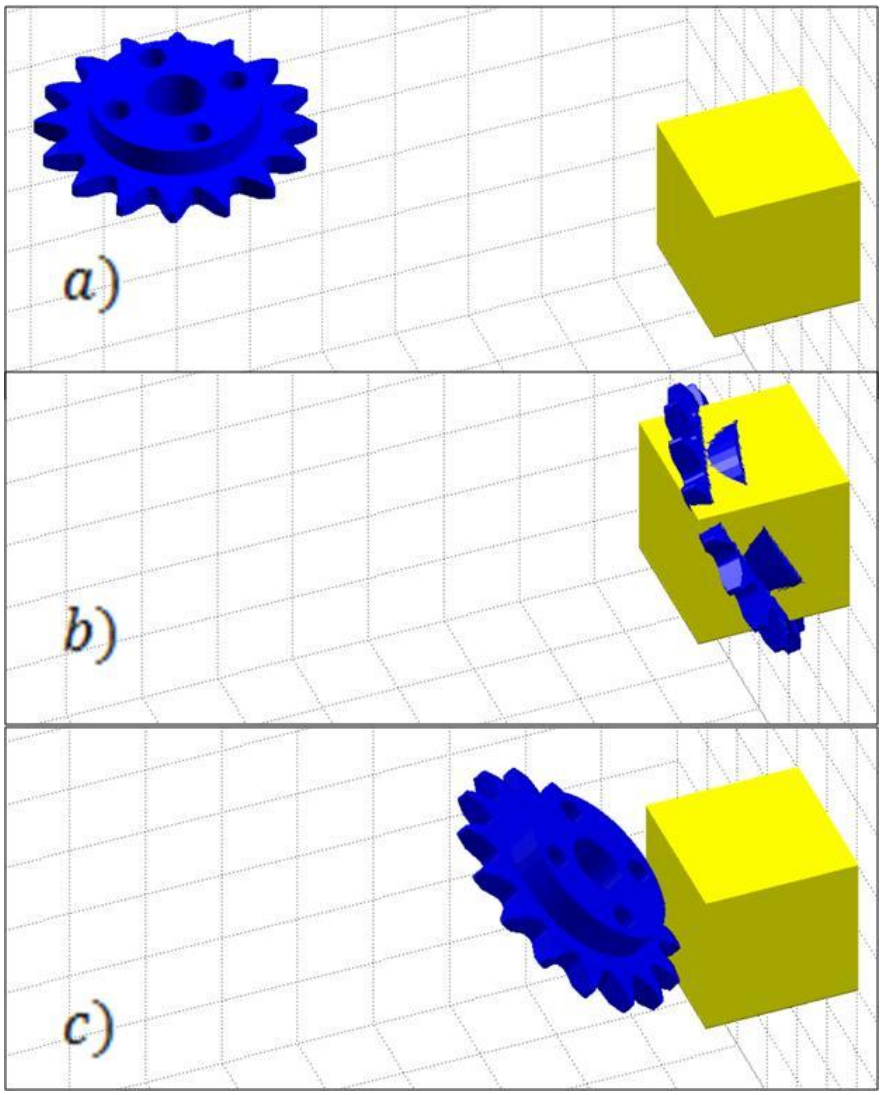


**Figure 6.8: CCD computations for two “Solid” objects.** Case a) corresponds to the initial configuration, case b) indicates the final configuration and case c) indicates the configuration at TOC.

## 6.8 Computational Geometry

The computational geometry module allows conducting several useful operations for geometries expressed in N-dimensional space. One of the most recurred operations is that of convex hull; this operation is available within the very popular computational package known as "Qhull" [49]. MatLab provides a simple implementation of Qhull within its default set of libraries. However, it was determined that this implementation was insufficient for the objectives of this work, since only a subset of the results computed by Qhull are passed back to the MatLab environment. In order to solve this problem, the original Qhull library, available in the C++ programming language, was embedded into a DLL file so as to make it accessible from MatLab. Consequently, the Qhull version in the Robotic Grasping toolkit can return facet lists, neighboring facets, offsets distances, normal vectors, area and volume results associated with a convex hull computation. Of special interest are the offset distances and normal vectors since they allow defining hyperplanes; the use of hyperplanes was a key aspect of the novel grasp synthesis technique presented in this thesis.

On the contrary, other programs such as the Ray-Shooting algorithm, GJK distance algorithm and the support mapping operation were not available elsewhere and so their mathematical methods had to be implemented from the ground up. As recalled, the Force-Closure test explained in chapter 4 required these two algorithms in order to determine if the origin was located in the interior of the 6-dimensional wrench space. Also, numerous support mapping operations were utilized for determining support points from the linearized friction cones during the grasp synthesis technique of chapter 5. In addition, the incorporation of the "Hyperplane" class facilitated conducting hyperplane tests in which the half-space corresponding to the location of a point was to be determined.

In this work, the operations available within the computational geometry module have been used to process the 6-dimensional wrench space that is produced by a grasp. However, they are not limited to this particular application and so they can be applied in more general tasks that involve 2-dimensional or 3-dimensional objects.

## 6.9 Grasp Analysis

Several computations related to the concepts of grasp mechanics have been made available in the toolkit. Hence, there are two modules for conducting grasp analysis calculations and these are the Wrench Space and Grasp modules. The Wrench Space module can generate useful constructs that allow representing important theoretical entities such as the wrench space, wrench vectors and friction cones. In particular, the Wrench Space class has been given sufficient functionality so as to evaluate and report the current state of relevant properties, such as Force-Closure and grasp quality, which define the wrench space of a given grasp. In addition, Wrench Space objects may perform a number of convenient tasks such as support mapping, convex hull and hyperplane generation. In fact, the ability to create a hyperplane, passing through a given facet of the wrench space, simplified the implementation of the novel grasp synthesis technique presented previously in this thesis.

Moreover, the wrench vectors associated with a friction cone can be placed into a "Wrench Set" object; this class definition belongs to the Wrench Space module as well. A Wrench Set object takes a contact and then maps the forces in the contact's friction cone into the corresponding wrenches in the 6-dimensional space of generalized forces. Thus, this class can treat a collection of associated wrench vectors as a single entity. It is important to maintain the association of the wrench vectors that belong to a friction cone so as to make an adequate

assessment of the contribution of a contact in the formation of a grasp. Also, instances of the Wrench Set class can be added or removed from a Wrench Space object in order to alter the properties of the wrench space.

The other component for conducting grasp analysis is the "Grasp" module. This module contains functions that help resolve grasp synthesis problems and so it relies on the constructs that are available in the Wrench Space module. Consequently, the two stages of the grasp synthesis technique described in chapter 5 have been implemented as separate routines inside the Grasp module. Moreover, both of these grasp synthesis functions are compatible with instances of the Finger and Hand classes carried inside the toolkit. Thus, the grasp synthesis methods in the Grasp module can produce adequate grasp configurations for any hand model available in the toolkit.

So, the functionality that is available in the Grasp and Wrench Space modules covers general grasp analysis tools and so both modules can be used to develop and evaluate new grasp methods. Additionally, the grasp analysis and hand representation capabilities make the Robotic Grasping Toolkit a suitable tool for evaluating the performance of hand mechanism carrying out a multitude of grasping tasks.

## 6.10 Physics Simulator

To truly evaluate the performance of a hand mechanism, it is necessary to reproduce close approximations of typical work environments so as to expose all the elements that affect the completion of a grasping task. In order to meet this necessity, a physics simulation program was developed and incorporated as an extension of the Robotic Grasping Toolkit. This physics simulator incorporates multiple programming libraries, but the core functionality has been

implemented by means of the Bullet Physics Engine [8] available for the C++ language. The Bullet Physics library features very robust dynamics solver and collision detection systems that allow creating very precise and reliable physical systems.

Nevertheless, combining the functionality of the Robotic Grasping Toolkit and the Physics Simulation software into a single package required a very long and thorough design phase. Hence, a number of classes and functions were produced in the C++ language so as to mirror equivalent classes that had been implemented in the MatLab environment. These C++ classes made it possible to place objects from the toolkit inside the physics simulation environment. Then, the entire program was compiled into a DLL file in order to make it accessible from the Matlab scripting environment.

The "Physics World" class in the toolkit is the main channel for accessing the physics simulation program. This class provides very intuitive mechanisms for adding or removing objects from the simulation at any given time. Furthermore, the objects added to the simulation can be flagged as kinematic, static or dynamic; each of these states is indicative of an object's response to simulated physical phenomena. In fact, there is tremendous amount of flexibility on how the user can place the objects inside the virtual environment. Several actions such as pausing, starting, closing or stepping the simulation can be controlled by the Physics World object as well. The images in Fig. 6.8 and Fig 6.7 show a MatLab script and the corresponding simulation results respectively.

The physics simulator is a great addition to the Robotic Grasping Toolkit since it allows placing existing hand models inside realistic work environments. Thus, a great deal of observations about the design of a hand model can be made so as to introduce several adjustments that aim to increase performance of the hand device. Furthermore, more complex

problems in robotic grasping, such as grasp planning and dexterous manipulation, can be thoroughly explored with this toolkit as well. Lastly, the control design tools available in MatLab and Simulink can be combined with the Robotic Grasping Toolkit in order to test and develop robotic hand systems.

```
%% creating physics world
world=PhysicsWorld;

%% setting sphere as ground
s=Sphere(25);
s.IsDynamic=false;
s.Frame=Solid.translation('z',-22);
world.GroundObject=s;

%% creating objects to be added to the physics simulation;
b(500)=Box;
l=1; % side length
for nx=1:5;
    for ny=1:5;
        for nz=1:20;

            b(ind)=Box([l l l]);
            b(ind).Frame=Solid.translation(...
                'zxy',[20-2*(l)+(nz-1)*3/2*l ...
                -2*(l)+(nx-1)*3/2*l ...
                -2*(l)+(ny-1)*3/2*l]);
        end
    end
end

%% adding objects to the world
for n=1:length(b);
    world.addBody(b(n));
end
%% running simulation
world.startSimulation;
%% pause simulation
world.pauseSimulation;
%% resume simulation
world.resumeSimulation;
%% stopping simulation
world.stopSimulation;
```

**Figure 6.7: Physics simulation script.** This script adds 500 boxes and starts the simulation.

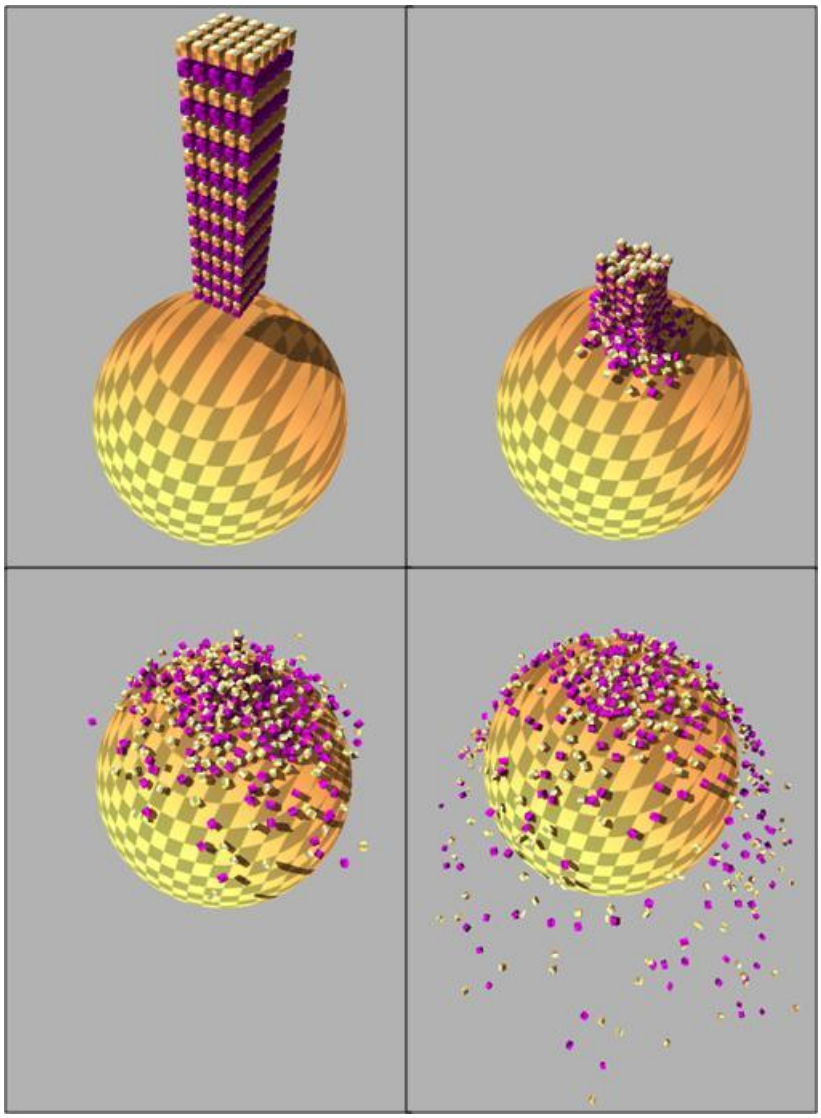

**Figure 6.8: Snapshots of physics simulation results.**

An illustrative example of a grasping operation is given in Fig 6.9. This image shows the Three Finger hand making a simplistic pick and release operation. Thus, the hand makes a descend movement until its base touches the sphere. Once there, it closes its fingers until the sphere is completely trapped. Then, it carries it to a new location at which point the sphere is released. In conclusion, this example serves to demonstrate how the sphere is affected by the forces exerted onto it by the fingers. A more elaborate example is discussed in chapter 7.

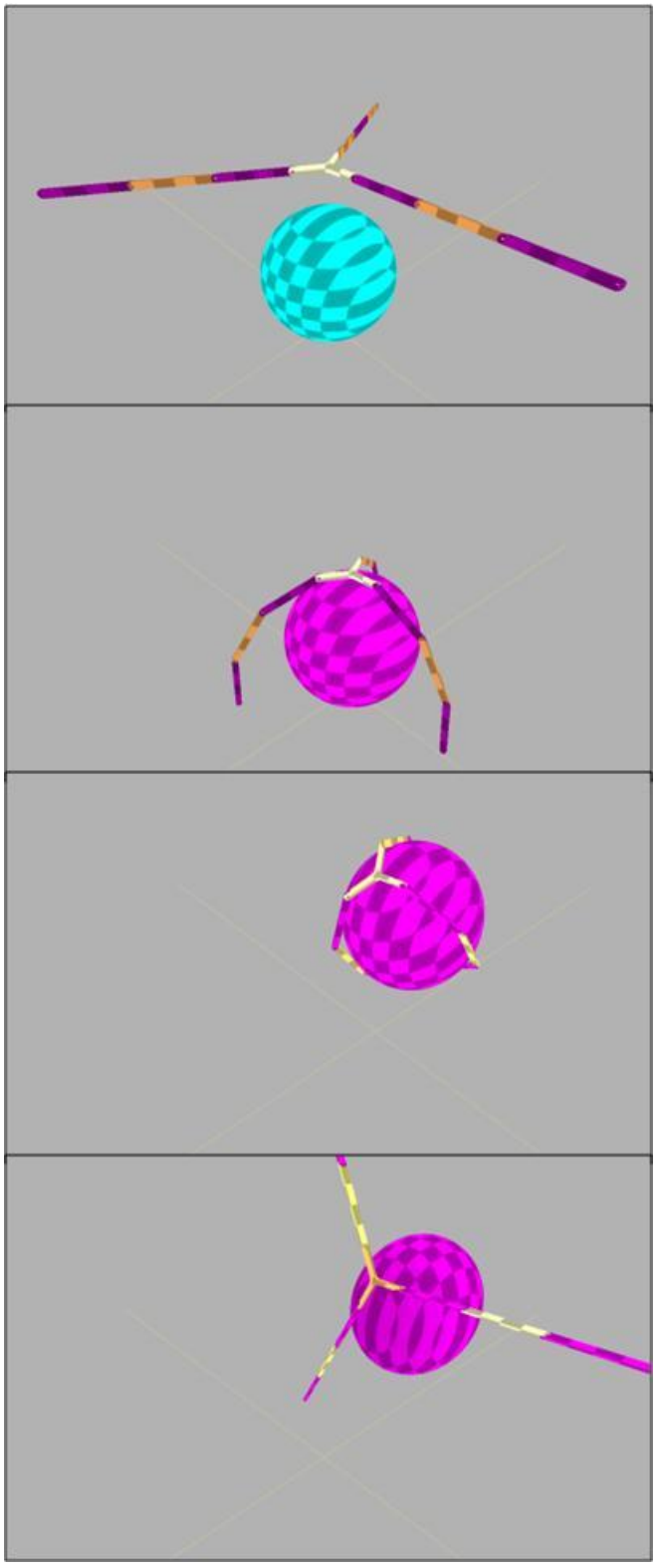

**Figure 6.9: Three Finger hand grabbing and releasing sphere.**

## 6.11 Toolkit Overview

As stated previously, the Robotic Grasping Toolkit exploits object-oriented design practices in order to categorize functions within individual modules. Hence, it was necessary to produce a proper arrangement of these modules so as to allow easy access to their corresponding functions. Furthermore, the arrangement used throughout the toolkit facilitates the communication between the MatLab programs and the many routines that were implemented in the C++ programming language. The Fig 6.10 shows a very simplified schematic of the organization exhibited by the toolkit.

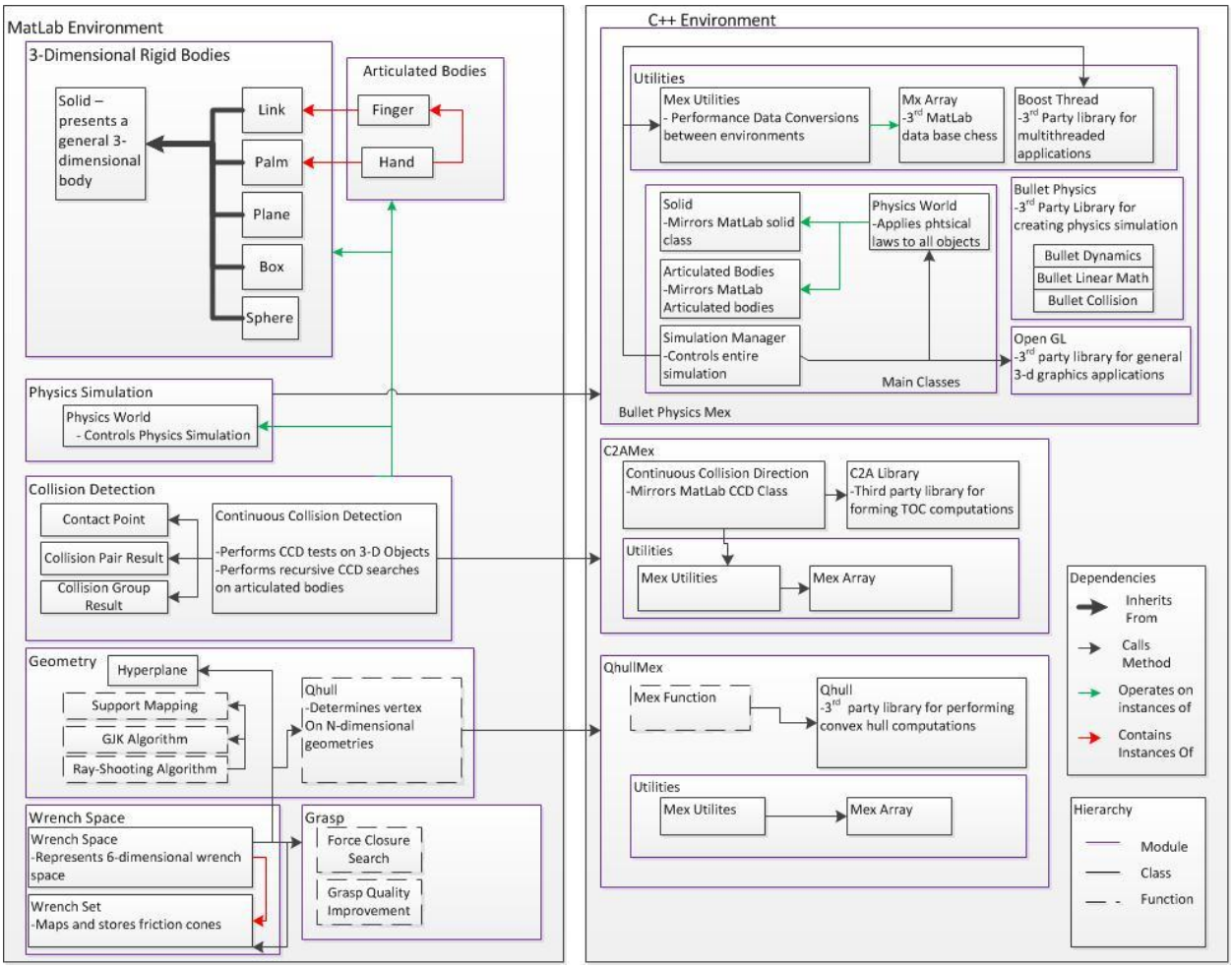


**Figure 6.10: Robotic Grasping Toolkit Flowchart.**

Furthermore, it became necessary to implement a number of internal classes in order to perform several essential tasks. So, these classes are not accessible to the user but their presence facilitates the utilization of various routines and resources. For instance, the "Mex Utilities" class, implemented in C++, allows converting MatLab data into compatible types that were understood by the C++ programming language. This class also performed data conversions in the reverse direction and so dataflow in both directions was achieved. However, the user never interacts with this class or its functions directly.

In addition, the 3rd party programming library "Boost" was relied upon in order to incorporate multithreading capabilities. In this case, these capabilities allowed relegating the execution of the simulation program into its own process thread, instead of interrupting MatLab's main processing thread. Thus, it became possible to run the physics simulator while performing other computations simultaneously, all within MatLab. If only one processing thread was available, this operation would've not been allowed.

# CHAPTER 7: SIMULATING A GRAPING TASK

The purpose of this chapter is to combine the grasp synthesis technique with the physics simulator in an experiment that aims to demonstrate the benefit of these tools for studying grasping problems. In addition, this experiment exposes the advantages of scripting for generating very elaborate details that define a task. Consequently, the details for the entire experiment were specified within a single script that utilizes a number of Matlab array processing routines in conjunction with several routines available in the Robotic Grasping Toolkit.

This experiment consists of using a hand mechanism for arranging a total of 16 cubic blocks into a tower like structure. This tower contains four levels which are built on top of each other; each level contains 4 blocks that are placed into a square arrangement. Furthermore, every new level is tilted 45 degrees in relation to the preceding level so as to increase the complexity of the tasks. Thus the hand must grasp each block at its starting location and transport it to destination while avoiding collisions with the structure under construction.

At first, the grasp between the hand mechanism and the block was computed by means of the grasp synthesis technique introduced in chapter 5. The hand model employed for this experiment was the Rowdy hand, while the cubic block was constructed by scaling a box to a size of 6 units wide, 32 units long and 8 units high. The resulting grasp showed a grasp quality of 0.481 and it was computed in a total in 28.7 seconds. An image of the grasp is shown in Fig. 7.1.

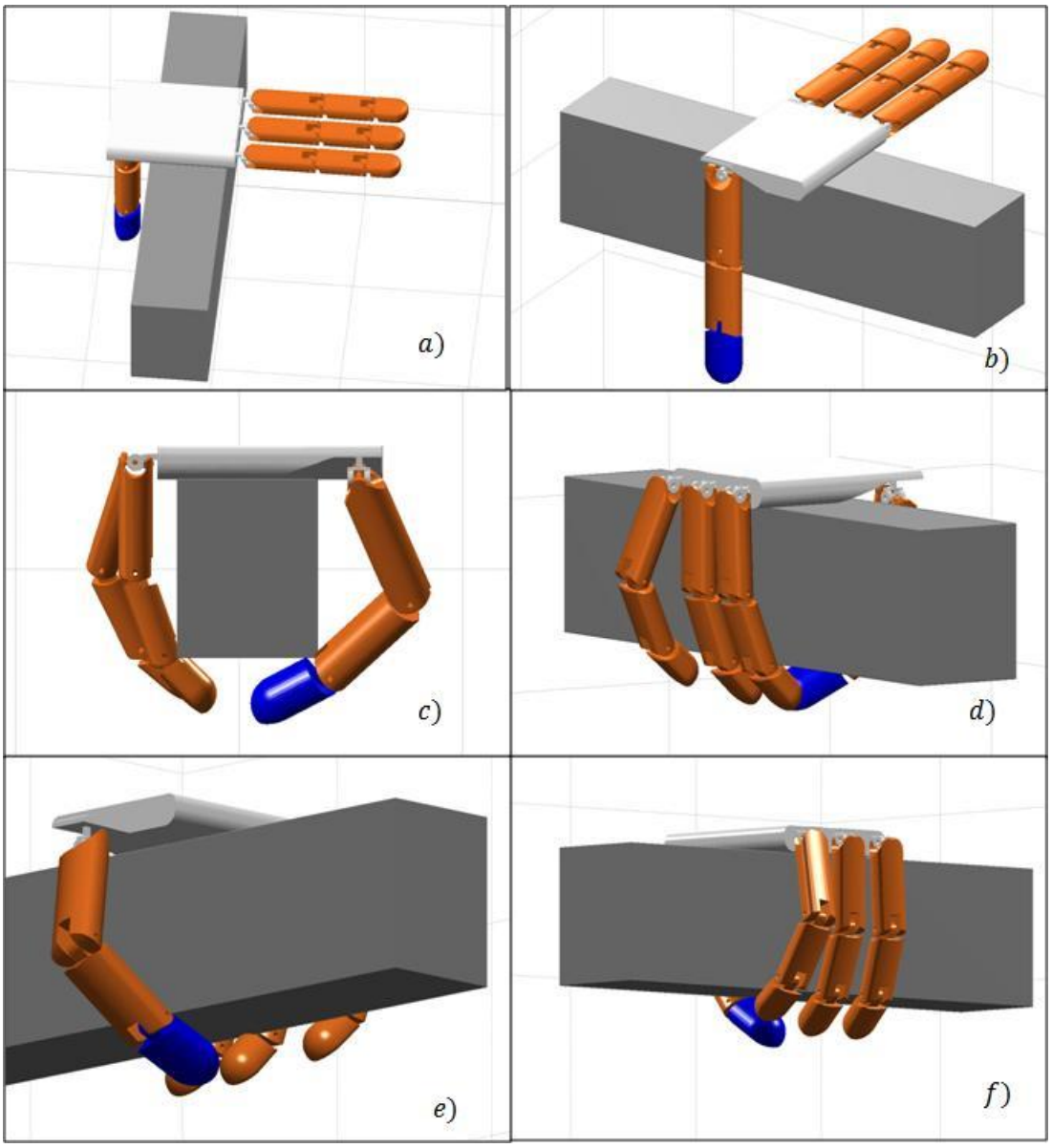


**Figure 7.1: Rowdy hand grabbing a rectangular block.** The images a) and b) correspond to the open hand configuration. In c), d), e) and f) different views of the computed grasp are shown.

Once a valid grasp was produced, a transform was computed and stored in a MatLab array so as to retain the relative spatial configuration of the hand in relation to the block. Also, the hand pose corresponding to this grasp was allocated into a MatLab variable as well. By

maintaining this data, it was possible to recreate the grasp at multiple times during the execution of the task in the simulator.

Next, the initial arrangement of the16 blocks consisted of a formation of two rows with 8 blocks each. The final configuration for each block was predetermined as well with a simple iterative heuristic. Hence, the final position and orientation of all the blocks constituted the structure of the tower to be created.

At last, the task was executed by means of an iterative procedure. Each iteration began by grasping the target block at its initial location with the hand. Hence, the hand position and orientation were computed in relation to the target block so as to reproduce the grasp properly. This computation was carried out by means of the transform data that had been obtained during the initial planning steps.

Once the hand had secured the target block, it ascended to a predetermined height while maintaining the grasp. Then, the hand was rotated about the world coordinate z-axis through a predetermined angle; this angle corresponded to the final orientation of the block in the tower structure. Next, the hand performed a linear displacement along the x-y plane until the target block was located right above its final location. The last actions consisted of a slow descend and release movements that allowed placing the block at its destination without disrupting the existing structure. This series of movements were made until all four levels of the tower structure were completed. The set of images corresponding to this experiment are shown from Fig. 7.2 to Fig. 7.5.

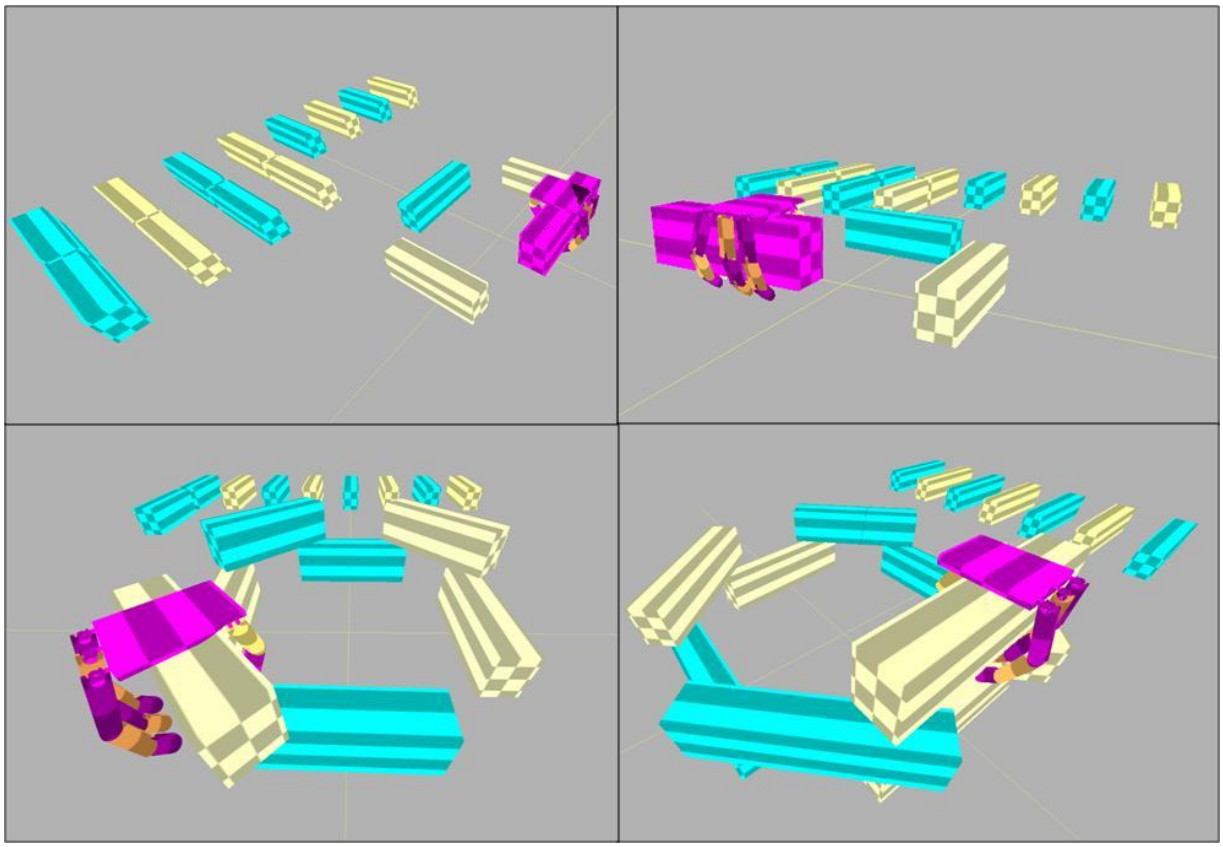

**Figure 7.2: Hand constructing first and second level of tower structure.**

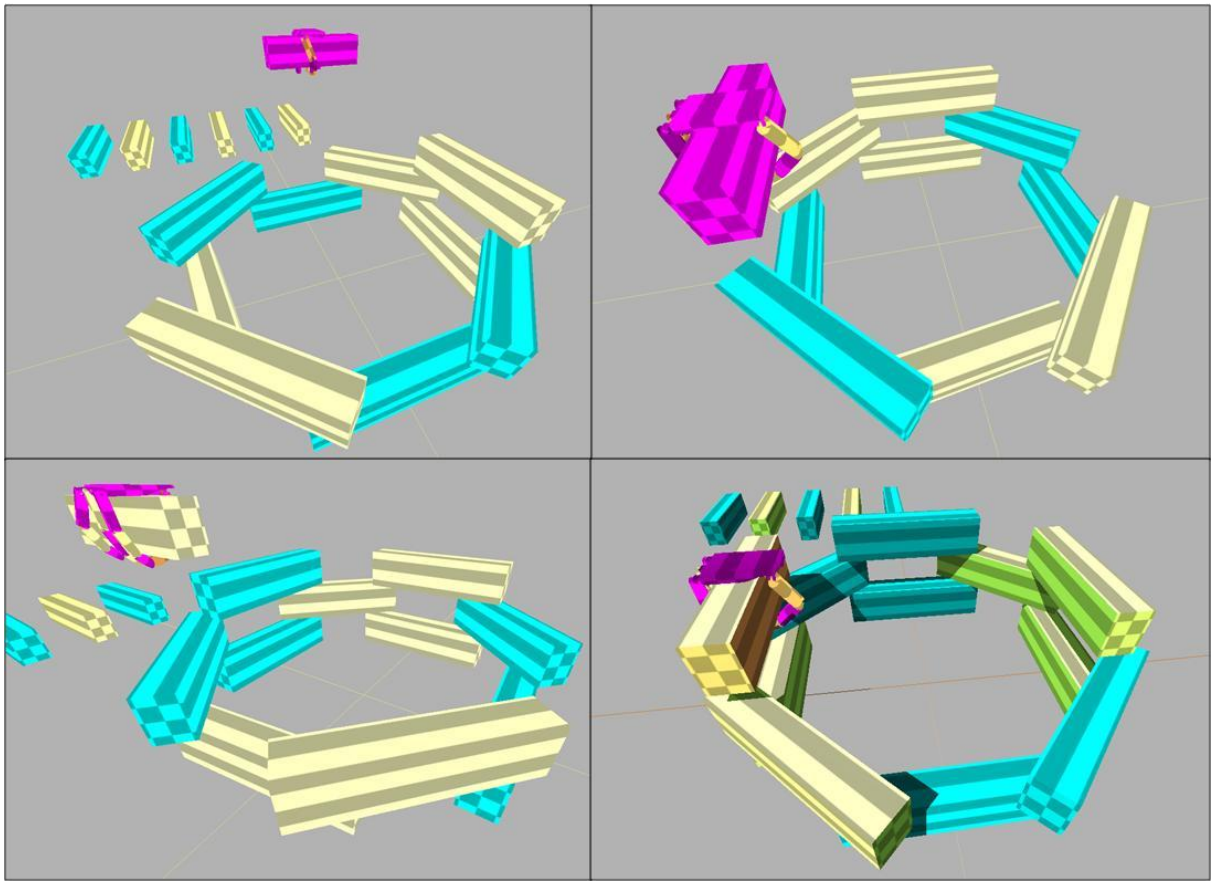

**Figure 7.3: Hand constructing second and third level of tower structure.**

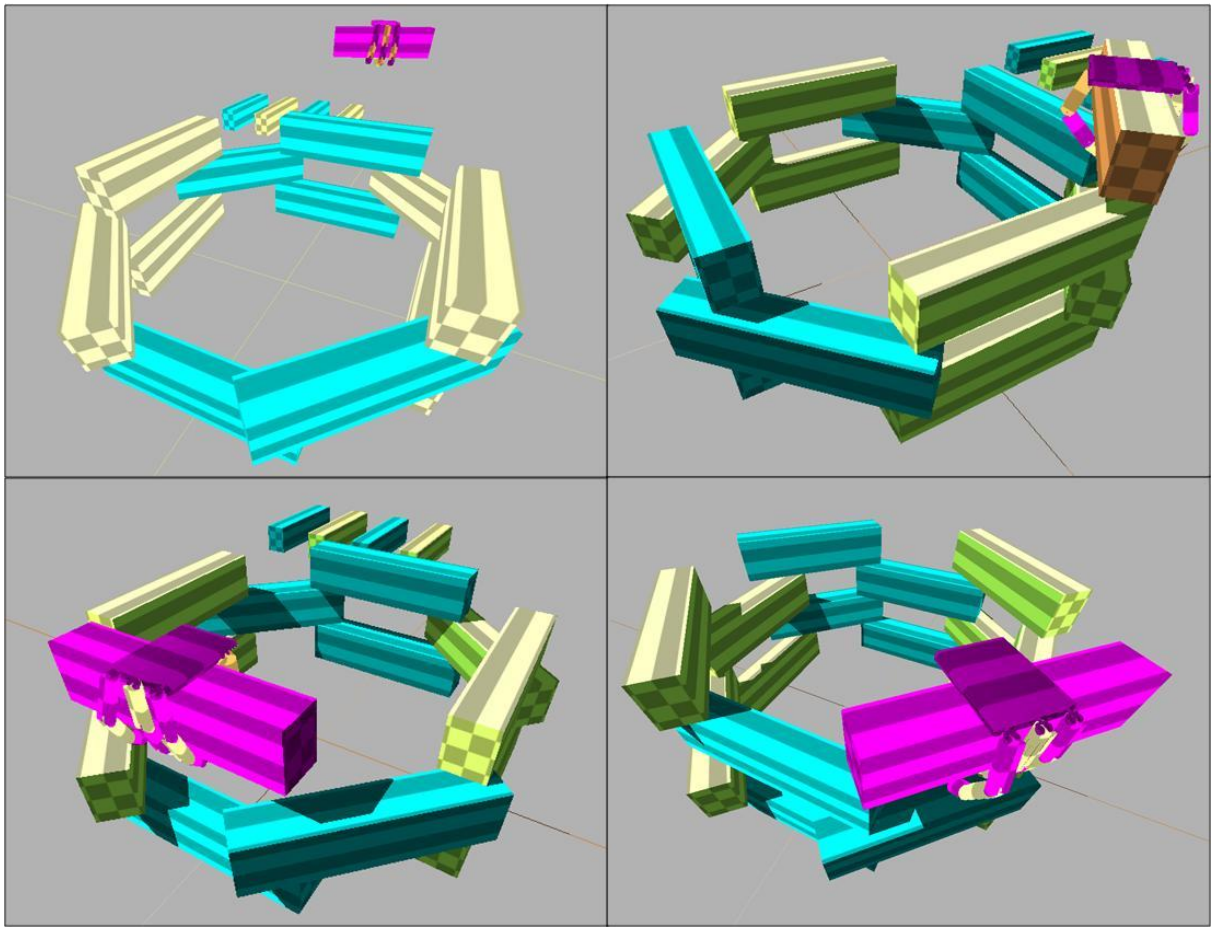

**Figure 7.4: Hand construction third and fourth level of tower structure.**

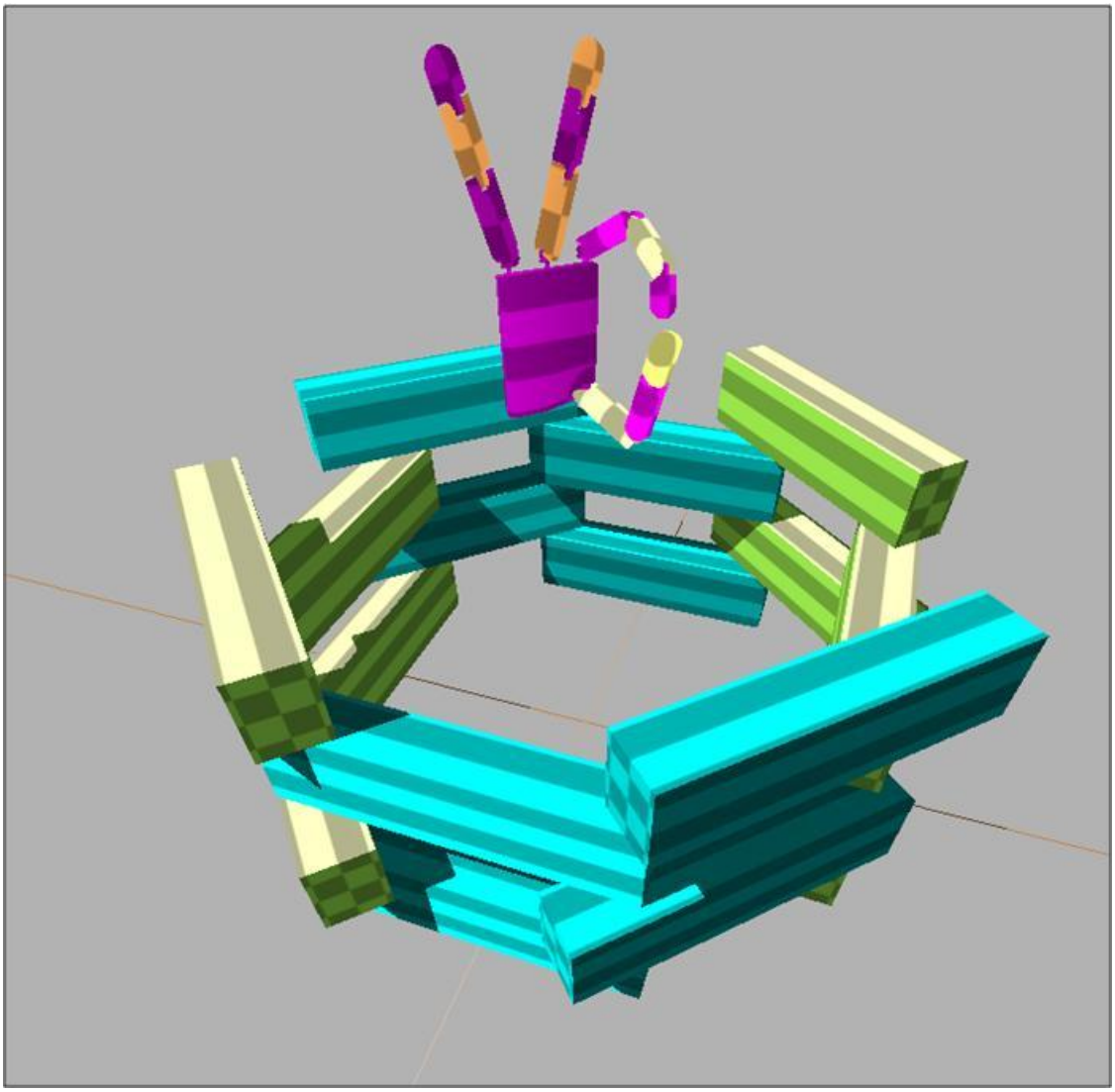

**Figure 7.5: Tower construction completed by hand.**

The movements described in this experiments required a great deal of path planning so as to ensured that the tower was not perturbed by any of the objects in motion during the its construction. In this case, planning these movements became rather intuitive once the placement of all the components was well understood. Despite any initial path planning and preparations, this task could've not reached a successful end if a suitable grasp was not available from the start. Consequently, this demonstration gives greater value to the grasp synthesis technique presented in this thesis.

Furthermore, this experiment revealed a number of interesting problems such as path planning in complex environments, grasp execution or the need for dexterous manipulation. Thus, this demonstrates that the Robotic Grasping Toolkit offers sufficient tools to study a multitude of problems that are relevant not only to grasping but robotics in general.

# CHAPTER 8: CONCLUSIONS AND FUTURE WORK

The work in this thesis has made two main contributions. The first one involved the presentation of a novel technique for effective grasp synthesis. For the second one, a versatile and efficient tool for studying robotic grasping has been introduced. So, the goal of this chapter is to summarize the key points and important problems that were addressed by each contribution. Furthermore, a brief discussion of the possible future work will be provided at the end.

## 8.1 Contributions

By relying on continuous collision detection (CCD) and the geometric notion of the wrench space, the grasp synthesis method was able to cope with several issues that are common in robotic grasping. First, a large reduction of the space of possible grasp configurations was achieved by sampling the work space of the fingers with an effective CCD search. Second, determining the grasp quality and Force-Closure for the remaining grasps, as done by other works, was not necessary since hyperplanes provided an efficient mechanism for selecting a grasp. Consequently, grasps can be synthesized by this technique in considerably less time than other methods that have been proposed for the same purpose. In addition, the grasp synthesis technique presented here is applicable to any hand model and object as it was demonstrated in the experiments of chapter 5. This capability has not been demonstrated by other grasp synthesis methods that have been proposed.

Another important issue that has been tackle by this work is that it integrated hand-based and contact-based methodologies very effectively. Thus, this allowed analyzing the grasp synthesis problem from different perspectives so as to formulate a solution that combined very innovative features contributed by each approach. Furthermore, the thought process that led to

the conception of the grasp synthesis technique can be used as a guideline for the development new methods that may address several grasping problems.

In regards to the Robotic Grasping Toolkit, several issues have been dealt with very effectively as well. The incorporation of scripting allows interacting with software in more superior ways than with graphical user interfaces. In particular, scripting made it possible to describe very detailed grasping tasks by means of flexible textual functions. Additionally, the object oriented design methodology made it possible to achieve a great level of modularity and flexibility. Also, this design approach accommodates very well to new features that extend the toolkit.

Moreover, the toolkit can store and process any type of hand model. This is a very welcome feature that can be used for exposing grasping methods to a number of challenges introduced by the inherent particularities of a hand model. Thus, several rigorous conditions can be generated so as to evaluate the feasibility of a particular grasping method very thoroughly.

Furthermore, the very robust physics simulation module permits evaluating the performance of hand models within virtual environments that respond to physical phenomena. Thus, this module can be used to study complex grasping problems such as grasp execution and dexterous manipulation, which are of dynamic nature.

## 8.2 Future Work

The grasp synthesis technique can resolve grasp very efficiently, but it could benefit from a number of improvements. These improvements include the autonomous determination of the palm location and automatic selection of the finger search order. The inclusion of these

capabilities would result in a greater level of autonomy and sophistication for the grasp synthesis technique.

Furthermore, the current state of the physics simulation only allows driving hand models kinematically. This means that the hand can evoke dynamic reactions from its surroundings but the reverse is not possible. Hence, allowing the hand to be dynamically driven would enable revealing how several dynamic factors affect its ability to perform a grasping task.

Moreover, several routines in the toolkit could welcome a number of optimizations so as to obtain greater computational performance. The use of parallel processing could offer adequate solutions in this regard. In particular, the burden of the CCD searches could be distributed across multiple processors and so a great reduction of the processing time would be achieved.

Lastly, the capabilities of the MatLab environment could be exploited even further by combining the Simulink control design tools with the physics simulation environment. Such feature would enable testing not only hand mechanism but also full robotic hand systems.

In conclusion, the work presented here has introduced a very effective grasp synthesis method that can be used for resolving very complicated grasping problems. Furthermore, this work has incorporated a number of conceptual grasp analysis tools within a flexible and modular computational toolkit and so it can be used for developing various types of elaborate grasping methods.

## BIBLIOGRAPHY


1 - [Salisbury, 1982] J. K. Salisbury. Kinematic and Force Analysis of Articulated Hands. PhD thesis, Department of Mechanical Engineering, Stanford University, 1982.

2 – Min Tang, Y. J. K. a. D. M. (2009). "C2A: Controlled Conservative Advancement for Continuous Collision Detection of Polygonal Models." IEEE International Conference on Robotics and Automation, May 12 - 17, Japan, 2009.

3 - [Hanafusa and Asada, 1982] H. Hanafusa and H. Asada. Stable prehension by a robot hand with elastic _ngers. In M. Brady, editor, Robot Motion Planning and Control, pages 323{336. MIT Press, 1982.

[4] S.C. Jacobsen, E.K. Iversen, D.F. Knutti, R.T. Johnson, K.B. Biggers. "Design of the Utah/MIT, Dextrous Hand." Center of Engineering Design, University of Utah. IEEE 1986.

[5] R. Platt, T.B. Martin, R.O. Ambrose, M.A. Diftler, M.J. Butzer. "Tactile gloves for Autonomous Grasping with the NASA/DARPA Robonaut." Proceedings of the 2004 IEEE International Conference on Robotics and Automation. New Orleans, LA , April 2004.

[6] Shadow Company. "Shadow Dexterous Hand C5 Technical Specificaton." August 10,2006

[7] ODE. Open dynamics engine. http://www.ode.org (visited 2010-02-14).

[8] Bullet. Bullet continuous collision detection and physics library. http://www.continuousphysics.com/Bullet/ (visited 2010-02-16).

[9] Newton Game Dynamics. Newton game dynamics. http://www.physicsengine.com (visited 2010-02-16).

[10] Nvidia Physx. http://developer.nvidia.com/object/physx.html( visited 2010-02-10)

[11] G. v. d. Bergen, "Collision Detection in Interactive 3D Environments," The Morgan Kaufmann Series in Interactive 3D Technology, 2003.

[12] OBBTree: A Hierarchical Structure for Rapid Interference Detection, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha, Proc. ACM SIGGRAPH, pp. 171-180, Aug. 1996. Also available as tech. rep. TR96-013, Dept. Comput. Sci., Univ. N. Carolina Chapel Hill, 1996.

[13] Goldstine, Herman H. (1972). *The Computer: from Pascal to von Neumann*. Princeton, New Jersey: Princeton University Press. ISBN 0-691-02367-0.

[14] "The ENIAC Story". Ftp.arl.mil. http://ftp.arl.mil/~mike/comphist/eniac-story.html. Retrieved 2008-09-22.

[15] Adrian Boeing, T. B. (2007). "Evaluation of real-time physics simulation systems."

[16] Axel Seugling, M. R. o. (2006). "Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool."

[17] J.K. Salisbury. "Robotic Hands and the Mechanics of Manipulation." Dept. of Computer Science , Stanford University., Stanford CA, May 1985

[18] Chen, I. M., Burdick, J. W.. A qualitative test for N-finger force closure grasps on planar objects with applications to manipulation and finger gaits, in Proc. of IEEE Int. Conf. Robotics and Automation, Atlanta, Los Alamitos: IEEE Computer Society Press, 1993, 814-820.

[19] Bicchi, A., On the closure properties of robotic grasping, Int. J. Robot. Res., 1995, 14(4):319-334.

[20] Ferrari, C. and J. Canny (1992). Planning optimal grasps. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, May 12, 1992 - May 14, 1992, Nice, Fr, Publ by IEEE.

[21] Ding, D., Liu, Y. H., Wag, S. G., Computation of 3-D form-closure grasps, IEEE Trans. Robot. Automat., 2003, 19(4): 669-679.

[22] Zhu, X. Y., Wang, J., Synthesis of force-closure grasps on 3-D objects based on the Q distance, IEEE Trans. Robot. Automat., 2003, 19(4):669-679.

[23] Buss, M., Hashimoto, H., Moore, J. B., Dexterous hand grasping force optimization, IEEE Trans. Robot. Automat., 1996, 12(3): 406-407.

[24] Buss, M., Faybusovich, L., Moore, J. B., Dikin-type algorithms for dexterous grasping force optimization, Int. J. Robot. Res., 1998, 17(8): 831-839.

[25] Zuo, B. R., Qian, W. H., A force-closure test for multi-fingered grasps, Science in China, Ser. E, 1998, 41(1): 62-67.

[26] E. Rimon and J. W. Burdick, "Mobility of bodies in contact—Part I: A second-order mobility index for multiple-finger grasps," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 696–708, Oct. 1998.

[27] Y. L. Xiong, "Theory of point contacts restraint and qualitative analysis of robot grasping," *Sci. China—Ser. A*, vol. 37, no. 5, pp. 629–640, 1994.

[28] V. D. Nguyen, "Constructing force-closure grasps," *Int. J. Robot. Res.*, vol. 7, no. 3, pp. 3–16, 1988.

[29] A. Bicchi, “On the closure properties of robotic grasping,” *Int. J. Robot. Res.*, vol. 14, no. 4, pp. 319–334, 1999.

[30] Y. Nakamura, K. Nagai, and T. Yoshikawa, “Dynamics and stability in coordination of multiple robotic mechanisms,” *Int. J. Robot. Res.*, vol. 8, no. 2, pp. 44–61, 1989.

[31] Matei Ciocarlie, C. L., Peter Allen (2007). "Soft Finger Model with Adaptive Contact Geometry for Grasping and Manipulation Tasks."

[32] Zheng, Y. and W. Qian (2005). "Linearizing the soft finger contact constraint with application to dynamic force distribution in multifingered grasping." Science in China, Series E: Technological Sciences **48**(2): 121-130.

[33] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Englewood Cliffs, NJ: Prentice-Hall, 1994.

[34] Y. H. Liu, “Qualitative test and force optimization of 3-D frictional form-closure grasps using linear programming,” *IEEE Trans. Robot. Automat.*, vol. 15, pp. 163–173, Jan. 1999.

[35] Yu Zheng. and C. Chee-Meng (2009). A numerical solution to the ray-shooting problem and its applications in robotic grasping. 2009 IEEE International Conference on Robotics and Automation (ICRA), 12-17 May 2009, Piscataway, NJ, USA, IEEE.

[36] N. S. Pollard. Parallel Methods for Synthesizing Whole-Hand Grasps from Generalized Prototypes. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1994.

[37] Ferrari, C. and J. Canny (1992). Planning optimal grasps. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, May 12, 1992 - May 14, 1992, Nice, Fr, Publ by IEEE.

[38] Richard M. Murray, Z. L., S. Shankar Sastry, (1994). "A Mathematical Introduction to Robotic Manipulation."

[39] Howard, W.S. and V. Kumar (1996). On the stability of grasped objects. IEEE Trans. Robotics and Automation 12(6), 904-917

[40] Buryninckx H., S. Demey and V. Kumar (1998). Generalized stability of compliant grasps. In: *Proc. IEEE ICRA 1998*. pp. 2396 – 2402.

[41] [Reuleaux, 1876] F. Reuleaux. The Kinematics of Machinery. Macmillan and Company, 1876. Republished by Dover in 1963.

[42] [Lakshminarayana, 1978] K. Lakshminarayana. Mechanics of form closure. Technical Report 78-DET-32, ASME, 1978.

[43] [Markensco_ et al., 1990] Xanthippi Markensco_, Luqun Ni, and Christos H. Papadimitriou. The geometry of grasping. International Journal of Robotics Research, 9(1):61{74, February 1990.

[44] [Mirtich and Canny, 1994] Brian Mirtich and John Canny. Easily computable optimum grasps in 2-D and 3-D. In Proc. of the 1994 IEEE International Conference on Robotics and Automation, pages 739{747, San Diego, CA, May 1994.

[45] Teichmann, M. and B. Mishra (1997), The power of friction: Quantifying the "goodness" of frictional grasps. In: Algorithms for Robotic Motion and Manipulation. A.K. Peters. Wellesley, MA, USA. Pp. 311-320

[46] Miller, A. and P. Allen (1999). Examples of 3D grasp quality computations. In: Proc. IEEE ICRA 1999. Pp. 1240-1246.

[47] Li, Z , and S. Sastry (1988). Task-oriented optimal grasping by multifingered robotic hands, IEEE J. Robotics and Automation 4(1), 32-44.

[48] Zhu, X., H. Ding and H. Li (2001). A quantitative measure for multifingered grasps. In: Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics 2001. pp. 213-219.

[49] Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls," ACM Trans. on Mathematical Software, 22(4):469-483, Dec 1996, http://www.qhull.org

[50] Ch. Borst, M. Fischer, G. Hirzinger, A fast and robust grasp planner for arbitrary 3D objects, in: Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI, 1999, pp. 1890_1896.

[51] X.Y. Zhu, J. Wang, Synthesis of force-closure grasps on 3-D objects based on the Q distance, IEEE Transactions on Robotics and Automation 19 (4) (2003) 669_679.

[52] Zheng, Y. and W.-H. Qian (2009). "Improving grasp quality evaluation." Robotics and Autonomous Systems **57**(6-7): 665-673.

[53] Roa, M., R. Suarez, et al. (2008). "Quality measures for object grasping." Revista Iberamericana de Automatica e Informatica Industrial **5**(Copyright 2008, The Institution of Engineering and Technology): 83-92.

[54] [Napier, 1956] J. Napier. The prehensile movements of the human hand. Journal of Bone and Joint Surgery, 38B(4):902{913, November 1956.

[55] M.R. Cutkosky and P.K. Wright, "Modeling Manufacturing Grips in Correlation with the Design of Robot Hands." Proc. IEEE Intl. Conf. on Robotics and Automation, San Francisco , CA, pp. 1533-1539, Apr. 1986

[56] [Iberall, 1997] Thea Iberall. Human prehension and dexterous robot hands. The International Journal of Robotics Research, 16(3):285{299, June 1997.

[57] [Lyons, 1985] D. Lyons. A simple set of grasps for a dextrous hand. In Proc. of the 1985 IEEE International Conference on Robotics and Automation, pages 588{593, 1985.

[58] [Stansfield, 1991] S. A. Stansfield. Robotic grasping of unknown objects: A knowledge-based approach. International Journal of Robotics Research, 10(4):314{326, August 1991.

[59] [Pao and Speeter, 1989] Lucy Pao and Thomas H. Speeter. Transformation of human hand positions for robotic hand control. In Proc. of the 1989 IEEE Inter-125 national Conference on Robotics and Automation, pages 1758{1763, Scottsdale, AZ, May 1989.

[60] Biggs, J. and K. Horch (1999). "A three-dimensional kinematic model of the human long finger and the muscles that actuate it." Medical Engineering and Physics **21**(9): 625-639.

[61] Valero-Cuevas, F. J., Y. Jae-Woong, et al. (2007). "The tendon network of the fingers performs anatomical computation at a macroscopic scale." IEEE Transactions on Biomedical Engineering **54**(Copyright 2007, The Institution of Engineering and Technology): 1161-1166.

[62] [Jameson, 1985] J. W. Jameson. Analytic Techniques for Automated Grasp. PhD thesis, Stanford University Department of Mechanical Engineering, 1985.

[63] D. Wren and R. Fisher, "Dexterous hand grasping strategies using preshapes and digit trajectories," in *IEEE Int. Conf. on Systems, Man and Cybernetics*, 1995.

[64] [Smith et al., 1999] G. Smith, E. Lee, K. Goldberg, K. Bohringer, and J. Craig. Computing parallel-jaw grips. In Proc. of the 1999 IEEE International Conference on Robotics and Automation, Detroit, MI, May 1999.

[65] R. Hester, M. Cetin, C. Kapoor, and D. Tesar, "A criteria-based approach to grasp synthesis," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1255–1260.

[66] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *IEEE International Conference on Robotics and Automation. ICRA '03.*, vol. 2, 2003, pp. 1824–1829.

[67] A. Miller and P. Allen, “Graspit! a versatile simulator for robotic grasping,” *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp.110–122, 2004.

[68] Xue, Z., J. M. Zoellner, et al. (2008). Grasp planning: find the contact points. 2007 IEEE International Conference on Robotics and Biomimetics, ROBIO, December 15, 2007 - December 18, 2007, Yalong Bay, Sanya, China, Inst. of Elec. and Elec. Eng. Computer Society.

[69] Xue, Z., J. Marius Zoellner, et al. (2008). Automatic optimal grasp planning based on found contact points. 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM 2008, August 2, 2008 - August 5, 2008, Xi'an, China, Institute of Electrical and Electronics Engineers Inc.

[70] [Markenscoff and Papadimitriou, 1989] Xanthippi Markensco_ and Christos H. Papadimitriou. Optimum grip of a polygon. International Journal of Robotics Research, 8(2):17{29, April 1989.

[71] [Chen and Burdick, 1992] I-Ming Chen and Joel W. Burdick. Finding antipodal point grasps on irregularly shaped objects. In Proc. of the 1992 IEEE International Conference on Robotics and Automation, pages 2278{2283, Nice, France,1992.

[72] [Ponce et al., 1993b] Jean Ponce, Steven Sullivan, Jean-Daniel Boissonnat, and Jean-Pierre Merlet. On characterizing and computing three- and four finger force closure grasps of polyhedral objects. In Proc. of the 1993 IEEE International Conference on Robotics and Automation, pages 821{827, Atlanta, Georgia, May 1993.

[73] Campbell RJ, Flynn PJ. A survey of free-form object representation and recognition techniques. Comput Vision Image Understanding 2001;81:166–210.

[74] Borst Ch, Fischer M, Hirzinger G. Grasping the dice by dicing the grasp. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, IROS; 2003. p. 3692–7.

[75] Wang MY. An optimum design for 3-D fixture synthesis in a point set domain. IEEE Trans Robotics Autom 2000;16(6):839–46.

[76] Liu YH, Lam ML, Ding D. A complete and efficient algorithm for searching 3-D form closure grasps in the discrete domain. IEEE Trans Robotics 2004; 20(5):805–16.

[77] Roa, M. A. and R. Suarez (2009). "Finding locally optimum force-closure grasps." Robotics and Computer-Integrated Manufacturing **25**(3): 536-544.

[78] Roa, M. A. and R. Suarez (2009). "Computation of independent contact regions for grasping 3-D objects." IEEE Transactions on Robotics **25**(Copyright 2009, The Institution of Engineering and Technology): 839-850.

[79] J. Ponce, S. Sullivan, A. Sudsang, J. Boissonat, and J. Merlet, “On computing four-finger equilibrium and force-closure grasps of polyhedral objects,” *Int. J. Robot. Res.*, vol. 16, no. 1, pp. 11–35, 1997.

[80] N. Pollard, “Closure and quality equivalence for efficient synthesis of grasps from examples,” *Int. J. Robot. Res.*, vol. 23, no. 6, pp. 595–614, 2004.

[81] Eric Larsen, S. G., Ming C. Lin, and Dinesh Manocha (1999). "Fast Proximity Queries with Swept Sphere Volumes,." Proc. IEEE Int.

[82] PQP (Proximity Query Package). http://gamma.cs.unc.edu/SSV/ (visited 2010-02-20)

[83] MIRTICH, B. V. 1996. Impulse-based Dynamic Simulation of Rigid Body Systems. PhD thesis, University of California, Berkeley.

[84] REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. Proc. Of Eurographics (Computer Graphics Forum).

[85] REDON, S., KIM, Y. J., LIN, M. C., AND MANOCHA, D. 2004. Interactive and continuous collision detection for avatars in virtual environments. In Proceedings of IEEE Virtual Reality.

[86] ORTEGA, M., REDON, S., AND COQUILLART, S. 2006. A six degree-of-freedom god-object method for haptic display of rigid bodies. In IEEE International Conference on Virtual Reality.

[87] MPK-TEAM. 2006. Motion Planning Kit. http://ai.stanford.edu/_mitul/mpk/.

[88] Zhang, X., S. Redon, et al. (2007). Continuous collision detection for articulated models using Taylor models and temporal culling. 34th Annual Meeting of the Association for Computing Machinery's Special Interest Group on Graphics, August 5, 2007 - August 9, 2007, San Diego, CA, United states, Association for Computing Machinery.

[89] S. Ramasamy and M.R. Arshad, “Robotic Hand Simulation With Kinematic and Dynamic Analysis” IEEE 2000.

[90] [Speeter, 1991] Thomas H. Speeter. Primitive based control of the Utah/MIT dextrous hand. In Proc. of the 1991 IEEE International Conference on Robotics and Automation, pages 866{877, Sacramento, CA, April 1991.

[91] L. Zollo, S. Roccella, E. Guglielmelli, M. C. Carrozza and P. Dario, “Biomechatronic Design and Control of an Anthropomorphic Artificial Hand for Prosthetic and Robotic Applications” IEEE/ASME Transactions on Mechatronics, Vol. 12, No. 4, August 2007.

[92] Yuru Zhang, Jiting Li, Jianfeng Li. “Robotic Dexterous Hand: Modeling, Planning, and Simulation” Mechanical Press, Beijing. 2007

[93] V. Nguyen, S. Oh, J. Lim,C. Kang and S. Han, " A Study on Design of Three - Finger Hand System" International Conference on Control, Automation and Systems 2008, Oct. 14-17, 2008 in COEX, Seoul, Korea.

[94] Howe, R. D., I. Kao, et al. (1988). The sliding of robot fingers under combined torsion and shear loading. Proceedings of the 1988 IEEE International Conference on Robotics and Automation (Cat. No.88CH2555-1), 24-29 April 1988, Washington, DC, USA, IEEE Comput. Soc. Press.

[95] MATLAB version 7.8.0. Natick, Massachusetts: The MathWorks Inc., 2009

[96] Craig, J. J. (1989). "Introduction to Robotics: Mechanics and Control." (Addison-Wesley Longman Publishing Co., Inc.): 450.

[97] El-Sawah, A., N. D. Georganas, et al. (2007). Finger inverse kinematics using error model analysis for gesture enabled navigation in virtual environments. 2006 IEEE International Workshop on Haptic Audio Visual Environments and Their Applications, HAVE 2006, November 4, 2006 - November 5, 2006, Ottawa, ON, Canada, Institute of Electrical and Electronics Engineers Inc.

# VITA