UNIVERSITY OF CALIFORNIA, MERCED

# Bridging the Gap in Grasp Quality Evaluation and Grasp Planning

by

Shuo Liu

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

Committee in charge:
Professor Stefano Carpin, Chair
Professor Marcelo Kallmann
Professor Ming-Hsuan Yang

Fall 2017

The dissertation of Shuo Liu is approved and it is acceptable in quality and form for publication on microfilm and electronically.

Stefano Carpin, Chair Date

Marcelo Kallmann Date

Ming-Hsuan Yang Date

University of California, Merced

To my parents and my wife

# Acknowledgments

I am greatly indebted my advisor, Dr. Stefano Carpin, for guiding and supporting me over these years. I was a careless person when I first entered the University of California, Merced, as I was more concerned with the speed at which I conducted my research instead of the quality of my work. He forced me to change the way I view my own research, which will carry me far in my career. Particularly, I want to thank him for letting me follow through on my ideas and helping me develop them into viable research topics. He set an excellent example as a researcher, mentor, and instructor. Although we had many disagreements, he always treated me with respect. It was a great honor to have him as my advisor. I would like to extend my gratitude to my committee members, Dr. Marcelo Kallmann and Dr. Ming-Hsuan Yang, for all of their guidance through this process; their feedback has been absolutely invaluable. I would like to thank my fellow graduate students, Dr. Seyedshams Feyzabadi, Jose Luis Rincon, Andres Torres Garcia, Thomas Thayer and many others, for their inspiring comments and valuable friendship. I've enjoyed every moment working with them.

I want to thank the National Institute of Standards and Technology for its partial support of my studies through cooperative agreement 70NANB12H143. I would like to thank University of California, Merced for providing me with Teaching Assistant opportunities in Fall 2014, Spring 2016 and Spring 2017. I am honored to receive Bobcat Fellowship from University of California, Merced in Spring 2014, Summer 2015, Fall 2016 and Summer 2017. I would also like to express my gratitude to Dorabot Inc. for offering me an internship in Summer 2016. This great opportunity had broadened my knowledge and brought me closer to the robotic industry.

I would especially like to thank my roommates, Dr. Youhong Zeng, Wei Liang, and Dr. Yanjun Su, for all the time we spent together, all the adventures we had, and much, much more. Without them, my life here would have been a horrible disaster. We clearly have built something permanent instead of just sharing a house. I would also like to thank Dr. Zhijiang Ye, Dr. Chengjie Qin, Dr. Jingru Shao, Dr. Jimei Yang, Dr. Zhe Hu, Dr. Tao Ren and many other friends that supported me throughout these years. Because of them, I was never alone when I was depressed or overwhelmed. It was a great honor to have them in my life. In particularly, I want to thank Dr. Zhijiang Ye and his family for taking care of me during the most depressing period of life. I also am grateful for all the fishing trips and poker nights we had together.

I want to express my deepest appreciation to my family, especially to my parents. My parents supported me through every bit of my life. They encouraged me to be an independent thinker, and have confidence in my ability to go after new things that inspired me. Although we are not always in touch, I know they will always be there for me and I hope they know that I will always be there for them. Finally, but not least, I want to sincerely thank my wife Huiying Wang. She is a very kind and caring woman and I am very grateful to have met her, gotten to know her and marry her. She is the reason that I keep pushing myself to become a better person. I could not imagine surviving all this without her loving support.

# Publications

## Conference Papers

- **S. Liu** and S. Carpin. Fast Grasp Quality Measure with Partial Convex Hull Computation. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pages 4279-4285, 2015

- **S. Liu** and S. Carpin. Global Grasp Planning Using Triangular Meshes. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pages 4904-4910, 2015

- **S. Liu** and S. Carpin. A Fast Algorithm for Grasp Quality Evaluation Using the Object Wrench Space. In *Proceedings of the IEEE Conference on Automation Science and Engineering,* pages 558-563, 2015

- **S. Liu** and S. Carpin. Kinematic Noise Propagation and Grasp Quality Evaluation. In *Proceedings of the IEEE Conference on Automation Science and Engineering,* pages 1177-1183, 2016

- **S. Liu**, Z. Hu, H. Zhang, M. Kwon, Z. Wang, Y. Xu and S. Carpin. Grasp Quality Evaluation and Planning for Objects with Negative Curvature. In *Proceedings of the IEEE International Conference on Robotics and Automation,* pages 2223-2229, 2017

- **S. Liu** and S. Carpin. Grasp Quality Evaluation with Whole Arm Kinematic Noise Propagation. In *Proceedings of the IEEE International Conference on Robotics and Automation,* 2018 (under review)

## Journal Papers

- J. Falco, K. Van Wyk, **S. Liu** and S. Carpin. Robotic Grasping: Facilitating Replicable Performance Measures via Benchmarking and Standardized Methodologies. In *IEEE Robotics and Automation Magazine,* 22(4):125-136, 2015

- **S. Liu** and S. Carpin. Partial convex hull algorithms for efficient grasp quality evaluation. *Robotics and Autonomous Systems* 86:5769, 2016

## Technical Report

- S. Carpin, **S. Liu**, J. Falco, K. Van Wyk. Multi-fingered Robotic Grasping: A Primer. *arXiv preprint arXiv:1607:06220,* 2016

# Contents


**1 Introduction** **2**

**2 Related Work** **5**
2.1 Grasp planner . . . 5
2.1.1 Force Closure Grasp . . . 5
2.1.2 Grasp Quality Driven Model-Based Grasp Planner . . . 8
2.1.3 Sensor Data Driven Model-less Grasp Planner . . . 9
2.2 Grasp Quality Metrics . . . 11
2.2.1 External Wrench Resistance Based Measure . . . 12
2.2.2 Hand Based Measure . . . 14
2.2.3 Combing Multiple Measures . . . 15
2.3 Convex Hull Computation . . . 16
2.3.1 Convex Hull in 2D and 3D . . . 16
2.3.2 Convex Hull in Higher Dimensions . . . 17
2.4 Inverse Kinematics . . . 18

**3 Background and Notation** **19**
3.1 Background in grasping and grasp metrics . . . 19
3.1.1 Grasp wrench space metric . . . 20
3.1.2 Object wrench space metric . . . 21
3.2 Background in Convex Hulls . . . 22
3.2.1 Convex Hulls . . . 22
3.2.2 The QuickHull algorithm . . . 23

**4 Efficient Grasp Quality Evaluation through Partial QuickHull Computation** **25**
4.1 Grasp Wrench Space Metric with Partial Convex Hulls . . . 26
4.2 Object Wrench Space Metric with Partial Convex Hulls . . . 31
4.3 Experimental Evaluation . . . 37
4.3.1 Grasp Wrench Space Metric . . . 37
4.3.2 Object Wrench Space Metric . . . 38
4.3.3 Impact on Planning . . . 41
4.4 Conclusions . . . 44

**5 Grasp Quality Evaluation with Whole Arm Kinematic Noise Propagation 46**
5.1 Problem Definition and Methodology . . . 48
5.2 Experiments . . . 52
5.2.1 Empirical estimation of noise on end-effector position . . . 52
5.2.2 Grasp Quality and Noise . . . 53
5.2.3 Analytical based estimation of noise on end-effector pose . . . 59
5.3 Conclusions and future work . . . 65

**6 Grasp Quality Metric Improvement Considering Hand Configuration and Target Object 67**
6.1 Grasp Quality Metric Improvement . . . 69
6.1.1 Grasp Quality Metric with Hand Configuration . . . 69
6.1.2 Grasp Quality Metrics with Negative Curvature . . . 71
6.1.3 Combining the Hand Friction Cone and Object Friction Cone . . . 74
6.2 Experiments and Results . . . 74
6.2.1 Preliminary Results Considering Hand Configuration . . . 74
6.2.2 Preliminary Results Considering The Target Object . . . 75
6.2.3 Results on Combined Friction Cone . . . 76
6.2.4 Impact on Grasp Planning . . . 77
6.3 Conclusions . . . 78

**7 Grasp Planner Development 81**
7.1 Grasp Planner using Negative Curvature . . . 81
7.1.1 Model Based Grasp Planning . . . 82
7.1.2 Model-less Grasp Planner . . . 84
7.1.3 Negative Curvature Planner Comparison . . . 90
7.1.4 Experiments and Results . . . 91
7.1.5 Real Robot Experiment . . . 92
7.2 Global Grasp Planning Using Triangular Meshes . . . 95
7.2.1 Continuity . . . 96
7.2.2 Proposed Approach . . . 96
7.2.3 Experimental Results . . . 103
7.3 Conclusion . . . 108

**8 Conclusions and Future Works 110**

**Bibliography 113**

# Abstract

Robot grasp planning has been extensively studied in the last decades often consisting of two different stages determining where to grasp an object and measuring the quality of a tentative grasp. Additionally, because these two processes are computationally demanding, form closure grasps are more widely used in practice than force closure grasps, even though the latter is, in many cases, preferable. In this dissertation, we introduce our framework to improve grasp quality evaluation by increasing the speed of evaluating a grasp and developing more informative metrics. Specifically, we accelerate the computation of the grasp wrench space, used to measure the grasp quality, by exploiting some geometric insights in the computation of a convex hull through identifying a cutoff sequence to terminate the convex hull calculation with guaranteed convergence to the quality measure. Furthermore, we go into detail about the metric improvement for the grasp quality. Specifically, we study how noise at each joint of the manipulator affects grasp quality and how different arm configurations will generate different noise distributions at the end-effector, which has a huge impact in the robustness of grasping. Moreover, we illustrate our method that evaluates arm configurations based on the probability of achieving a force closure grasp. Then we introduce our work taking into account the hand structure and the local geometry of the object to be grasped as the second aspect for improving grasp quality metrics. In particular, for concave objects, we exploit the fact that grasping the concave region can make the grasp more robust. These insights are explored through theory and then validated on an experimental platform. Finally, we present three grasp planners we developed. We constructed two planners taking advantage of the negative curvature feature. The first the planner uses the geometry model of the object and constructs a database for online use. The second planner does not require the model but instead, detects negative curvature features on the fly and calculates candidate grasps in real time. Lastly, our third grasp planner searches through the objects' surface, represented as a triangular mesh, and tries to find the global optimal grasp.

# Chapter 1

# Introduction

Grasping is one of the most widely studied problems in robot science. However performing reliable grasps on objects used in everyday activities continues to be one of the most important unsolved problems. This limitation is well described by Moravec's Paradox: *"it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility"* [69]. Considering all strategies a robot needs to perform in assisting and interacting with human beings in everyday life, grasping is -in the long term vision - still a missing block. It has been argued that grasping, manipulation, and speech are among the most fundamental human abilities unparalleled by animals [10]. Then it should not be a surprise that many problems in this area are still open.

As a starting point, we can not ignore the most commonly used strategy in real world grasps, form closure grasps. Form closure grasp aims to prevent the object from moving in any direction. Currently, form closure grasps are still vastly used in both academia and industry. Force closure grasps refers to grasps that can resist any motion of the object by the contact forces. The primary difference between form closure and force closure grasps is the latter relies on contact friction which translates into requiring fewer contacts and is suitable for executing precision grasps. In the early years, the limitations of the manufacturing robot hand and the expenses required to calculate good force closure grasp put constraints on producing complicated grasps. Nowadays, with the development of mutifingered robot hands, and increased computing speeds, preforming good force closure grasp is feasible. One might ask what a good force closure grasp is and how it is defined. This question can be answered in many ways since there exists many different quality measures. But in general people agree on a quality measure which determines the robustness of a grasp that can resist arbitrary wrenches while applying minimal forces with the finger. In this dissertation, the grasp quality measure will be based on force closure grasps.

The general purpose of grasp planning problems is to determine how to grasp an object in order to finish a given task. Assume that a robot is programmed to grasp an object on a table at point A and move it to point B with some obstacle on its path. An example is shown in Figure 1.1. One might focus on where to grasp the object, and others might be more interested in how to plan a safe path. As a matter of fact, both problems are equally important and in some ways related to grasp planning. Due to the errors of the actuator, we can not count on it to preform exactly the

motion we programmed; thus the path around obstacles will have effects on the grasp. Also, if implementing grasping in the real world, it is obvious to see the how failure of grasping process can be caused by the error of the actuator. This kind of error cannot be corrected completely with the current development of hardware and control system. In order to fix this, we should apply certain knowledge in the grasp planner, and improve the robustness of the grasp.

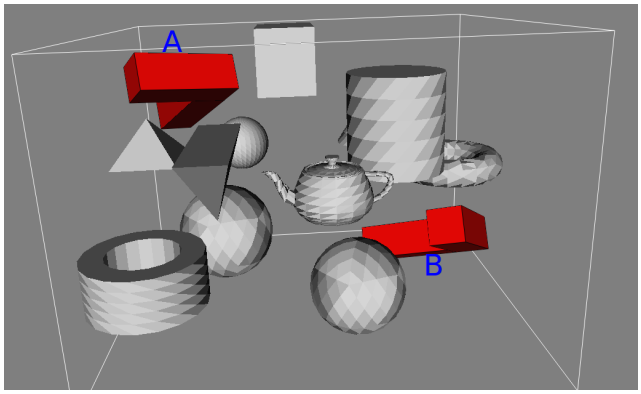



Figure 1.1: Example of a working environment. The manipulator will be planning to move the red object from A to B. The figure was generated using OMPL [98]

In this dissertation, we posit that the key to solving grasp planning problems is the grasp quality metric. Most well know grasp quality metrics are constructed based on a single feature, such as hand configuration and position of each contact. However, we think that the optimum grasp quality metric should be able to combine all features that have an effect on grasping, including hand force, contact position, local geometry of the target object, and the configuration of the entire arm. A force closure grasp requires two important aspects when evaluating its quality metric. The first one is the construction of grasp wrench space. In short, the grasp wrench space represents a space of wrenches that can be resisted by a given grasp configuration. The concept will be further introduced in chapter 3. The second aspect is the evaluation of the grasp wrench space. For example, Ferrari and Canny proposed a metric to evaluate the grasp wrench space by the minimal wrench in any direction [29]. Our focus is mainly on the construction part. In general, the construction of the grasp wrench space is based on the friction cone at each contact point defined by the grasp configuration. However, we believe that by modifying the friction cone, we will be able to add in more information such as the local geometry of the object being grasped and the hand configuration. Thus, we can build a comprehensive measurement system that combines multiple features into a single grasp quality metric.

In this thesis, our focus is listed as follows:

- Improve the calculation speed for two important quality metrics, i.e., the grasp wrench space metric and the object wrench space metric.
- Connect grasp quality measure with arm configuration.
- Connect grasp quality measure with local geometry information.

- Connect grasp quality measure with hand configuration.
- Develop a framework to combine items 1 to 4 into a single grasp quality metric.
- Develop a global grasp planner that generates high quality force closure grasps using triangular meshes as object representations.
- Develop a model based grasp planner and a model-less grasp planner based on local geometry information.

The main research focus of this dissertation is on robot grasping. Topics 1 through 5 contributes to improving grasp quality measure in calculation speed, i.e., topic 1, and in metric, i.e., topics 2 through 4. Topics 2 through 4 improve the quality metric by adding in information about the object to be grasped, the hand configuration and the manipulator configuration. These three aspects together form the entire system of grasping with a manipulator. The last two topics, built upon the improved quality measure, connect the quality and actual grasping strategies through grasp planning for both model and model-less scenario.

The rest of this dissertation is organized as follows. Chapter 2 will discuss some related work in grasping. In chapter 3 we introduce background knowledge and basic notations. Chapter 4 will be introducing our previous work which focuses on improving the calculation speed of the grasp quality measure published in [60]. Chapter 5 will be introducing our work that combines grasp quality metrics with arm configurations by taking grasp probabilities into account. This chapter is based on work published in [59]. In the next chapter, chapter 6, we address our work that connects hand and object local geometry with grasp quality measure using the friction cone. Part of this chapter was published in [61]. Chapter 7 will be introducing the grasp planners we developed over the years based on published papers [61] and [58]. The last chapter will conclude this dissertation.

# Chapter 2

# Related Work

In this chapter, we will provide a recap of relevant literature in grasp planners, grasp quality metrics, and convex hulls. Throughout this chapter, we exclusively consider force closure grasps.

## 2.1 Grasp planner

Because of its practical importance, grasp planning has received significant attention since the very dawn of robotics research. For a general introduction on grasping, the reader is referred to [70, 79]. Literature in grasp planning is vast and different taxonomies could be considered to classify the various approaches proposed.

### 2.1.1 Force Closure Grasp

When discussing the literature of grasp planning, we are often referred to grasp planning based on force closure grasps. A force-closure grasp is a type of grasp that can exert, through a set of contacts, arbitrary force and moment on this object [73]. In other words, a force closure grasp can generate any external wrench that the grasped object may have resisted and counteract any external disturbing wrenchs. Before going into grasp quality measures for force closure grasp, researchers have identified a multitude of properties that a good force-closure grasps must consider in comparison to a human hand in order to perform everyday tasks [22, 39].

- Dexterity
- Equilibrium
- Stability
- Dynamic behavior

Each of these properties contains different sub-problems. For example, *dexterity* studies how should each finger be configured; *equilibrium* studies how much force should be applied at each contact; *Stability* studies how to balance external disturbances and *dynamic behavior* studies how a grasp

should be configured for a given task. Figure 2.1 shows the interrelationship between these four properties and other well known grasp properties proposed in the grasping literature [92].

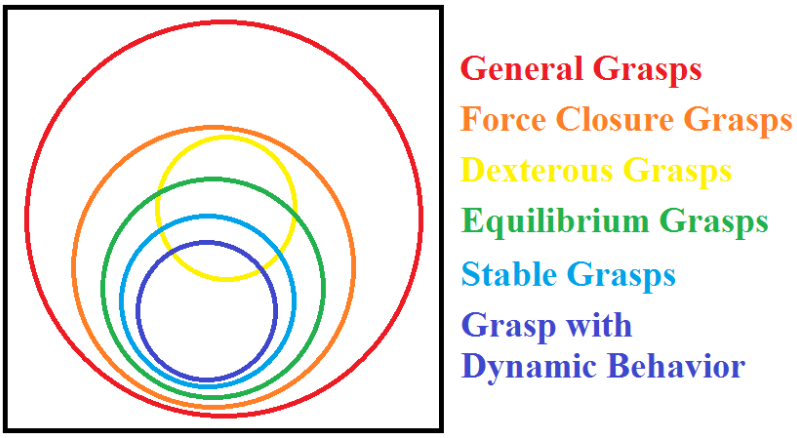



Figure 2.1: The interrelation between multiple grasp related properties proposed in the literature [92].

This paragraph will introduce of some works based on these properties. Beginning around the early 1980s, researchers started to investigate force closure problems. Laugier proposed a framework to determine the potential sets of grasping object positions based on rejection of infeasible grasps [52, 53]. His work is done on two finger hands grasping polyhedral objects in 3D. Wolter et al. developed an efficient method for a two finger hand to automatically generate grasp positions for polygonal objects with parallel planes in 2D [109]. Barber improved Wolter et al.'s work by considering more general conditions [6, 7]. Similarly, Abel et al. also developed an algorithm for two finger hands grasping polygonal objects in 2D [1]. In addition, they investigated the effect of surface roughness and the feasibility of achieving equilibrium grasps. Later on, Nguyen proposed an algorithm that computes stable grasps with frictional contacts on polygonal objects based on *independent contact regions* [73]. *Independent contact regions* are defined as a set of regions on the object where each finger can be placed independently and keep its force closure property within these regions. Examples for computing *independent contact regions* were given in [50, 81]. Nguyen also proved in that paper that all 3D force closure grasps can be made stable. Such paper is considered one of the fundamental works of grasp planning. Following this paper, works in calculating grasps for two-fingers, three-fingers and four-fingers on a polygonal object were published in [103], [74] and [64], respectively. In [77], the authors compute four-finger equilibrium force closure grasps on polyhedral objects. The authors presented a geometric characterization of all possible types of four-finger equilibrium grasps and solved the problem using linear optimization.

One of the earliest papers that planned grasps considering the palm was published in [101]. Palm in this case stands for the flat surface where all the fingers are placed on. This work also proposed the idea of an enveloping grasp, where multiple contacts are allowed between chains and the object. Fingertip force closure grasps are ideal, but for more realistic cases, the fingers often have more than one contact. The authors also defined *tippablilty regions* to help achieve enveloping

grasps quickly. A *tippablilty region* is defined as a region to place a contact where increasing contact forces will drive at least one of the supporting contact forces to zero. Under-actuated robotic hands are defined as hands which have more degrees of freedom than actuators, which can be self-adaptive, robust, and easy to control. Readers are referred to [11] for more in depth information. The grasp characteristics needed for under-actuated hands to perform enveloping grasps are studied in [62]. Furthermore, [47] proposed a quality measure for under-actuated hands regarding their ability to grasp and hold while performing enveloping grasps.

Most force closure grasp studies are based on polygonal or polyhedral objects, however, some researchers are interested in more complicated objects, i.e., objects with curved edges. An approach to compute stable grasps of curved two-dimensional objects is presented in [28], where fingers can be positioned independently by optimization within the grasp regions. In [54], the authors computed three-fingers grasps on irregular 2D and 3D objects based on geometrical analysis, and developed a simple algorithm which only needs a few algebraic calculations. They also proposed new necessary and sufficient conditions for 2D and 3D equilibrium and force closure grasps and a corresponding algorithm for computing force closure grasps.

The works introduced are solve problems related to dexterity, equilibrium and stability properties. These properties are all directly related to the structure of the hand or the shape of the object, i.e., the two most important components to perform a grasp. The dynamic behavior property, on the other hand, is task oriented. It is often set aside from other properties due to the fact that modelling a task and providing objectives to compare different grasps based on the task is difficult. In order to take task information into account, Chiu [19] proposed a task compatibility index to measure the similarity between the optimal direction of the manipulator and the actual moving direction based on the task. This work is done on the manipulator scale, i.e., unrelated to grasps, but the idea can be transferred to deal with task based grasping problems. The actual moving direction can be considered a source of disturbance which activates grasp planning approaches based on disturbance force rejection. To this matter, Li and Sastry proposed the *task ellipsoid* in [55]. The *task ellipsoid* addresses the likelihood of collision in each direction. This ellipsoid is further generated in six-dimensions, as in wrench space, which was later referred to as task wrench space (TWS). Discussions on how to evaluate grasp quality based on TWS will be introduced in section 2.2. However, the process of modelling a task using a task ellipsoid is very complicated and currently not used in practice. This fact made the task ellipsoid model very hard to scale up.

Instead of finding a grasp and evaluating its quality based on task information, a different approach that takes the task into account using hand-preshapes at early grasp planning stages was proposed in [78] by Prats. The authors defined different hand-preshapes and classified them based on their grasp wrench space. When a task is given, the robot will select the most suitable hand-preshape according to its grasp wrench space and automatically plan a set of actions for grasping the object and performing the task. The authors then used the concept of a *task frame*, introduced in [65], to fill the gap between the grasp and the task. A *task frame* is a concept from task planning, which implies a geometric coordinate frame that is attached to the object being manipulated. *Task frame* is adapted for tasks occurring along a specific direction, such as opening a door or a drawer. Prats' method is thereby limited to easy tasks and fails to associate proper hand-preshapes with complicated tasks.

### 2.1.2 Grasp Quality Driven Model-Based Grasp Planner

Grasp quality measures are often treated as objective functions to guide the grasp planning system in the searching process. A literature review of existing grasp quality measures is given in section 2.2. Optimal output of such system usually represents local optimum grasps which are considered to be stable by the grasp quality measure the author chose. In [110], the author introduced a highly integrated grasp planning system, that starts with an initial grasp and tries to improve the grasp quality by moving the fingers to its neighbouring joint positions until a local maximum grasp quality is reached. This way of using grasp quality metric is also addressed in [81], [23] and [116]. This kind of planner is usually not designed for a particular type of hand and can have an arbitrary number of contacts. In order to work with a real robot, simulation softwares are often used. In [67], a robotic grasping simulator, called Graspit!, which focus on the grasp analysis of force closure grasps is proposed. GraspIt! embeds the collision detection and contact determination system allowing a user to evaluate a grasp and compute optimal grasping configurations. Openrave [26] and V-REP [83] are also two commonly used simulators for grasp planning. By using simulation, we can construct an offline database for a specific hand and a collection of objects we aim to grasp. This database is usually based on a sampling approach where the quality of resultant grasps satisfies a fixed threshold. This approach is well addressed in [66]. At runtime, model-based grasp planners often requires human guided input. An example of a common pipeline for this kind of grasp planner being applied on a real robot is shown in Figure 2.2. These methods rely on collections of formerly computed grasps for *typical* objects. The Columbia Grasp Database is probably the most well known datasets supporting these methods [32]. However, a more autonomous way to handle a grasp database based on model at runtime is to combine it with a model-less grasp planner to recognize the object and estimates the object's pose. This will be discussed more in the next subsection.

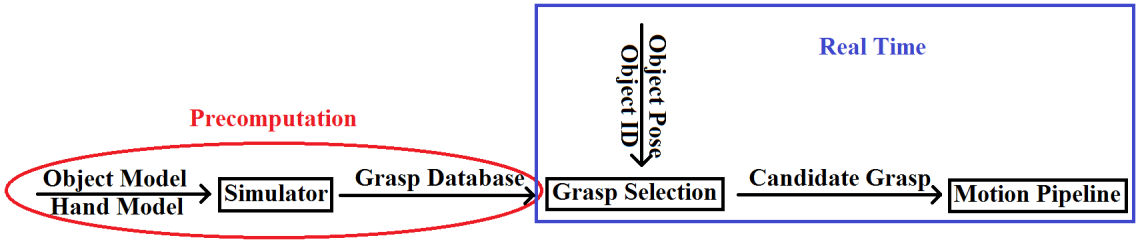



Figure 2.2: The pipeline diagram for grasp quality driven model-based grasp planner applied on real robots.

Most grasp planners are focused on the search process to generate grasp candidates, however, some works are focused on the object's geometric representation while doing grasp planning. Some commonly used geometric representations are shown in Figure 2.3. The availability of geometric models based on well defined primitives like superquadrics, or those that can be analytically described will favor the optimization process due to its continuity. In [42], the authors assume that the object to be grasped is modeled by a superquadric, and, notably, include in the grasp planning stage a set of constraints imposing that the planned grasp can actually be performed by a given robotic

hand. Superquadric representation can give a close approximation to objects which cannot be analytically described by one continuous function. This globally continuous function is beneficial to perform the optimization process for grasp planning and enables the grasp planner to explore the entire surface of the object. However, there is a trade off between surface approximation and function complexity. Superquadric representation also lacks the ability to generate a very close approximation for complicated objects. In [116], the authors instead assume that the surface of the object is represented by a patch of parametric surfaces like planes, spheres, etc. This kind of representation may work well for objects which can be simply parametrized. But for complicated objects such as teapots or drills, it requires a huge amount of constrained functions which are hard to obtain by hand and hard to perform grasp planning on. Another common limitation of these two approaches that we try to overcome, is that a valid initial placement of the fingers needs to be given. Moreover, during the grasp planning process, the finger position is constrained to remain on the surface where it started. Therefore, in the case of a complex object whose shape is given by the union of many surfaces, the planning process does not explore the whole search space, but only a subset. A method based on triangular meshes was published in [35]. This method is based on a hierarchy of triangular meshes at different resolutions. This method does not rely on initial setup, but it might narrow the result to few local optimum grasps which are not enough to build a comprehensive database, i.e., a database containing grasps from all directions. Furthermore, this method does not have the ability to explore the entire surface. The method we developed based on triangular meshes is capable of travelling across adjacent triangles. This technique enables the planner to explore the object surface globally and output a large number of local optimum grasps for constructing a grasp database. This method will be discussed in section 7.2.

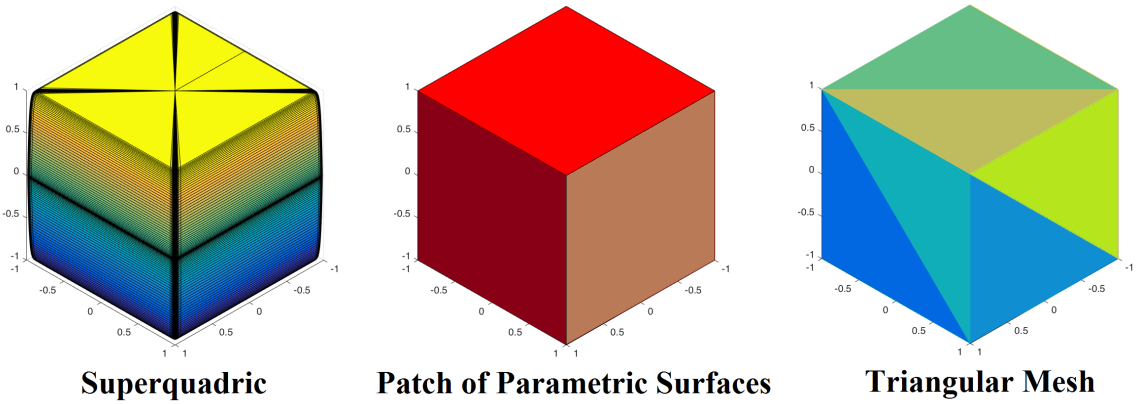



Figure 2.3: Three commonly used geometric models to represent a cube.

### 2.1.3 Sensor Data Driven Model-less Grasp Planner

In numerous situations, people are interested in grasping unknown objects, namely model-less grasps. Model-less grasp planning usually refers to grasp planning based on real capture sensorial data, rather than a perfect geometric model given up front. A common way of solving such

problems, as in [13, 93, 99, 102], is to construct the target object from the captured data. In order to be able to reconstruct the entire object, the data capturing process can become time consuming. Then a grasp planning algorithm is used to analyze the model and generates a robust force closure grasp. This kind of method can be considered the first step towards a model-less grasp planning. Although they do not have a perfect model up front, the grasps are still planned based on a built model. As an extension, the idea of data-driven grasping is proposed. Data-driven grasping is based on the existence of a large indexed dataset of object data, so that new inputs can be quickly matched to similar instances and obtain precomputed grasps. This idea was first introduced in [31], where they also proposed a framework for a data driven grasp planner that indexes partial sensor data into a database of 3D models along with precomputed grasps and transfers grasps from models in the database to target objects. Due to construction and querying reasons, it is impossible to store every possible object in the database. In fact, for commonly used objects, although they are not stored in the database, the planner is still capable to of grasping it by matching it with similar objects. However, since only partial sensor data is used, mismatching or wrong angle matching often occurs. Using deep learning, [104] developed a system to capture a single view depth image and reconstruct it in 3D object. Then, a grasp planner is used to plan and execute based on this single view image. This work improves the problem of online time consuming object reconstruction in the works mentioned earlier. It saves time in capturing data and performing grasp planning based on the target object. The only downside of this work is that the accuracy of this deep learning model is not guaranteed. Figure 2.4 shows an example of the pipeline diagram for sensor data driven model-less grasp planner applied on real robots. Up to now, these works are still connected with the model of the object. However, researchers have come up with a very efficient model-less grasp planners that are only based on features captured by the vision system, instead of learning what the object is. In [88], the authors treated image features as a representation of good grasping points, used synthetic images as training data and applied supervised learning. The outcome was then used to recognize *good grasping points* and perform grasps in real time. However, the feature vector characterizing a *good grasping point* is in very high dimension. Although [4] improved this algorithm using dimensionality reduction, the problem is still complicated. Apart from this, the authors of [30] proposed a system to compute stable grasps based on Height Accumulated Features (HAF) and their extension, Symmetry Height Accumulated Features, extracted from the point cloud directly. HAF is a simple topographic feature which is common among target objects and can be simply computed. The result indicates very robust performance. Furthermore, in [16], the authors proposed a novel grasping algorithm that uses active vision and curvature information obtained from the silhouette of the object to guide the grasp. Comparing to HAF, curvature features are rather hard to find among novel objects. However, these two methods lack theoretical support. It is not guaranteed that the candidate grasps they selected is the one with the highest quality. In fact, if they can quantify the quality of their candidate grasps and rank them in order for execution, they might achieve a higher success rate. In conclusion, these methods can be improved by formulating the effect of their features as grasp quality metrics.

Compared to model-less grasp planners, model-based grasp planners require additional precomputation to build the grasp database. This step is very time consuming because a large amount of grasps needs to be calculated for every object. Also, the grasping ability for this type of planner is only limited to objects which are similar to the ones stored in the database. On the other hand,

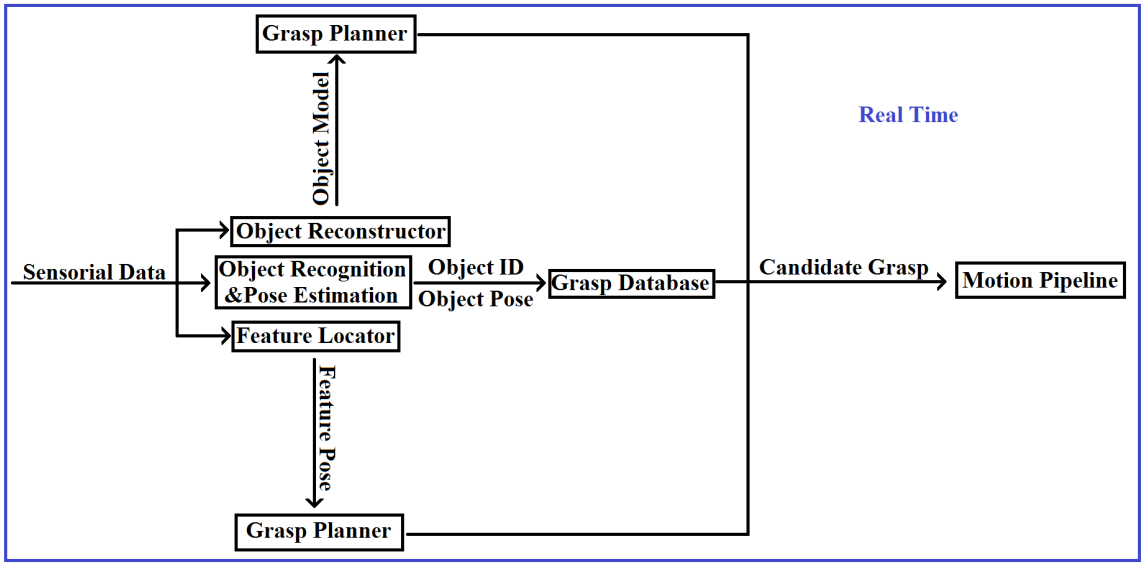



Figure 2.4: The pipeline diagram for sensor data driven model-less grasp planner applied on real robots.

because of this precomputation step, when this type of planner runs on real robots, it is more efficient and accurate for grasping objects in its database. This type of planner performs very well in a manufacturing environment. For model-less grasp planners, methods that require reconstructing the target object from multiple view points are the most time consuming. The positive side of this type of planner is that it works on any object. Model-less grasp planners based on feature extraction are considered to be the most efficient. This type of planners do not require precomputation, and when performed in real time, they only need to extract graspable features instead of recognizing the object. The negative side of this type of planners are that when the sensor data does not contain any graspable feature, they will fail. Although they have the ability to grasp unknown objects, they are only limited to the set of objects that contains recognizable graspable features. However, for an advanced grasping system, these three types of planner can be combined in serial. The first planner in the pipeline can be the planner based on features. If no graspable feature can be identified, we can shift to the planner using the database to check if the target object is similar to what is stored. If this planner fails, it means that the target object is unknown with no graspable features. Then we need to put it through the planner that reconstructs the object and plans the grasp. A planner pipeline example combining these three different type of planners is shown in Figure 2.5.

## 2.2 Grasp Quality Metrics

Everyday experience suggests that objects can be restrained in different ways and it is therefore natural to ask which grasping configuration is preferred. Being able to quickly evaluate the effectiveness of a metric impacts grasp planning efficiency, because planning can be seen as a search in the space of possible grasps driven by the quality metric [81, 86]. With these motivations, various

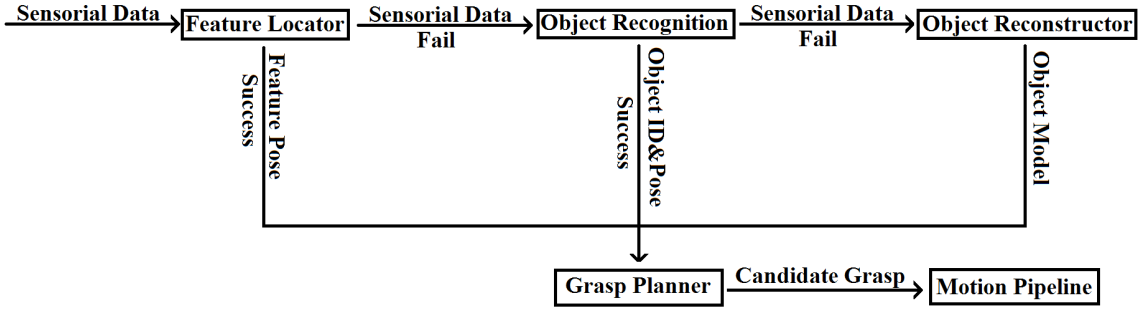



Figure 2.5: The pipeline diagram to combine three grasp planners.

grasp quality metrics have been proposed and the reader is referred to [82] for a recent survey.

### 2.2.1 External Wrench Resistance Based Measure

A force closure grasp is a grasp that can resist arbitrary wrenches on the object. In other words, any disturbance wrench on the object is resisted by the contact forces. Given two force closure grasps, one would intuitively prefer the grasp that can balance disturbances with the minimal effort, i.e., applying the smallest forces. This idea was originally introduced by Kirkpatrick et al. [44] and later on refined by Ferrari and Canny [29]. This metric proposed by Ferrari and Canny is obtained computing the convex hull of a grasp wrench space (GWS), i.e., six-dimensional vector space for 3D. The concept of grasp wrench space will be introduced in detail in section 3.1. This grasp quality measure has de-facto become the most frequently used grasp metric, and the QuickHull algorithm [5] is commonly used to compute the corresponding convex hull. Despite its popularity, this quality measure has some notable drawbacks. To be specific, it is not scale invariant, it does not consider the geometry of the object being grasped, and its value depends on an arbitrary choice for the point about which torques are computed. Moreover, it is too conservative, in the sense that it considers the set of all possible disturbance wrenches instead of considering the set of disturbance wrenches that can occur in practice. Finally, its computation relies on an approximation of the friction cone, and in order to expedite the computation there is then an incentive to use coarse approximations. Despite these limitations, however, it continues to be the most commonly used. In an effort to overcome some of these drawbacks, alternative approaches have been introduced. Zheng proposed to replace QuickHull with an algorithm that iteratively approximates the convex hull by growing a polytope guaranteed to be inside the convex hull [114]. Different from partial QuickHull for grasp wrench space (PQHGWS), which we published in [60], this algorithm cannot be applied when evaluating grasps that do not achieve force closure, and this is a limitation because when a grasp is not force closure the distance between the hull and the origin is still useful to guide the planning process [115]. The algorithm presented in [114] provides used the exact friction cone instead of a pyramid approximation. However, this algorithm is still calculating an approximation of the convex hull and the approximation error converges to zero only asymptotically with an unknown convergence rate. As for PQHGWS, approximated friction cone was used but the convex hull is then determined. Between [114] and PQHGWS, it is hard to argue which method is more

efficient and which method is more accurate. No publicly available implementation is available for [114], so no experimental comparisons can be made. Pokorny and Kragic [75] provided the first accurate study of the analytical properties of the grasp wrench space metric proposed by Ferrari and Canny and of its approximations. Starting from this study, they proposed an efficient algorithm to recognize and reject non-force closure grasps, but this method does not rank grasps achieving force closure, i.e., it does not compute a grasp quality metric. Importantly, [75] presents an analytical bound for the approximation error introduced by the discretization of the friction cone, and we exploit their result to quantify the quality of our findings. Various methods have been proposed with the objective of removing the approximation introduced by discretizing the friction cone [57, 115]. None of these has however gained much traction, mostly because of the associated computational costs. In an effort to consider not all disturbance wrenches, but only those occurring in practice, the object wrench space (OWS) was introduced [76]. Strandberg and Wahlberg [96] introduce a method for the direct computation of the OWS metric. The key observation is that a disturbance wrench is almost invariably generated by a disturbance force, and they formulate a metric considering how much the object wrench space can be inflated before it reaches the boundary of the grasp wrench space. The grasp wrench space metric can be thought as inflating a unit sphere within grasp wrench space until the sphere hits the boundary, and the quality is determined as the ratio between the inflated sphere and the unit sphere. It is clear that by defining the grasp wrench space this way, object wrench space metric is simply replacing the unit sphere with the object wrench space. The object wrench space metric overcomes two major problems of grasp wrench space metric, i.e., it is scale invariant and its value does not depend on an arbitrary choice for the point about which torques are computed. However, the object wrench space metric has limited use because it is costly to compute and the method provided by [96] is only slightly better than a brute force approach. Borst et al. [14] proposed a method to approximate the object wrench space by computing an enclosing ellipsoid rather than its exact shape. This ellipsoid is then transformed into a sphere using a linear transformation and the GWS metric is used to assess the quality of the grasp. This method is computationally compared to grasp wrench space metric. However, it is approximate and there can be a significant mismatch between the actual object wrench space and the enclosing ellipsoid. Both methods in [96] and [14] also face a problem, where the grasp wrench space computation must also include the origin. In other words, their methods can only be used on force closure grasps. We improved the calculation speed of this metric and made it comparable to the calculation of the GWS metric and also give a solution for non-force closure grasp in [60]. A similar method was proposed in [41]. Li and Sastry defined the concept of task wrench space (TWS), i.e., the set of wrenches that can be generated while executing a specific task [55]. This wrench space is even more specific than the OWS (i.e., it is a subset), but it has found limited applications because it is difficult to determine TWS for an arbitrary task [56]. Again, since TWS is a subset of OWS, our method of fast computation of the OWS in [60] can also be used in the calculation for TWS. Figure 2.6 shows an example of the relationship between the unit sphere for grasp wrench space metric, object wrench space and task wrench space. Task wrench space is clearly a subset of object wrench space, while both task wrench space and object wrench space are not subsets of the unit sphere used in grasp wrench space metric. However, if we express the unit sphere as a combination of all unit directional vectors, then task wrench space and object wrench space can be defined as a weighted combination of all unit directional vectors, where the weight can be larger or equal to zero. Different from the

determined measures, [106] proposed the notion of probabilistic force closure in order to consider the uncertainty affecting the object parameters. This idea has been has been applied in [63], were the authors proposed a large-scale cloud-based approach to sampling perturbations of grasps and leveraged multi-armed bandits and deep learning to determine grasps with a high probability of force closure. This is also the fundamental method we will be using to evaluate grasps in chapter 5.

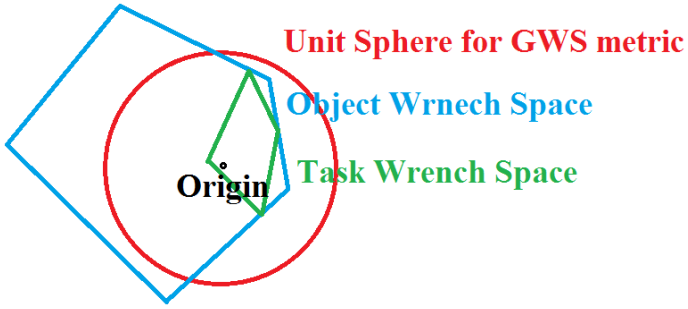



Figure 2.6: The relationship between the unit sphere for GWS metric, object wrench space and task wrench space.

### 2.2.2 Hand Based Measure

The methods we mentioned above focus on external wrench resistance determined by the position of the contact points on the object. However, there exists grasp quality metrics that come from other features. The second group of grasp quality measures consider hand configurations. In [82], a measure was designed to maximize the smallest singular value of the hand Jacobian in order to keep the hand away from singular configurations. This idea was originally used in [45], where the author aims to keep redundant arms away from singular configurations using manipulator Jacobian. The purpose of this metric is to keep the hand configuration away from singularity. But in reality, singularity is not a major concern in grasping. The relation between singularity and stabilty of a grasp configuration is not clearly determined. To achieve better stability with grasping, the authors in [112] considers the volume of the manipulability ellipsoid. This quality indicates a larger velocity of the grasped object is produced when the same velocities are applied in the finger joints. The volume of the manipulability ellipsoid can be thought of as a general guidance reflecting stability. However, when considering a specific problem, the situation is usually biased, such as the manipulability ellipsoid biased along the direction with higher probability of the object to escape is considered to be more stable compared to the manipulability ellipsoid with the largest volume. Similarly, in [87], a measure was constructed based on the uniformity of transformation, i.e., defined as the similiarity in all the components of the object velocity, while the transformation in the velocity domain from the finger joints to the object is uniformly distributed. This measure is also facing the same issue as [112]. However, a recently published measure, in [49], optimizes over the maximum force applied at each contact. This measure is capable of ranking grasps solving both general and specific task related problems. In current models of measures based on disturbance force rejection, the maximum finger force along the object's surface normal direction is considered to be the key

factor in preventing the object from escaping the hand. Larger contact forces can enlarge the grasp wrench space to resist higher wrenches. This measure can be though as holding the object tighter with a limited amount of power. In general, this measure is better compared to the other measures based on hand configuration. But it also faces a problem where the best grasp candidate of this measure might have a configuration with a singularity or close to the mechanic boundary. A way to avoid these kinds of situations would be to set a limit to keep the joints away from the true bound and then use this measure to evaluate grasps.

### 2.2.3 Combing Multiple Measures

Many grasp quality measures have been proposed, but most of them are only focused on a single feature. Therefore, researchers have started to combine multiple features to obtain more comprehensive measurements. The two major ways of combining grasp quality metrics are in serial or in parallel. The serial way means applying multiple measures sequentially. This procedure can be thought as setting multiple thresholds, where the candidate output must satisfy all the thresholds for every measure. In [37], the authors proposed a method to combine two grasp quality measures in serial. The authors used one of the quality criteria to generate candidate grasps, and passed the best candidates forward to be ranked by the next quality measure. However, the serial way is only capable of filtering out grasps that fail to meet the thresholds set by all measures, but it can not provide a reasonable ranking to argue which grasp is the best. In [37], the author does output a ranking, but it was only based on the last measure, which may be biased and might not be any different from just applying the last measure. On the other hand, the parallel approach combines different quality measures into a single global index, which creates a more solid argument for the ranking of the output grasps. A simple addition of unit weighted metrics to choose optimum grasps for 2D and 3D objects was used in [12] and [2], respectfully. The output ranking can be seen as the overall optimum among the two measures the authors used. However, the parallel technique still faces some issues. Deciding the importance for each measure and adding each of them with weights is complicated. Different measures usually have different ranges. For example, the grasp wrench space metric [29] usually ranging from 0 to 1, where the higher bound is limited by the scale of the object or the position of contact points with respect to the object's center of mass. On the other hand, the maximum contact force measure [49] is completely affected by the torque limit of each joint. If only one measure is used or a combination of measures is used in serial, grasp candidates are compared to each other under the same objective. But for the parallel case, if one measure is significantly larger, other measures will have little impact on the objective, which may not be noticeable. Different from these two ways, in chapter 6, we will be introducing our way of combing measures during the construction of GWS. Since we use only one measure combing multiple features as the objective function, we overcome the problems mentioned for both serial and parallel combinations of grasp quality measure. The only problem we face is that since we are adding in features through the force domain, other types of features such as the volume of manipulability ellipsoid are hard to account for.

## 2.3 Convex Hull Computation

The problem of computing the convex hull of $n$ points in $\mathbb{R}^d$ is one of the most studied problems in computational geometry [24,91]. The computational complexity for convex hull algorithms can be described as a function of three parameters, i.e., the number of points $n$, the dimensionality of the space $d$, and the number of facets in the convex hull $h$.

### 2.3.1 Convex Hull in 2D and 3D

For the planar case, i.e., $d = 2$, an $\Omega(n \log n)$ lower bound is easy to prove, and there exist various optimal algorithms matching this bound [111]. Before going into these optimal algorithms we want to introduce the gift wrapping algorithm [40], which is considered as one of the most fundamental algorithms in solving convex hull in 2D. This algorithm first start with the leftmost point and wrap around the point set by comparing polar angles. The complexity of this algorithm is $\Omega(nh)$, where $h$ is the number of vertices in the resulting convex hull, i.e., the size of the output. If the size of $h$ is smaller than $\log n$, then this algorithm will actually perform faster. One of the algorithms that holds the $\Omega(n \log n)$ bound was published by Ronald Graham in 1972, called the Graham's scan algorithm [33]. This algorithm finds all vertices of the convex hull along its boundary following either clockwise or counterclockwise order. A stack is used to detect and remove concave angles on the boundary. This is a very intuitive and straight forward algorithm that meets the optimal lower bound $\Omega(n \log n)$. The downside of this algorithm is that it can be only used in 2D. A variation of this algorithm is Andrew's Monotone Chain Convex Hull algorithm [3]. It first sort the points lexicographically, and then constructs the upper and lower hulls of the points in $\Omega(n)$ following the Graham scan algorithm. Another algorithm that is considered optimal in 2D is Divide and Conquer algorithm published by Preparata and Hong in 1977 [80]. This algorithm uses a basic divide and conquer strategy and is capable to calculate the convex hull in 2d and 3D. Another algorithm, Quickhull [5], uses a divide and conquer approach similar to quicksort [38], and has an average case complexity of $\Omega(n \log n)$. Although, just like quicksort, the worst case of quickhull is $\Omega(n^2)$, it is still considered one of the fastest algorithms to compute convex hull. Moreover, quickhull can be generalized to compute the convex hull in any dimension. Notably, vertices in the convex hull are a subset of the input points. This has motivated the development of algorithms with *output sensitive* complexity. In [17, 43] two algorithms with complexity $O(n \log h)$ were proposed. Note that in $\mathbb{R}^2$ the number of vertices and facets in the convex hull are the same. The first algorithm [43] uses a variation of the divide conquer paradigm called marriage-before-conquest. This algorithm is only applied in 2D and does not generalize to higher dimension naturally. The second algorithm [17] is a simpler algorithm compared to [43], first it divides the point set into smaller sets and use one of the $\Omega(n \log n)$ algorithms to compute each sub convex hull. Then it applies the gift wrapping algorithm on the boundary points the of sub convex hulls to merge them into the final convex hull. This algorithm can be simply generalized to 3 dimensions and keeps the time complexity $\Omega(n \log h)$.

As mentioned before, many algorithms designed for 2D can also be used in 3d while keeping their efficiency [5,17,80]. In $\mathbb{R}^3$, the convex hull of $n$ points can still be computed on $O(n \log n)$ [80] and the number of facets is still linear in the number of vertices in the hull (see e.g., [24], chapter 11).

### 2.3.2 Convex Hull in Higher Dimensions

For $d > 3$ the number of vertices and facets in the resulting convex hull is no longer linear on $n$. It was proposed by Seider [91] that for arbitrary dimension, the number of facets $h$ can grow as $\Theta(n^{\lfloor d/2 \rfloor})$. Therefore, output sensitive algorithms are not necessarily the best option. Clarkson and Shor, in [21], proposed a Las Vegas algorithm, a type of randomized incremental algorithms, with optimal expected complexity $O(n \log n + n^{\lfloor d/2 \rfloor})$. This algorithm was later simplified by Seidel in [90]. The fisrt algorithm that can deterministically compute higher dimension convex hulls in $O(n \log n + n^{\lfloor d/2 \rfloor})$ was proposed by Chazelle [18], and simplified by Bronnimann, Chazelle, and Matousek in [15]. The algorithm Chazelle proposed is a derandomized incremental algorithm, based on derandomizing Clarkson and Shor's algorithm. It is optimal in the worst case but it is not an output sensitive algorithm. Seidel instead proposed an output sensitive algorithm [89] with complexity $O(n^2 + h \log n)$. To compare these two alternative approaches, it is useful to recall that in $\mathbb{R}^d$ the number of facets $h$ can grow as $\Theta(n^{\lfloor d/2 \rfloor})$ (see e.g., [91]), which has become the limitation of run time in high dimensions.

The QuickHull algorithm [5] has become the de-facto standard when it comes to computing convex hulls in higher dimensional spaces. QuickHull improves the randomized algorithm presented in [21] by introducing some heuristics that significantly improve its performance. QuickHull works in the space of points and convex hulls instead of the dual space of halfspaces and polytopes compared to [21]. Also, QuickHull uses less space than most of the randomized incremental algorithms and runs faster for distributions with nonextreme points. Although the QuickHull algorithm is a variation of an randomized incremental algorithm, it is in fact not a randomized algorithm. Instead of picking a random point, the QuickHull algorithm always select the "farthest" point. So far, an analytical characterization of its computational performance has remained elusive, but its observed empirical performance suggests an $O(n \log s)$ complexity, where $s$ is the number of processed vertices. Among the reasons for QuickHull's popularity is the availability of a freely available, highly optimized implementation.[1] Chapter 3 provides a short recap of its principles. Note that convex hull algorithms for grasping are mainly used to generate wrench spaces.

Algorithm wise, the QuickHull algorithm is hard to beat. However, some work was developed to improve the QuickHull algorithm by taking advantage of the hardware system. Although the QuickHull algorithm is an incremental algorithm, where the convex polytope is growing step by step, the point redistribution after creating new faces can be performed in parallel. In [113], the authors proposed a novel implementation of the QuickHull algorithm on the GPU for planner point sets. They claimed that their implementation can achieve the speedups of up to 10.98x. Another work that takes advantage of GPU was published in [94], where they developed a novel parallel algorithm for computing the convex hull of a set of points in 3D using the CUDA programming model. It is said by the authors that their implementation can achieve 30-40 times speedup compared to CPU-based implementation. However, implementations based on GPU of the QuickHull algorithm in dimensions higher than 3 is still missing.

[1] `http://www.qhull.org`

## 2.4 Inverse Kinematics

Inverse kinematics (IK) problems is extensively studied in literature. It has been one of the most fundamental problems for robot manipulation. For a manipulator with $d$ joints the IK problem is defined as follows: given a pose $\mathbf{p} \in SE(3)$, compute a configuration $\mathbf{q} \in R^d$ such that by applying $\mathbf{q}$ to the manipulator, the end-effector pose is $\mathbf{p}$. If the joint number is smaller than six, the IK problem is in general not solvable for arbitrary poses, whereas if the joint number is exactly six, a unique solution can generally be obtained. When the number of joints is greater than six, redundancy is introduced, resulting in a solution space in which a specific configuration can then be selected based on one or more objectives (e.g., clearance from obstacles, used energy, etc.). In general, many manipulators are designed with redundancy. This is due to the fact that multiple IK solutions can be a huge benefit to solve motion planning problem for the arm. For a six degree of freedom arm, the solution for motion planning problems are often limited and the IK solution might not exist when collision is taken into account.

In the literature, IK problems are often solved using Jacobian based approaches and/or iterative methods. General solutions include pseudoinverse methods [107] and Jacobian transpose approaches [108]. These methods solves the IK problem from an initial configuration and use the pseudoinverse or the transpose of the Jacobian matrix to iteratively approach to the target pose. However, these methods lack the ability to handle singularity problems. As an improvement to deal with these limitations, the damped least square method was proposed in [71, 105].

In contrast to iterative methods, closed-form solutions can be computed by analytic approaches. The IKFast method, presented by Diankov in [26], automatically determines a set of equations for closed-form IK solving. The algorithm performs well for solving IK problems with manipulators with up to six degrees of freedom. As mention earlier, a redundunt manipulator has more then six degree of freedom. So as an improvement, Diankov combined IKFast with a discretized sampling strategy in [25] to deal with redundancy. This method can then produce multiple solutions for a redundant manipulator with more than six joints. Our work in chapter 5 is highly dependent on the IKFAST method in determining multiple IK solutions efficiently.

Recently, deep learning attracted a lot of attention from researchers. In [46], a neural-network committee machine was introduced to solve the inverse kinematics of a 6-DOF redundant robotic manipulator. The goal of this work is to improve the precision of the solution while maintaining its efficiency. In [100], the authors proposed an online adaptive strategy based on the Lyapunov stability theory and Radial Basis Function Neural Networks. The results showed good performance in obtaining successful configurations of the robot with a feasible inverse kinematics solution. Since the inverse kinematics problem is often complex, highly nonlinear, coupled and have multiple solutions for complicated redundant manipulator system, deep learning methods are still facing certain limitations. As an improvement, generate-and-test artificial intelligent optimization methods based on the Darwinian principles of biological evolution is proposed in [68] to solve inverse kinematics problem. This method is efficient in producing smooth joints paths while maintaining excellent accuracy along the Cartesian path.

# Chapter 3

# Background and Notation

## 3.1 Background in grasping and grasp metrics

We briefly recap basic facts concerning grasping and grasp metrics, and refer the reader to [70, 79] for more detailed introductions. A rigid body $\mathcal{B}$ is grasped using a multi-fingered hand with $m$ fingers. A grasp $\mathcal{G}$ is specified by the $m$ contact points $\mathbf{p}_1, \ldots, \mathbf{p}_m$ on the surface of $\mathcal{B}$ together with $m$ contact forces $\mathbf{f}_1, \ldots, \mathbf{f}_m$. It is customary to express the coordinates of the contact points with respect to a frame whose origin $O$ coincides with the center of mass of $\mathcal{B}$. Each force generates a *wrench*, i.e., a six-dimensional vector including both the force and the torque with respect to $O$, i.e., $\mathbf{w}_i = [\mathbf{f}_i \;\; \mathbf{p}_i \times \mathbf{f}_i]^T$. To prevent slippage at the contact point, forces are constrained in a given range defined by the assumed friction model. As common in literature, in the following we consider the contact with friction model (also known as hard finger model). More complex contact models can be expressed as multiple contacts of this type, whereas the simpler frictionless model is too restrictive and usually not utilized in practice. Therefore our assumptions is general enough to be useful in many practical scenarios. The hard finger assumption is built upon Coulomb's friction model, i.e., to prevent slippage at the contact point each force must lie inside the friction cone defined at the contact point. More formally, at every contact point $\mathbf{p}_i$ we establish an orthonormal reference frame with vectors $\mathbf{t}_i, \mathbf{u}_i, \mathbf{v}_i$ where $\mathbf{t}_i$ is inward and $\mathbf{u}_i, \mathbf{v}_i$ lie on the plane tangent to $\mathcal{B}$ in $\mathbf{p}_i$. Contact force $\mathbf{f}_i$ can then be expressed as $\mathbf{f}_i = [f_{i1} \; f_{i2} \; f_{i3}]^T$ where $f_{i1}$ is the component along $\mathbf{t}_i$ and $f_{i2}, f_{i3}$ are the components along $\mathbf{u}_i$ and $\mathbf{v}_i$. The set of forces that do not cause slippage at $\mathbf{p}_i$ is then

$$F(\mathbf{p}_i) = \left\{ \mathbf{f}_i \in \mathbb{R}^3 \mid f_{i1} \geq 0 \wedge \sqrt{f_{i2}^2 + f_{i3}^2} \leq \mu f_{i1} \right\}$$

where $\mu$ is the friction coefficient between the object and the fingers, and for simplicity we assume it is constant. The condition $f_{i1} \geq 0$ implies no separation, whereas the second condition follows Coulomb's friction law. Geometrically, $F(\mathbf{p}_i)$ defines a cone. Hence the name friction cone at $\mathbf{p}_i$.

From a practical point of view, it is common to discretize the cone as a regular pyramid with $k$ edges (see Figure 3.1). Using this approximation each contact force lying within inside the can be written as a non-negative combination of forces along the boundary of the friction cone, i.e., $\mathbf{f}_i = \sum_{j=1}^{k} \alpha_{i,j} \mathbf{f}_{i,j}$ with $\alpha_{i,j} \geq 0$. Based on this discretization the wrench generated by the $i$th contact

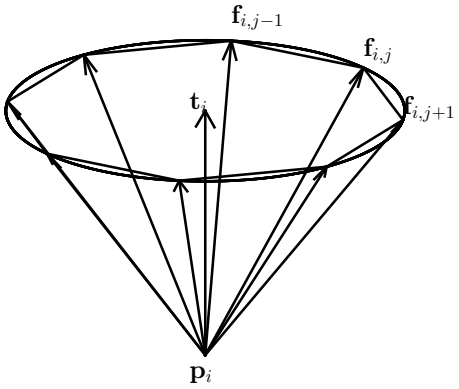



Figure 3.1: The friction cone $F(\mathbf{p}_i)$ is commonly approximated by a regular pyramid with $k$ edges.

force can then be written as

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{p}_i \times \mathbf{f}_i \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{k} \alpha_{i,j}\mathbf{f}_{i,j} \\ \mathbf{p}_i \times \sum_{j=1}^{k} \alpha_{i,j}\mathbf{f}_{i,j} \end{bmatrix} = \sum_{j=1}^{k} \alpha_{i,j} \begin{bmatrix} \mathbf{f}_{i,j} \\ \mathbf{p}_i \times \mathbf{f}_{i,j} \end{bmatrix} = \sum_{j=1}^{k} \alpha_{i,j}\mathbf{w}_{i,j}$$

where each of the $\mathbf{w}_{i,j}$ is called *elementary wrench*. The $km$ elementary wrenches $\mathbf{w}_{i,j}$ can be arranged in a $6 \times km$ matrix $\mathbf{G}$ called the *grasp matrix*. The grasp achieves force closure if for each disturbance wrench $\mathbf{w}$ there exists a vector $\mathbf{x} \in \mathbb{R}^{km}$ such that

$$\mathbf{w} = \mathbf{G}\mathbf{x} \;\; x_i \geq 0, \;\; i = 1, \ldots, km. \tag{3.1}$$

That is to say, the grasp matrix $\mathbf{G}$ positively spans $\mathbb{R}^6$, and in such case $\mathbf{x}$ contains the $\alpha_{i,j}$ factors for all contact points. The set of wrenches obtained through the grasp matrix $\mathbf{G}$ when the sum of the elements in $\mathbf{x}$ is one is known as the *unit grasp wrench space* (UGWS), and this assumption is often made in practice [29, 96]. In this case it is known that an equivalent condition for force closure is that the convex hull of all the $km$ elementary wrenches $\mathbf{w}_{i,j}$ includes the origin (see e.g., [70]).

### 3.1.1 Grasp wrench space metric

Ferrari and Canny formalized the idea that between two different grasps achieving force closure, preference should be given to the one capable of resisting an arbitrary external wrench with minimum effort [29]. This intuition is formalized through the following geometric interpretation. Any wrench exerted on $\mathcal{B}$ through the $m$ contacts can be scaled to belong to the set

$$W_{L_\infty} = \mathrm{CH}\left( \bigoplus_{i=1}^{m} \{\mathbf{w}_{i,1} \ldots \mathbf{w}_{i,k}\} \right) \tag{3.2}$$

where $\oplus$ represents the Minkovski sum. From an operative point of view it is worth observing that inner Minkovski sum in Eq. 3.2 gives a set of finite elements, and therefore $W_{L_\infty}$ is the convex hull of a finite set of elements in $\mathbb{R}^6$. The grasp metric $Q_\infty$ is then defined as the distance of the closest

facet of $W_{L_\infty}$ from the origin, i.e.,

$$Q_\infty = \inf_{\mathbf{x} \in \partial W_{L_\infty}} \|\mathbf{x}\|_2 \tag{3.3}$$

where $\partial W_{L_\infty}$ indicates the boundary of $W_{L_\infty}$, i.e., the union of its facets. Physically, $Q_\infty$ aims at minimizing the maximum force exerted by any of the $m$ fingers to resist an arbitrary external wrench. Alternatively, one could aim at minimizing the sum of forces exerted by all fingers. To this end, consider

$$W_{L_1} = \mathrm{CH}\left(\bigcup_{i=1}^{m} \{\mathbf{w}_{i,1} \dots \mathbf{w}_{i,k}\}\right). \tag{3.4}$$

Similarly to $Q_\infty$, the grasp metric $Q_1$ is then defined as the distance of the closest facet of $W_{L_1}$ from the origin

$$Q_1 = \inf_{\mathbf{x} \in \partial W_{L_1}} \|\mathbf{x}\|_2. \tag{3.5}$$

Note that when $\sum_{i,j} \alpha_{i,j} = 1$, then $W_{L_1}$ is the formerly defined unit grasp wrench space.

The reader is referred to [29] for a thorough explanation of the relationship between the geometrical and physical interpretation of $Q_\infty$ and $Q_1$. It is also important to remark that the two measures are not equivalent, i.e., a grasp that is optimal according to $Q_1$ may not be optimal according to $Q_\infty$ and viceversa. Hence, from a practical standpoint they are both needed. This metric is related to the ability to resist an arbitrary wrench, and is therefore referred to GWS metric.
As evident from Eq. 3.2 and Eq. 3.4, both $Q_\infty$ and $Q_1$ rely on the computation of a convex hull of a set of vectors in $\mathbb{R}^6$, and the connection between grasp quality measures and convex hulls is therefore evident. From a practical standpoint, QuickHull is the algorithm most commonly used to compute these metrics. Despite the fact that these measures are de-facto the most commonly used in practice, as we pointed out earlier, they are affected by the drawbacks we formerly outlined.

### 3.1.2 Object wrench space metric

The metrics described in the previous subsection are somehow too conservative because they consider the set of all possible disturbance wrenches. But from a practical point of view a disturbance wrench is almost invariably caused by a disturbance force, and it would therefore make sense to accordingly restrict the set of disturbance wrenches to be considered. Starting from this observation, Strandberg and Wahlberg defined a method to evaluate grasps based on the ability to reject a disturbance force [96]. The construction is related to the object wrench space (OWS) introduced by Pollard [76], i.e., the set of wrenches that can be generated by an arbitrary unit force $\mathbf{e}_i$ acting on the surface of the object. Assuming that the surface $\partial\mathcal{B}$ of the object is composed by $l$ triangles, OWS can then be defined as follows

$$\mathrm{OWS} = \bigcup_{\mathbf{e}_i} \bigcup_{\mathbf{v}_{j,c}} \mathrm{CH}\left(\begin{bmatrix} \mathbf{e}_i \\ \mathbf{v}_{j,c} \times \mathbf{e}_i \end{bmatrix} \quad j = 1 \dots l,\ c = 1, 2, 3,\ \mathbf{e}_i \in F(\mathbf{v}_{j,c})\right)$$

where CH indicates the convex hull, $\mathbf{v}_{j,c}$ is the $c-$th vertex of the $j-$th triangle on the triangle mesh, and $\mathbf{e}_i$ is a unit length force vector. Note: In the definition the force, $\mathbf{e}_i$ is constrained to be inside the friction cone $F(\mathbf{v}_{j,c})$ at the contact point. Under the hypothesis that the surface is represented with

a triangulation, the computation can be limited to the vertices of the mesh [96]. The OWS leads to a different grasp quality metric defined as follows. Assume a grasp $\mathcal{G}$ is given together with its associated grasp matrix $\mathbf{G}$. Let $\mathbf{e}$ be a unit vector specifying the direction of a disturbance force. All disturbance forces in that direction can be written as $f\mathbf{e}$ with $f \geq 0$ to avoid tractional forces. By *sweeping* $\mathbf{e}$ over the surface of $\mathcal{B}$ it is possible to characterize all disturbance wrenches generated by forces parallel to $\mathbf{e}$. Let $f^*$ be the smallest value such that the associated wrench is exactly on the border of the unit grasp wrench space. Varying $\mathbf{e}$, one gets to the following min-max formulation

$$f^* = \min_{\mathbf{p}} \max_{\mathbf{x} \in \mathbb{R}^{km}} \left\{ f \in \mathbb{R}^+ : -f \begin{bmatrix} \mathbf{e} \\ \mathbf{p} \times \mathbf{e} \end{bmatrix} = \mathbf{G}\mathbf{x}\mathbf{e} \in F(\mathbf{p}),\ \mathbf{p} \in \partial\mathcal{B},\ x_i \geq 0,\ \sum_{i=1}^{km} x_i = 1 \right\}$$

where the contact point $\mathbf{p}$ is constrained to be on the surface $\partial\mathcal{B}$ of the object being grasped, and the force $\mathbf{e}$ is constrained to be inside the friction cone $F(\mathbf{p})$ at the contact point. $f^*$ is then the intensity of the maximum disturbance force that can be resisted by the unit grasp $\mathcal{G}$ associated with $\mathbf{G}$ and can be used to define the grasp quality metric

$$Q_{\text{OWS}} = \max \{r > 0 \mid r \cdot \text{OWS} \subseteq \text{GWS}\} \tag{3.6}$$

where $r \cdot$OWS is the set obtained multiplying all elements of OWS by $r$. In the following $r$ is also referred to as *inflation factor* because its role is to expand or shrink OWS. From a computational standpoint, to consider all possible directions for the disturbance force one expresses $\mathbf{e}$ in spherical coordinates and then accordingly samples the associated space of angles $\varphi, \theta$ (see [96] for details.) Note that in the original formulation an additional wrench $\mathbf{w}_0$ accounting for the gravity force is included, but without loss of generality, we do not consider it here.

The advantages of this metric are manifold. It is scale and reference invariant, it explicitly includes the shape of the object being grasped, and it considers just the forces that appear in practice. Moreover, one can determine the optimal $f$ value varying $\mathbf{e}$ over the unit sphere, thus leading to a graphical interpretation in three dimensions that is easy to visualize and understand (see [96] for examples.) The major downside is that the method is computationally demanding and therefore it is not practical to use it to guide a search over the space of possible grasps.

## 3.2 Background in Convex Hulls

### 3.2.1 Convex Hulls

We summarize some facts about computational geometry and convex sets. The reader is referred to [24] for more details.

First, let us start with some definitions.

**Definition 1.** *Let $\mathcal{S}$ be a subset of $\mathbb{R}^d$. $\mathcal{S}$ is a* convex *set if for each $x, y \in \mathcal{S}$ and each $\lambda \in [0, 1]$ the point $\lambda x + (1 - \lambda)y$ is an element of $\mathcal{S}$.*

**Definition 2.** *Let $\mathcal{N}$ be a set of n points in $\mathbb{R}^d$. The* convex hull *of $\mathcal{N}$ is the smallest convex set including $\mathcal{N}$. The convex hull of $\mathcal{N}$ will be indicated as* CH($\mathcal{N}$).

**Definition 3.** *Let* CH($\mathcal{N}$) *be the convex hull of* $\mathcal{N}$. *A facet* $\mathcal{F}$ *of* CH($\mathcal{N}$) *is a convex set determined by the vertices of* CH($\mathcal{N}$) *that lies on the same hyperplane.*

It is known that CH($\mathcal{N}$) is the intersection of all convex sets including $\mathcal{N}$. Moreover, CH($\mathcal{N}$) is the intersection of a finite number of half spaces in $\mathbb{R}^d$. Given a set $\mathcal{N}$ of $n$ points in $\mathbb{R}^d$, its convex null CH($\mathcal{N}$) is a convex polytope in $\mathbb{R}^d$ that can be represented by its vertices and its facets. Each of its facets $\mathcal{F}_i$ is a convex subset of a $(d-1)$-dimensional hyperplane $\mathcal{H}_i$ and each vertex of CH($\mathcal{N}$) is an element of $\mathcal{N}$. The hyperplane $\mathcal{H}_i$ including facet $\mathcal{F}_i$ is also called the hyperplane supporting $\mathcal{F}_i$. In the following, we will also write $\mathcal{H}(\mathcal{F})$ to indicate the hyperplane supporting facet $\mathcal{F}$. Hyperplane $\mathcal{H}_i$ splits $\mathbb{R}^d$ in two halfspaces, one of which contains CH($\mathcal{N}$). The *inside set* of hyperplane $\mathcal{H}_i$ is defined as the half space including CH($\mathcal{N}$), and it will be indicated as $\mathcal{I}(\mathcal{H}_i)$. Similarly, the *outside set* of $\mathcal{H}_i$ is the half space not including CH($\mathcal{N}$) and it is indicated as $\mathcal{O}(\mathcal{H}_i)$. We associate to $\mathcal{H}_i$ a unit normal vector $\mathbf{n}_i$ directed towards its outside set $\mathcal{O}(\mathcal{H}_i)$.

**Definition 4.** *A simplex in d dimensions is a polytope that is the convex hull of its* $d+1$ *vertices.*

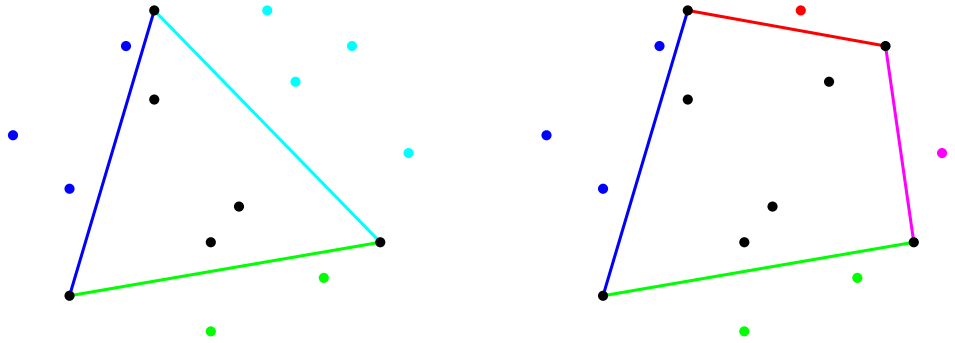
It is known that a simplex in $d$ dimensions has $d+1$ facets.

### 3.2.2 The QuickHull algorithm

We shortly recap how the QuickHull algorithm works and we refer the reader to [5] for a more thorough discussion. Assume the $n$ points in $\mathcal{N}$ are in general position, i.e., they do not all lie on the same $(d-1)$-dimensional hyperplane. If the points are not in general position, a small perturbation can be applied to remove this degenerate condition. QuickHull computes CH($\mathcal{N}$) as follows. An initial simplex is created selecting $d+1$ points from $\mathcal{N}$. If possible, QuickHull picks points with an extreme coordinate, though this is not necessary for the correctness of the algorithm. Every facet $\mathcal{F}_1, \ldots, \mathcal{F}_d, \mathcal{F}_{d+1}$ in the initial simplex is supported by an hyperplane. For every facet $\mathcal{F}_i$, its outside set[1] is defined as the set of all points in $\mathcal{N}$ located in the outside set of the supporting hyperplane $\mathcal{H}_i$. Following the same notation introduced for the supporting hyperplanes, the inside and outside sets of facet $\mathcal{F}_i$ will be indicated as $\mathcal{I}(\mathcal{F}_i)$ and $\mathcal{O}(\mathcal{F}_i)$, respectively. Note that in general a point in $\mathcal{N}$ may belong to more than one outside set, and an outside set can also be empty (see Figure 3.2.a).

From the initial simplex, the convex hull is iteratively grown as follows. If there exists a facet with a non empty outside set, the convex hull is expanded by growing the convex hull to include the point in its outside set that is the farthest from the facet. This step is called *expansion*. Expansion eliminates the facet being expanded, and generates a set of new facets for which their respective outside sets are created (see Figure 3.2.b). Once the outside sets of all facets are empty, the process terminates. By *greedily* growing the convex hull towards the farthest point in outside set, it was empirically shown that QuickHull outperforms other algorithms utilizing different criteria to expand the current hull, e.g., [21].

[1] Note that in QuickHull the outside set of a facet is given by a set of points, whereas in our general definition $\mathcal{O}(\mathcal{H}_i)$ is an half space. With a slight abuse of notation we use the same term to avoid introducing additional symbols.



(a) (b)

Figure 3.2: a) QuickHull initialization. In $\mathbb{R}^2$ the initial simplex includes 3 points. The outside set of each facet is displayed using the same color of the facet. Black points are inside the simplex and do not belong to any outside set. b) QuickHull expansion. The cyan facet is expanded and replaced by two new facets (red and purple), and their respective outside sets are updated.

Throughout its computation, QuickHull maintains a list of the vertices and facets of the convex hull being built. Moreover, for each facet $\mathcal{F}_i$ it maintains also the normal $\mathbf{n}_i$ associated with the supporting plane $\mathcal{H}_i$.

# Chapter 4

# Efficient Grasp Quality Evaluation through Partial QuickHull Computation

In this chapter we present two algorithms that greatly expedite the computation of two formerly proposed grasp quality metrics. The first was introduced by Ferrari and Canny in [29]. It relies on the construction of a convex hull in six dimensions and it is extensively used in practice. Background material and notation regarding convex hulls is given in chapter 3. Although this metric is used extensively in practice, it has many drawbacks, which we aim to address. For example, it is not scale invariant, it does not consider the shape of the object being grasped and is too conservative because it considers a set of all possible disturbances, including those that are unlikely to occur in practice. In the following we refer to this measure as the grasp wrench space (GWS) metric because it considers all possible disturbance wrenches. The second metric we examine was proposed by Strandberg and Wahlberg [96] and, in principle, it offers notable advantages when compared to [29]. Particularly, it is scale invariant, it takes the shape of the object into consideration and it focuses on disturbances that are likely to occur in practice. However, so far it has rarely been employed because there is no efficient method to compute this metric. In the following we refer to this measure as the object wrench space (OWS) metric because it considers only the disturbance wrenches that can occur on the object being restrained. Our algorithms are built upon a common observation: both metrics can be determined by computing a partial convex hull. Together, both algorithms start building a six-dimensional convex hull, but through exploiting computational geometry, they can identify and can be stopped when enough information has been computed to obtain the needed values. We have named this approach "partial quick hull computation."

The contributions of this chapter are the following.

- We present an algorithm (PQHGWS – Partial Quick Hull for Grasp Wrench Space) to efficiently compute the metric proposed in [29]. The algorithm can be applied for both force closure and non-force closure grasps. In the first case, it will return the grasp quality measure, whereas in the second case, it will return the distance of the convex hull from the origin - a value that can be used to guide the planning process.

- We present an algorithm (PQHOWS – Partial Quick Hull for Object Wrench Space) to efficiently compute the metric proposed in [96]. PQHOWS is faster than the brute force method,

is faster than a previously used method to compute an approximation of the metric, and is comparable time wise to PQHGWS. If the grasp being evaluated is a non-force closure grasp, PQHOWS quickly identifies the metric as not defined and immediately returns the appropriate flag. With these improvements, the metric proposed in [96] can be used in lieu of [29] during grasp planning.

- We provide the theoretical foundations supporting of our methods.
- We show that our algorithms offer significant advantages when compared to current methods through experimental data.

Note that the two ways we deal with non-force closure grasp, i.e., calculating the distance of the convex hull from the origin and quickly identifying this case returning an appropriate flag, can replace each other. That is to say, PQHOWS can approach a non-force closure grasp by calculating the distance of the convex hull from the origin while PQHGWS can quickly identify this case to return appropriate flag.

The code implementing our findings and the datasets used to generate the results presented in this chapter are made freely available to the scientific community.

This chapter is organized as follows. Our two algorithms are presented in section 4.1 and 4.2. Extensive experimental results substantiating the computational advantages of our methods are given in section 4.3, and conclusions are presented in section 4.4.

## 4.1 Grasp Wrench Space Metric with Partial Convex Hulls

In this section we show how the computation of the grasp metric presented in section 3.1.1 can be greatly accelerated. When considering a force closure grasp, both $Q_1$ and $Q_\infty$ are defined as the radius of the largest ball centered in the origin and fully inside a suitably defined convex hull. If the grasp is not force closure, the origin is outside the hull and the grasp quality measure is not defined. However, the distance between the origin and the convex hull still provides valuable information because it indicates how far a grasp is from achieving force closure [115]. Therefore, for both force and non-force closure grasps, Eq. (3.3) and (3.5) provide an informative value. In both cases it is not necessary to compute the entire convex hull, since the distance, and then the metric, is exclusively determined by the closest facet of the hull. The algorithm we propose, labeled Partial Quick Hull for grasp wrench space, builds upon this observation. From a practical standpoint, the convex hull is iteratively grown using the same principles used in QuickHull, but its computation is stopped when the closest facet is determined. The following definitions introduce the relevant quantities we use to describe the algorithm. We then follow the supporting lemmas to prove our proposed approach.

**Definition 5.** *Let $\mathcal{H}$ be a hyperplane in $\mathbb{R}^d$ and $\mathbf{y}$ a point in $\mathbb{R}^d$. The Euclidean distance between $\mathcal{H}$ and $\mathbf{y}$ is indicated as $d(\mathcal{H}, \mathbf{y})$.*

**Definition 6.** *An oriented hyperplane is an hyperplane $\mathcal{H}$ associated with a unary vector $\mathbf{n}_\mathcal{H}$ orthogonal to $\mathcal{H}$.*

The role of $\mathbf{n}_{\mathcal{H}}$ is to identify one of the two halfspaces defined by $\mathcal{H}$. As QuickHull and PQHGWS incrementally build the resulting convex hull, both algorithms maintain for each facet $\mathcal{F}$, an orthogonal unit-length vector pointing into the outside set of the supporting hyperplane $\mathcal{H}$. The definition of the outside set is given in chapter 3.
*Assumption:* when considering a convex hull, the hyperplanes supporting its facets will always be oriented with a unit vector pointing into the outside set.

**Definition 7.** *Let $\mathcal{H}$ be an oriented hyperplane in $\mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^d$. We define $o(\mathcal{H}, \mathbf{y})$ (offset from $\mathbf{y}$ to hyperplane $\mathcal{H}$) as*

$$o(\mathcal{H}, \mathbf{y}) = d(\mathcal{H}, \mathbf{y})\text{sgn}((\mathbf{p}_{\mathcal{H}} - \mathbf{y}) \cdot \mathbf{n}_{\mathcal{H}})$$

*where $\mathbf{p}_{\mathcal{H}}$ is any point on the hyperplane $\mathcal{H}$, $\cdot$ is the dot product, and* sgn *is the signum function.*

From the above definitions it follows that $d(\mathcal{H}, \mathbf{y})$ is always non-negative, whereas $o(\mathcal{H}, \mathbf{y})$ can be positive, negative, or zero. The offset $o(\mathcal{H}, \mathbf{y})$ can be interpreted as a *signed distance*, i.e., it is $d(\mathcal{H}, \mathbf{y})$ when $\mathbf{y}$ is in the inside set, whereas it is $-d(\mathcal{H}, \mathbf{y})$ when $\mathbf{y}$ is in the outside set.

**Definition 8.** *Let $\mathcal{F}$ be a facet of a convex hull and $\mathbf{y}$ be a point in $\mathbb{R}^n$. The distance between $\mathbf{y}$ and $\mathcal{F}$ is*

$$d(\mathcal{F}, \mathbf{y}) = \min_{\mathbf{x} \in \mathcal{F}} \|\mathbf{y} - \mathbf{x}\|_2.$$

**Definition 9.** *Let $\mathcal{H}$ be a hyperplane in $\mathbb{R}^d$ [1] and $\mathbf{y} \in \mathbb{R}^d$. The projection of $\mathbf{y}$ into $\mathcal{H}$ is the point $\mathbf{x}$ obtained by the intersection between $\mathcal{H}$ and the line through $\mathbf{y}$ orthogonal to $\mathcal{H}$. The projection length of $\mathbf{y}$ into $\mathcal{H}$ is the Euclidean distance between $\mathbf{y}$ and $\mathbf{x}$.*

**Definition 10.** *Let $\mathcal{F}$ be a facet on the convex hull, $\mathcal{H}$ be the hyperplane supporting $\mathcal{F}$, and $\mathcal{H}(\mathbf{y})$ the projection of $\mathbf{y}$ onto $\mathcal{H}$. We define $o(\mathcal{F}, \mathbf{y})$ (offset from point $\mathbf{y}$ to facet $\mathcal{F}$) as*

$$o(\mathcal{F}, \mathbf{y}) = \begin{cases} o(\mathcal{H}, \mathbf{y}) & \text{if } \mathcal{H}(\mathbf{y}) \in \mathcal{F} \\ \text{sgn}(o(\mathcal{H}, \mathbf{y})) \cdot d(\mathcal{F}, \mathbf{y}) & \text{otherwise.} \end{cases}$$

According to definition 8, $d(\mathcal{F}, \mathbf{x})$ can be solved as a constrained optimization problem using methods such as the interior point method or the conjugate gradient method. However, the solution is easy to determine by a few simple calculations.

Let us start with the calculation in 2D. A convex hull in 2D is a convex polygon where all the facets are represented as an edge $E$ with two vertices $V_1$ and $V_2$. The distance between a 2D point $P_0$ and $E$ yields to two cases, depending on the projection of $P_0$ on the line supporting $E$, as shown in figure 4.1. If the projection of $P$ on the line supporting $E$ is within $E$, then the distance is equal to the length of the segment connecting $P_0$ and $P_v$, where $P_v$ is the projection of $P_0$ into $E$. Otherwise, the distance is equal to the distance between $P_0$ and the closest vertex of $E$, e.g., $V_1$, as shown in the right part of figure 4.1.

Next, we consider the calculation in 3D. Faces in 3D are determined by at least 3 vertices. However, if a face contains more than 3 vertices, then these vertices must be lying in the same

[1] $\mathcal{H}$ is a line in 2D and a plane for 3D.

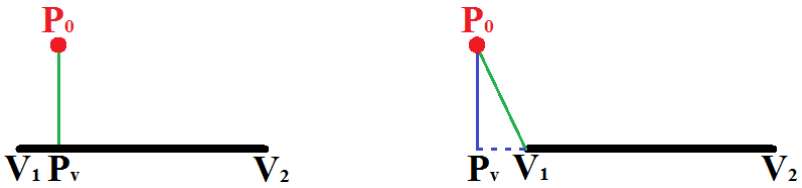



Figure 4.1: Two cases for calculating the distance between a point and a facet in 2D. The green line shows the distance between point $P_0$ and the edge defined by $V_1$ and $V_2$.

plane. Assume a facet $F$ in 3D is represented with $n$ vertices $V_1$, $V_2$ ... $V_n$. The distance between $P_0$ and $F$ also depends on the projection of $P_0$ on the plane supporting $F$. Assume the projection of $P_0$ on the plane supporting $F$ is $P_0^p$. If $P_0^p$ is within $F$, then the distance is equal to the projection length, $l_p$, determined by the distance between $P_0$ and $P_0^p$. An example for this case is shown in figure 4.2(a). Otherwise, we need to check the projection of $P_0^p$ on the edges of $F$. If the projection of $P_0^p$ on the line supporting any edge is within the edge, assume the projection length is $l_e$, then a possible distance between $P_0$ and $F$ is equal to $\sqrt{l_p^2 + l_e^2}$. This case is shown in figure 4.2(b), where the corresponding edge is the one determined by $V_1$ and $V_2$. However, there exists another possible case, shown in figure 4.2(c), where the distance between $P_0$ and $F$ is equal to the distance between $P_0$ and the closest vertex of $F$, i.e., $V_2$.

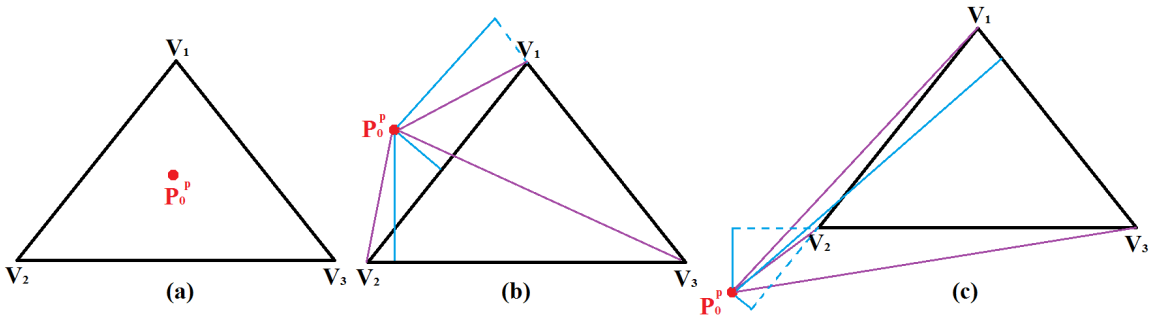



Figure 4.2: Three cases for calculating the distance between a point and a facet represented as a triangle in 3D.

In conclusion we can calculate $d(\mathcal{F}, \mathbf{x})$ defined in definition 8 as follows:

- If the projection of $\mathbf{x}$ on the hyperplane supporting $\mathcal{F}$ is within $\mathcal{F}$, then $d(\mathcal{F}, \mathbf{x})$ is equal to the projection length of $\mathbf{x}$ on $\mathcal{F}$.
- Otherwise, $d(\mathcal{F}, \mathbf{x})$ is equal to the minimum projection length between $\mathbf{x}$ and all sub-facets of $\mathcal{F}$, which the projection of $\mathbf{x}$ onto the sub-facet is within the sub-facet, from dimension $n-1$ to 1. Note that a sub-facet in dimension $i$ of a facet in dimension $n$, where $i < n$, is defined as the convex combination of $i$ non-linear vertices of this facet.

  Note that for dimension 1, the sub-facets are represented as points. We assume the projection of a point to another point is the other point.

From these definitions we can state the two lemmas supporting the PQHGWS algorithm.

**Lemma 1.** *Let* $\mathrm{CH}(\mathcal{N})$ *be the convex hull of n points in* $\mathbb{R}^d$ *and* $\mathbf{x}$ *be a point in* $\mathbb{R}^d$*. If* $\mathbf{x}$ *is inside* $\mathrm{CH}(\mathcal{N})$*, then the offset from point* $\mathbf{x}$ *to all the oriented hyperplanes supporting all facets of* $\mathrm{CH}(\mathcal{N})$ *is larger than 0. If* $\mathbf{x}$ *is outside* $\mathrm{CH}(\mathcal{N})$*, there exists at least one facet of* $\mathrm{CH}(\mathcal{N})$ *for which* $\mathbf{x}$ *has negative offset from the supporting hyperplane.*

*Proof.* $\mathrm{CH}(\mathcal{N})$ is the intersection of all the inside sets of the hyperplanes supporting its facets (see e.g., [24]), i.e.,

$$\mathrm{CH}(\mathcal{N}) = \bigcap_i \mathcal{I}(\mathcal{H}_i)$$

where $\mathcal{H}_i$ is the supporting plane for the $i$-th facet and $i$ varies over all the facets of $\mathrm{CH}(\mathcal{N})$. If $\mathbf{x}$ is inside $\mathrm{CH}(\mathcal{N})$ then it is in the inside set of all the hyperplanes supporting its facets. As per definition 7, $o(\mathcal{H}_i, \mathbf{x})$ is then positive. If $\mathbf{x}$ is outside $\mathrm{CH}(\mathcal{N})$, then there exists at least one facet $\mathcal{F}_i$ such that $\mathbf{x} \notin \mathcal{I}(\mathcal{H}_i)$. Therefore $\mathbf{x}$ is in the outside set of $\mathcal{H}_i$, with its offset $o(\mathcal{H}_i, \mathbf{x})$ as negative, per definition 7. □

**Lemma 2.** *Let* $\mathbf{x}$ *be inside* $\mathrm{CH}(\mathcal{N})$ *and let* $\mathcal{F}$ *be a facet of* $\mathrm{CH}(\mathcal{N})$ *with the smallest value of* $d(\mathcal{F}, \mathbf{x})$*. Then* $d(\mathcal{F}, \mathbf{x}) = o(\mathcal{H}, \mathbf{x})$*, i.e., the distance of* $\mathbf{x}$ *from* $\mathcal{F}$ *is equal to the offset of* $\mathbf{x}$ *from the hyperplane* $\mathcal{H}$ *supporting* $\mathcal{F}$*.*

*Proof.* Let $\mathcal{H}$ be the plane supporting $\mathcal{F}$. Observe that it must be $\mathcal{H}(\mathbf{x}) \in \mathcal{F}$, i.e., the projection of $\mathbf{x}$ onto the plane supporting $\mathcal{F}$ must be inside $\mathcal{F}$. If this were not the case, then the line through $\mathbf{x}$ and orthogonal to $\mathcal{H}$ would intersect another supporting plane before intersecting $\mathcal{H}$ outside $\mathcal{F}$. However, this would contradict the hypothesis that $\mathcal{F}$ is the closest facet. Then the claim follows as per the first case in definition 10. □

Algorithm 1 sketches the details of PQHGWS. The algorithm creates the initial simplex as in QuickHull (line 1). Inside the main loop the algorithm looks for the next facet to expand, $\mathcal{F}_e$. The first for loop (line 4 to 6) establishes if the grasp is force closure. If the grasp is force closure, as indicated by the boolean variable FCflag, the facet with the smallest offset from the origin (line 8) is expanded. Note that in this case all facets have positive offsets, so the search for the facet to expand happens on the whole set of facets. However, if the grasp cannot be determined as a force closure yet, the facet with the largest offset amongst those with negative offsets (line 10) will be expanded. In both cases, if the outside set of expanding facet is empty, then the closest facet has been calculated and the computation is terminated (line 12). Otherwise, the facet is expanded (line 14), and a new loop starts.

According to definitions 7 and definition 10, calculating $o(\mathcal{H}, 0)$ is simpler compared to calculating $o(\mathcal{F}, 0)$. In order to calculate $o(\mathcal{F}, 0)$, we first need to check whether the projection of the origin on $\mathcal{F}$ is within $\mathcal{F}$. If the projection is inside $\mathcal{F}$, $o(\mathcal{H}, 0)$ can be used. Otherwise, we have to iterate over all sub-facets of $\mathcal{F}$ to identify the correct value of $o(\mathcal{F}, 0)$. It is important to notice when FCflag is set, we just need to calculate $o(\mathcal{H}, 0)$, as shown in line 8, instead of $o(\mathcal{F}, 0)$. The correctness of this step is guaranteed by Lemma 2. For the case where FCflag is not set, i.e., some facets have a negative offset to the origin, we must calculate $o(\mathcal{F}, 0)$ for all facets with negative values of $o(\mathcal{H}, 0)$, as shown in line 10.

**Algorithm 1** PQHGWS algorithm

```
1:  Create initial simplex CH with d + 1 points
2:  loop
3:      FCflag ← true
4:      for all F ∈ CH do
5:          if o(H(F), 0) < 0 then
6:              FCflag ← false
7:      if FCflag then
8:          F_e ← arg min o(H(F), 0)
9:      else
10:         F_e ← arg max{o(F, 0) | o(H, 0) < 0}
11:     if O(F_e) = ∅ then
12:         return o(F_e, 0)
13:     else
14:         Expand F_e
```

It is immediate to observe that the algorithm is guaranteed to terminate, because every iteration expands one facet, so that eventually, all facets will have an empty set.

### Non-force Closure Grasp Solution

As we indicated in the earlier section, we can quickly terminate with a return value equal to the distance from the GWS to the origin for non-force closure grasps. This is guaranteed by line 10 of the algorithm. For a non-force closure case, if the facet with the largest offset among those with a negative offset can not be expanded, then no point that is currently outside of the convex simplex can grow the convex hull towards the origin. Thus, the distance from the GWS to the origin is determined at this point. Furthermore, greedily choosing the facet with the largest negative offset to expand will speed up the process to include the origin for a force closure grasp.

### Computational Complexity

QuickHull is observed to be competitive in practice, but its computational complexity has not been formally determined [5] and its performance characterization is mostly empirical. Similarly, no accurate computational complexity analysis is available for PQHGWS. For each iteration of PQHGWS, we need to scan all of the facets to identify the closest facet to expand. On the other hand, it not necessary for QuickHull, because all facets without empty outside sets can be chosen for expansion. However, QuickHull exploits the heuristic observation that choosing the point farthest from the current convex polytope will accelerate the overall process. Therefore, instead of looking for the first facet with a non-empty outside set, it iterates over the facet list to search for the point that is furthest to the current hull. This strategy aligns the cost for each iteration of PQHGWS to QuickHull. Other than this, PQHGWS follows the same framework as QuickHull but terminates early when the closest facet with an empty outside set is found. Due to these facts, it is safe to argue

that by construction QuickHull's (conjectured based on a variety of experiments) computational complexity $O(n \log s)$ is an upper bound for PQHGWS.

## 4.2 Object Wrench Space Metric with Partial Convex Hulls

In this section we present an algorithm to efficiently compute the grasp metric based on the object wrench space described in section 3.1.2. Before starting, it is important to recall that $Q_{\text{OWS}}$ is defined only for force closure grasps. Therefore when computing its value, one has to verify whether the GWS associated with the grasp being evaluated includes the origin or not, otherwise the metric is not defined. Our algorithm includes a greedy initialization step aimed at determining whether or not the grasp is force closure. In the latter case, the computation terminates, indicating that the metric is undefined. As per Eq. (3.6), to compute $Q_{\text{OWS}}$ one needs to determine the inflation factor $r$, such that the boundary of OWS touches the boundary of GWS. If $Q_{\text{OWS}}$ is used to guide a grasp planner, the shape of the object is fixed. So, OWS can be pre-computed and will not change. On the contrary, every time a new grasp is evaluated, a new GWS is needed. The following theorem provides a first step towards simplifying the computation of the metric.

**Theorem 1.** *Let OWS be an object wrench space and GWS be a grasp wrench space. Then,*

$$\max\{r \mid r \cdot CH(OWS) \subseteq GWS\} = \max\{r \mid r \cdot OWS \subseteq GWS\}.$$

*Proof.* By definition, OWS $\subseteq$ CH(OWS). Therefore

$$\max\{r | r \cdot \text{CH(OWS)} \subseteq \text{GWS}\} \leq \max\{r | r \cdot \text{OWS} \subseteq \text{GWS}\}.$$

To see that the inequality cannot be strict, we have to remember that each vertex of CH(OWS) is also a vertex of OWS. Let

$$r^* = \arg\max\{r | r \cdot \text{CH(OWS)} \subseteq \text{GWS}\}.$$

Since GWS and CH(OWS) are both convex sets by construction, the intersection between $r^* \cdot$ CH(OWS) and GWS will always occur at a vertex[2] of CH(OWS). Since such vertex is also a vertex of OWS, $r^*$ is also the optimal inflation rate for OWS. Consequently, the inequality cannot be strict.

□

According to Theorem 1 we can conclude that the largest inflation factor $r$ is obtained when one of the vertices of $r \cdot$ CH(OWS) intersects one of the facets of GWS. Note that this statement is true in general, including when the intersection happens between two vertices, or between two facets. Therefore the expression to compute $Q_{\text{OWS}}$ can be further rewritten as

$$Q_{\text{OWS}} = \min_r \left\{ r \mid r \cdot \mathbf{v}_i \bigcap \text{GWS} \neq \varnothing, \mathbf{v}_i \in \mathcal{V}(\text{CH(OWS)}) \right\}$$

[2] In the special case in which the intersection happens between two parallel facets, the reasoning still holds considering one vertex of the facet of CH(OWS).

where $\mathcal{V}(\text{CH(OWS)})$ is the set of vertices defining the convex hull of OWS. This last expression can be used to derive a brute force algorithm to compute $Q_{\text{OWS}}$. For example, for each vertex $\mathbf{v}_i \in \mathcal{V}(\text{CH(OWS)})$ and each facet $\mathcal{F}_j$ in GWS we can compute

$$r_{i,j} = \frac{o(\mathcal{H}_j, 0)}{\mathbf{v}_i \cdot \mathbf{n}_{\mathcal{H}_j}} \tag{4.1}$$

where $o(\mathcal{H}_j, 0)$ is the offset from the origin of the hyperplane $\mathcal{H}_j$, supporting $\mathcal{F}_j$ and $\mathbf{n}_{\mathcal{H}_j}$ is the outward normal to $\mathcal{H}_j$. $r_{i,j}$ is the factor by which which we need inflate vertex $\mathbf{v}_i$ so that it lies on hyperplane $\mathcal{H}_j$ (note that this value may also be negative). Based on these values, the needed metric is then

$$Q_{\text{OWS}} = \min\{r_{i,j} | r_{i,j} > 0\}. \tag{4.2}$$

While correct, the brute force method is not practical because it has complexity $\Theta(V_{\text{OWS}} H_{\text{GWS}})$, where $V_{\text{OWS}}$ is the number of vertices defining the OWS convex hull and $H_{\text{GWS}}$ is the number of hyperplanes in the convex hull of GWS. To fully comprehend the limitations of this approach, it is useful to recall [91] that the convex hull of $n$ points in $\mathbb{R}^d$ can include up to $\Theta(n^{\lfloor d/2 \rfloor})$ hyperplanes. Since we are operating in $\mathbb{R}^6$ it follows that $H_{\text{GWS}}$ can grow as $\Theta(V_{\text{GWS}}^3)$ where $V_{\text{GWS}}$ is the number of vertices in GWS. Another computational challenge comes from the discretization of the friction cones. To reduce the impact of the approximation, one has interest in increasing $k$, i.e., the number of edges in the pyramid approximating the cone (see figure 3.1). However, increasing $k$ further increases $V_{\text{GWS}}$. Due to these computational costs, the $Q_{\text{OWS}}$ metric has been, so far, rarely used in practice during the planning stage. In the following, we propose an efficient algorithm to compute $Q_{\text{OWS}}$, based on principles similar to the partial convex hull computation we introduced in section 4.1. This expedient greatly accelerates the computation.

Starting from the observation that CH(OWS) can be precomputed upfront, the idea is to iteratively grow GWS and to stop the process as soon as the correct value of $Q_{\text{OWS}}$ can be determined. Recall that by definition GWS is a convex hull, so this iterative growing step has some aspects in common with the algorithm for PQHGWS. To determine when the computation can be stopped, each vertex of OWS is assigned to one of the facets in the partial convex hull of GWS being computed. The association is done so that the inflation factor in Eq. (4.1) is computed only between associated vertices and facets, and not among all possible vertices and facets. The main challenge, is in efficiently establishing and updating these associations while the convex hull is iteratively growing with facets being added and deleted. The following lemma provides the foundation for our algorithm.

**Lemma 3.** *Let $\mathcal{V}$ be the set of vertices in* CH(OWS) *and $H$ be the set of hyperplanes supporting the facets of the convex hull* GWS. *If $\mathbf{v}_i \in \mathcal{V}$ and $\mathcal{H}_j \in H$ achieve the minimum in the expression given by Eq. 4.2, then $r_{i,j}\mathbf{v}_i$ lies in the facet supported by $\mathcal{H}_j$.*

*Proof.* According to the definition of $r_{i,j}$ and elementary geometry, $r_{i,j}\mathbf{v}_i$ lies on $\mathcal{H}_j$ for every point $\mathbf{v}_i$ and every hyperplane $\mathcal{H}_j$. Let $\mathbf{v}'_i = r_{i,j}\mathbf{v}_i$ be such point, and by contradiction assume $\mathbf{v}'_i$ is not on a facet of GWS. Then there must exist a positive $r'_{i,j} < r_{i,j}$ such that $r'_{i,j}\mathbf{v}_i$ intersects a facet $\mathcal{F}_k$ supported by hyperplane $\mathcal{H}_k$. Then by definition $r'_{i,j}$ must be the inflation factor between vertex $\mathbf{v}_i$

and hyperplane $\mathcal{H}_k$, i.e., $r'_{i,j} = r_{i,k}$. However, this contradicts the hypothesis that $r_{i,j}$ achieves the minimum in Eq. (4.2) and the claim follows. □

The algorithm works as follows. As first step CH(OWS) is computed. Let $\mathcal{V}$(CH(OWS)) be the set of vertices of this convex hull. Then, a simplex for the space of elementary wrenches is initialized with $d + 1$ vertices, using the same initialization procedure used by QuickHull. Since we work in $\mathbb{R}^6$ the initial simplex features 7 points. The initial convex hull is then greedily expanded with the facet that has the smallest negative offset until it includes the origin. This precondition is necessary to ensure the correctness of the successive steps. If this expansion fails, i.e., the facet being expanded has an empty outside set, then it is not possible to include the origin, and the grasp is not force closure. Therefore the algorithm terminates because the grasp quality metric is not defined. Figure 4.3 shows the initialization phase for the simpler $\mathbb{R}^2$ case. The following lemma is a special case of Lemma 3 because only a partial convex hull has been computed.

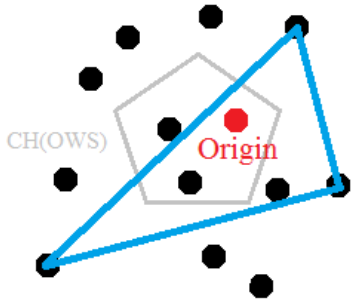



Figure 4.3: In the initialization stage, CH(OWS) is computed (gray) as well as a partial convex hull over the set of elementary wrenches (black dots). The process stops when the origin is included in the partial convex hull (cyan).

**Lemma 4.** *Let $\mathcal{V}$ be the set of vertices in* CH(OWS) *and $H$ be the set of hyperplanes supporting the facets of the current convex hull. If $\mathbf{v}_i \in V$ and $\mathcal{H}_j \in H$ achieve the minimum in the expression given by Eq.* (4.2) *over all hyperplanes in $H$, then $r_{i,j}\mathbf{v}_i$ lies on the boundary of the current convex hull, i.e., it lies in the facet supported by $\mathcal{H}_j$ and does not intersect with any other hyperplane.*

*Proof.* According to the definition of $r_{i,j}$ and elementary geometry, $r_{i,j}\mathbf{v}_i$ lies on $\mathcal{H}_j$ for every point $\mathbf{v}_i$ and every hyperplane $\mathcal{H}_j$. Let $\mathbf{v}'_i = r_{i,j}\mathbf{v}_i$ be such point, and by contradiction, assume $r_{i,j}\mathbf{v}_i$ intersects with another hyperplane $\mathcal{H}_k$ at point $\mathbf{v}'_i$. Then there must exist a positive $r'_{i,j} < r_{i,j}$ determined by the ratio between $\mathbf{v}'_i$ and $\mathbf{v}_i$. Then by definition $r'_{i,j}$ must be the inflation factor between vertex $\mathbf{v}_i$ and hyperplane $\mathcal{H}_k$, i.e., $r'_{i,j} = r_{i,k}$. However, this contradicts the hypothesis that $r_{i,j}$ achieves the minimum in Eq. (4.2) and the claim follows. □

Next, associations between vertices of OWS and facets of GWS are determined. Each vertex is associated with the facet with which it will intersect when inflated. The search for the correct face

to associate is done brute force over the set of all hyperplanes supporting the facets in the current partial convex hull. Lemma 4 assures that taking the smallest value computed for each vertex will identify the correct facet. Although this initial association is performed by trying all possible vertex-facet combinations, it is not time consuming because, at this stage, the convex hull only has a small number of facets (see figure 4.4). Note also, that since the association starts after the origin has been included in the partial GWS, each vertex in OWS is associated with a positive inflation factor.

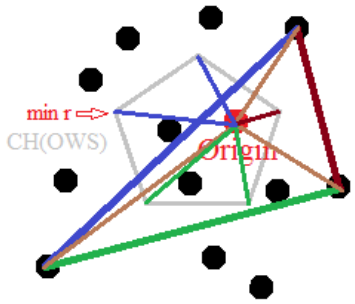



Figure 4.4: Preliminary association between vertices of OWS with facets of the partial GWS (associations are shown by colors). For some vertices (e.g., the blue ones) the actual inflation factor is smaller than 1 because they lie outside the partial GWS. The figure also shows the facet associated with the vertex with the smallest inflation factor.

In the main cycle, the algorithm iteratively expands the partial convex hull adding a new vertex for every iteration. As the objective is to determine the smallest inflation factor, the algorithm always expands the facet associated with the point with the smallest inflation factor. The expansion step follows the same heuristic of QuickHull and GWSPQH, i.e., the partial GWS is expanded by adding the farthest point in the outside set of the expanding facet (see figure 4.5). During this step the facet is replaced by a set of new facets. Additionally, point in OWS formerly associated with the removed facet becomes associated with one of the new facets. These points can only be associated with one of the new facets, because the directions these points represent remain unchanged and the old facets are replaced by new ones. Therefore, it is not necessary to search the whole set of facets in the partial convex hull.

This iterative expansion process continues until the facet associated with the vertex with the smallest positive inflation rate has an empty outside set. At that point the computation can be stopped because the smallest inflation rate has been determined and this value is the needed metric.

Algorithm 2 sketches the pseudocode for the strategy we just described. In line 1 the initial simplex CH is initialized as in the QuickHull algorithm. Next, (loop from line 3 to 15) CH is iteratively expanded until the origin is included. This condition is satisfied when the offset of the hyperplane supporting every facet is not negative, as per Lemma 1. If this is not the case, the simplex is expanded first by selecting from the facets with a non-empty outside set, then by selecting he one with the smallest offset (lines 11 and 15). This heuristic accelerates the process of including the origin. If the origin cannot be included in CH the metric $Q_{OWS}$ cannot be computed, and the

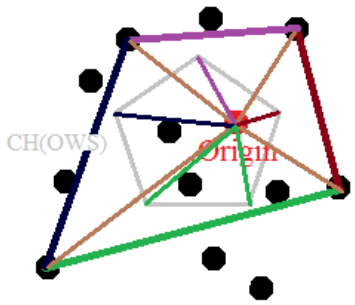



Figure 4.5: Expansion step. A facet in figure 4.4 (navy blue) is removed to expand the convex hull towards the farthest vertex in its outside set. Two new facets are created and the vertices of OWS formerly associated with the green face are now associated with the two new facets.

algorithm terminates (line 13). The initial assignment of the vertices of CH(OWS) to the facets is performed in lines 16-19 , by scanning through the set of all facets in the partial convex hull. For each facet $\mathcal{F}$, $r_{min}(\mathcal{F})$ is the smallest inflation factor among all vertices associated with the facet. Then, in the final loop (lines 20-27) the convex hull is iteratively expanded. At each iteration the facet with the smallest value for $r_{min}(\mathcal{F})$ is expanded, if possible, (line 21-23) and after the expansion the assignment of vertices to facets is updated. If the facet with the smallest value for $r_{min}(\mathcal{F})$ cannot be expanded, then the algorithm terminates and returns the corresponding inflation value (line 27).

### Non-force Closure Grasp Solution

The strategy used in algorithm 2 to deal with non-force closure grasps is simple rejection. This step is done in line 11 to 13 in the algorithm. The sufficient condition to identify a non-force closure grasp is any facet with negative offset that holds an empty outside set. This condition is simple to prove. Facets with an empty outside set implies it is one of the facets of the resultant convex hull. If the distance between a facet and the origin is negative, then according to Lemma 1, the resultant convex hull does not contain the origin, i.e., this grasp must be non-force closure. Since the condition is built for any facet with a negative offset, it is not necessary to obtain the actual distance from a facet to the origin, defined by $o(\mathcal{F}, 0)$. Compared to the solution we proposed in PQHGWS for non-force closure grasps, i.e., return the distance from the origin to the convex hull, the solution we proposed here, i.e., rejecting non-force closure grasps, is much more efficient.

### Computational Complexity

As for PQHGWS (and QuickHull), the computational complexity of PQHOWS cannot be easily determined, and therefore its complexity analysis has to rely on approximations and conjectures. Note that the following discussion does not include the complexity for the upfront computation of the convex hull of the object wrench space CH(OWS). This preliminary computation

**Algorithm 2** PQHOWS algorithm

1: Create intial simplex CH with 7 points
2: OriginInside ← **false**
3: **while not** OriginInside **do**
4: FCflag← 1
5: **for all** $\mathcal{F} \in$ CH **do**
6: **if** $o((\mathcal{F}, 0) < 0$ **then**
7: FCflag← 0
8: **if** FCflag=1 **then**
9: OriginInside ← **true**
10: **else**
11: $\mathcal{F}_e \leftarrow \arg\min_{\mathcal{F}} o(\mathcal{H}(\mathcal{F}), 0)$
12: **if** $O(\mathcal{F}_e) = \emptyset$ **then**
13: **return** "Grasp is not force closure"
14: **else**
15: Expand $\mathcal{F}_e$ and update CH
16: **for all** facets $\mathcal{F}_j \in$ CH **do**
17: **for all** $\mathbf{v}_i \in \mathcal{V}$(CH(OWS)) **do**
18: Compute $r_{i,j}$ as per Eq. 4.1.
19: Associate each $\mathbf{v}_i$ to the facet $\mathcal{F}_j$ with smallest $r_{i,j}$
20: **loop**
21: $\mathcal{F}_e \leftarrow \arg\min_{\mathcal{F}} r_{min}(\mathcal{F})$
22: **if** $O(\mathcal{F}_e) \neq \emptyset$ **then**
23: Expand $\mathcal{F}_e$
24: Reassign vertices associated with $\mathcal{F}_e$ to the new facets ($\mathcal{F}_{new}$)
25: Calculate $r_{min}$ for the new facets ($\mathcal{F}_{new}$)
26: **else**
27: **return** $r_{min}(\mathcal{F}_e)$

can be done with QuickHull. The complexity of PQHOWS (Algorithm 2) is determined by the sequential execution of three phases, each associated with a loop. The first (line 3 to 15) expands the initial simplex until the origin is inside and it is based on the QuickHull algorithm. Therefore it inherits its conjectured $O(n \log s)$ complexity, where $n$ is the number of vertices and $s$ is the number of processed vertices. The second loop (line 16 to 18) has complexity $O(V_{OWS} H_{CH})$ where $V_{OWS}$ is the number of vertices of CH(OWS) and $H_{CH}$ is the number of hyperplanes in the convex hull CH determined in the first loop. The third and last loop (line 20 to 27) continues to expand CH and at each iteration it reassigns some of the vertices in CH(OWS) associated with the facet in CH being expanded. Therefore its complexity is $O(V_{OWS} n \log s)$ where we again used the conjectured complexity for QuickHull. Therefore the conjectured complexity for PQHOWS can be stated as $O(V_{OWS} H_{CH} + V_{OWS} n \log s)$.

## 4.3 Experimental Evaluation

In this section we experimentally show that the two algorithms presented in section 4.1 and 4.2 provide significant computational improvements when compared with the state of the art.[3] Our code is obtained through modifying the freely available QuickHull implementation and, therefore, benefits from a solid code base. All tests were run on a standard desktop running Linux with a 2.8GHz Intel i7 processor and 8 Gb RAM.

### 4.3.1 Grasp Wrench Space Metric

QuickHull is the algorithm most commonly used to compute the GWS based grasp quality metric. We compared PQHGWS with QuickHull over a set of 200 grasps with 4 contact points and an approximation of the friction cone with 32 edges. The object being grasped is a bar. To put the following results into perspective, it is worth recalling that PQHGWS does not approximate the value of the grasp quality metric, but rather returns exactly the same value obtained using QuickHull. Since the grasps are randomly generated, some of them are force closure and some are not. Figure 4.6 shows the results.

We contrast the two algorithms using three performance measures, namely the number of processed points, the number of generated hyperplanes, and the time spent to compute the metric. Note that the number of points and the number of hyperplanes are the measures used in literature to assess the scalability of QuickHull [5]. The three plots on the left contrast the absolute performance, with the blue line showing QuickHull's performance and the the green line showing PQHGWS' performance. Since the gap between the two is large, the three plots on the right display the ratio between them (the larger the better). It is interesting to note that the average speedup for the number of processed points is only about 6, whereas the average speedup in terms of number of processed hyperplanes exceeds 70 and the average speedup in terms of computational time is slightly below 40. With regard to this last number, it shall be outlined that our current implementation is not optimized, and therefore this gain could be further improved.

In practical scenarios the friction cone is approximated using a regular pyramid. Hence to reduce the approximation error one would increase the number of edges in the pyramid. There is a linear relationship between the number of edges and the number of points for which the convex hull will be computed. Figure 4.7 shows the ratio between QuickHull's and PQHGWS' performance as a function of the number of edges approximating the friction cone. The figure shows that as the number of edges $k$ increases the performance gap between QuickHull and PQHGWS grows as well. Therefore, PQHGWS allows us to approximate the friction cone with a pyramid with a large number of edges without introducing a significant computational burden. This would not be possible with QuickHull. Figures 4.7 is particularly important in light of the findings presented in [75]. Pokorny and Kragic have shown that when the grasp is force closure the approximation error introduced by discretizing the friction cone with $k$ edges is $M(1 - \cos(\pi/k))$, where $M$ is a constant accounting for geometric aspects unrelated to $k$. Using PQHGWS, one can afford to use large $k$ values to approximate the cone. For example, for $k = 64$ PQHGWS is still extremely fast

[3]The code implementing the algorithms described in this paper is freely available on the authors' website, together with the datasets used to generate the results presented in this section.

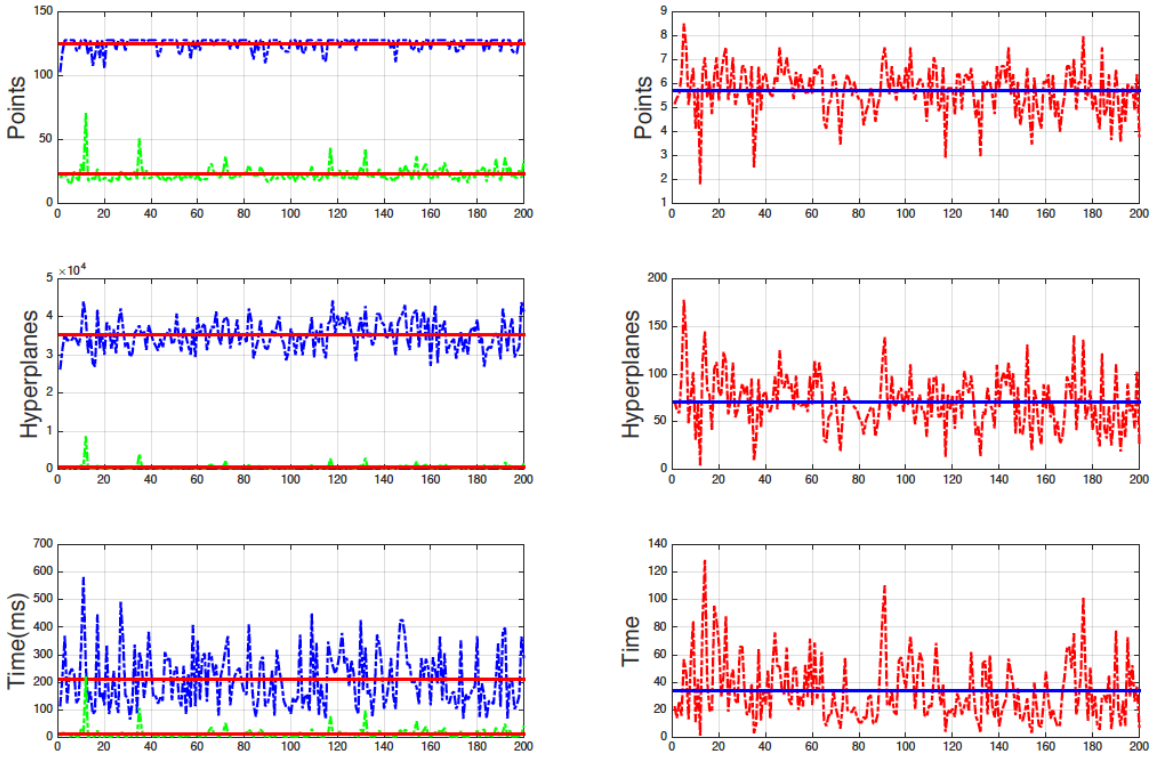



Figure 4.6: Left: Performance of QuickHull (blue) and PQH (green) for randomly generated grasps that may or may not be force closure. Right: Ratio between the two performances. In all figures the horizontal colored line indicates the average.

and $(1 - \cos(\pi/k)) \approx 0.0012$.

### 4.3.2 Object Wrench Space Metric

We next evaluate the performance of the PQHOWS algorithm. In this case there is no widely used implementation available for comparison, so we contrast our method with two alternatives. The first considers the brute force approach obtained by applying Eq. (4.1) for every vertex $\mathbf{v}_i$ and every facet $\mathcal{F}_j$. This approach is slow but computes the correct result without introducing any approximation. The second method was proposed in [14] and is approximate. The idea is to enclose the OWS with the smallest ellipsoid, and to then transform the ellipsoid into a sphere with a linear transformation applied to both OWS and GWS. After this transformation the metric is obtained by computing the radius of the largest sphere enclosed in the transformed GWS. This computation can be then performed using QuickHull or PQHGWS. Note that besides being approximated, this method also does not provide an explicit bound on the approximation error.

Figure 4.8 shows a time comparison of the various methods over 200 different grasps for the same OWS. Every method requires the preliminary computation of OWS, so this time is excluded from the chart because it is determined upfront for all algorithms. We consider two com-

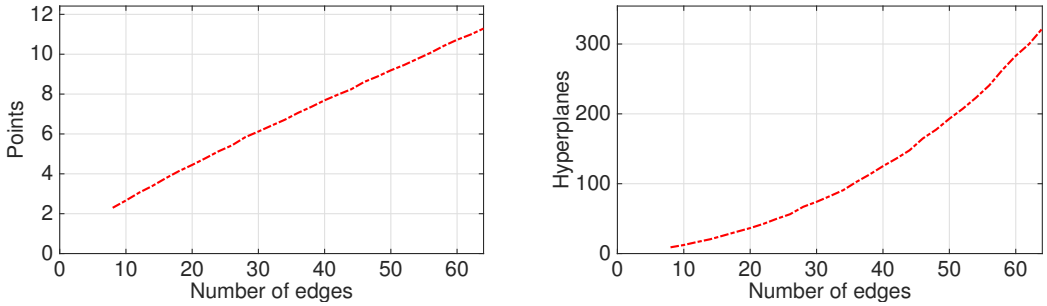



Figure 4.7: Ratio between the performance of QuickHull and PQH as a function of the number of edges used to approximate the friction cone.

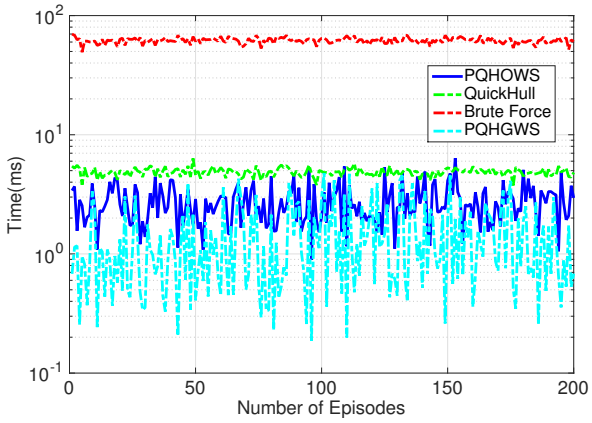



Figure 4.8: Time comparison for different methods computing the metric $Q_{\text{OWS}}$. Note the logarithmic scale on the $y$ axis.

parisons. First, we compare the two methods computing the exact value for the OWS metric, i.e., the brute force method (red line) and PQHOWS (blue line). The chart clearly shows that PQHOWS largely outperforms the brute force method. Next, we compare PQHOWS with two different implementations for the approximate method proposed in [14]. The two implementations differ in how they compute the convex hull, i.e., the first uses QuickHull (green line), whereas the second uses PQHGWS (cyan line). The chart shows that the three methods are more or less comparable in terms of time. However, the main difference is that PQHOWS computes the exact result, while the other two methods provide only an approximation, and the next experiment shows that the approximation error can be, at times, significant.

Figure 4.9 shows the relative approximation error, that is for each of the 200 grasps evaluated we plot $\frac{|Q_{\text{OWS}} - \tilde{Q}_{\text{OWS}}|}{Q_{\text{OWS}}}$ where $Q_{\text{OWS}}$ is the exact value and $\tilde{Q}_{\text{OWS}}$ is the approximate value computed with [14]. The figure shows that the approximation error can be at times very large and, therefore, the method in [14], albeit slightly faster, should be avoided. One should add that to the best of our knowledge the quality of the approximation proposed in [14] has not been analytically studied,

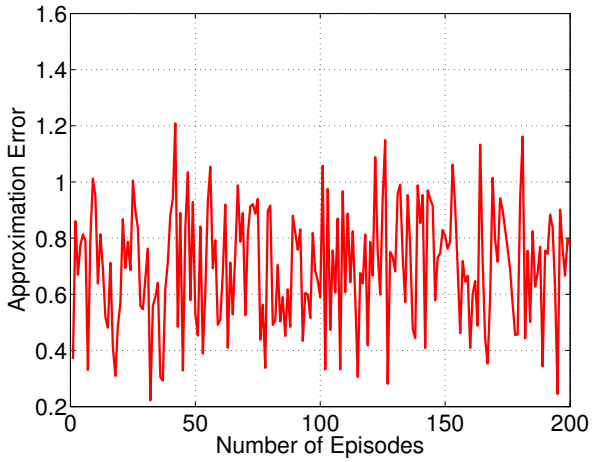



Figure 4.9: Approximation error introduced by the method proposed in [14].

and, therefore, an experimental evaluation is the best one can aim for at the moment. The analysis demonstrates that the method we propose is preferred because it provides the exact result and its computational requirements are comparable with the faster of the implementations for the approximate methods. Finally it is important to observe that a direct comparison of PQHOWS and PQHGWS is ill posed because the two algorithms compute two different metrics. For a given grasp PQHGWS is, in general, faster, but the metric it computes is different (and in a sense less desirable or accurate) than the one computed by PQHOWS.

One factor affecting the time complexity of PQHOWS is $\mathcal{V}$(CH(OWS)), i.e., the number of vertices of the convex hull of OWS. Table 4.1 shows the relation between $\mathcal{V}$(CH(OWS)) and the average run time of PQHOWS for four different objects. Results for averages over 1000 runs. Referring to the last column of table 4.1, the ratio between the average planning time of PQHOWS and $\mathcal{V}$(CH(OWS)) is nearly constant. This result suggests that the time complexity of PQHOWS is linear over $\mathcal{V}$(CH(OWS)). Another meaningful result we can obtain from table 4.1 is the computation time comparison of PQHOWS, QuickHull and PQHGWS. For objects with relatively small $\mathcal{V}$(CH(OWS)), such as the cube, the performance of PQHOWS is closer to PQHGWS. For more complicated objects with large $\mathcal{V}$(CH(OWS)), our algorithm is worse, but still comparable to the performance of QuickHull. Note that the calculation of $OWS$ is not shown in the table. Since $OWS$ is fixed for all grasp candidates, it is precalculated upfront.

$\mathcal{V}$(CH(OWS)) is affected by the construction of OWS. OWS is calculated by applying unit disturbance force along all directions on the surface of the object. The term *any directions* introduces an approximation into the calculation of OWS, which is determined by a sample of a uniform unit sphere. Figure 4.10 shows the result of four important factors with respect to the change of the number of vectors used to approximate the unit sphere. All cases refer to the joystick object. The number of vectors we tested are $11^2$, $16^2$, $21^2$, $26^2$, $31^2$, and $36^2$. The top left figure shows the average quality value of $Q_{ows}$ over the same 1000 samples. The change in the average value from the lowest sphere approximation number to the highest sphere approximation number is lower than 0.0006, which is approximately 2%. We also calculated the difference in standard deviation shown

| | | Average Computation Time | | | |
|---|---|---|---|---|---|
| Object | $\mathcal{V}$(CH(OWS)) | PQHOWS | QuickHull | PQHGWS | $\frac{AvgT(PQHOWS)}{\mathcal{V}(\text{CH(OWS)})}$ |
| Cube | 1458 | 3.345 | 13.876 | 0.977 | 0.0023 |
| Sphere | 7809 | 16.076 | 6.610 | 1.154 | 0.0021 |
| Joystick | 5293 | 9.673 | 7.859 | 0.971 | 0.0018 |
| Teapot | 5704 | 10.938 | 7.043 | 0.999 | 0.0019 |

Table 4.1: For four different test cases, the table displays the number of vertices in $\mathcal{V}$(CH(OWS)) and the average computation time using PQHOWS, QuickHull and PQHGWS. The last column displays the ratio between the average planning time of PQHOWS and $\mathcal{V}$(CH(OWS)). Time is expressed in seconds.

in the bottom left of the figure. This value is calculated by using $Q_{OWS}$, calculated for each case, and subtract the lowest sphere approximation case. Then we use this set of values, which for each case contains 1000 values, to calculate its standard deviation. As shown in this subfigure, the difference in standard deviation is in the order of $10^{-4}$. Additionally, consider the two subfigures on the right, displaying that the average time used to calculate QOWS and the number of vertices of the convex hull of the OWS, shows a linear growth with respect to the number of vectors used to approximate the unit sphere. Combining the results shown in figure 4.10, we can conclude that it is not necessary to approximate the unit sphere with a very high value. A very accurate approximation does not gain much with respect to the quality, but will be a huge burden on run time. An appropriate number to approximate the unit sphere would be in the range of $16^2$ to $21^2$. The approximation number we used for all previous experiments regrading to OWS are with 441 vectors, which ensures a rather accurate approximation. The performance of PQHOWS can be improved by decreasing the approximation number to around 200, and the time to calculate $Q_{OWS}$ can be cut in half.

### 4.3.3 Impact on Planning

While expediting grasp quality evaluation is of independent interest, efficient implementations are particularly valuable to improve the performance of grasp planners relying on grasp metrics to search the space of possible grasps. In this section we substantiate this claim considering two different planners and showing how they are affected by the choice of the algorithms used for grasp evaluation. First, for qualitative purposes consider figure 4.11, where we display how the choice of the grasp metric is significant. For the object being grasped, 1000 force closure grasps were determined upfront. The grasp wrench space metric and the object wrench space metrics were then used to select the best one among them. The top three figures show the best grasp determined using the first metric and the bottom three figures the best grasp obtained using the second metric.

The difference is evident, and similar discrepancies were observed in all the experiments we performed, outlining that the choice of the metric may have a dramatic impact on the grasp selected by a planner. Thus it is important to integrate more than one quality evaluation algorithm into the planner. To put this statement into perspective, it is important to recall that, to date, the object wrench space metric is rarely used in practice because of its computational complexity. As we

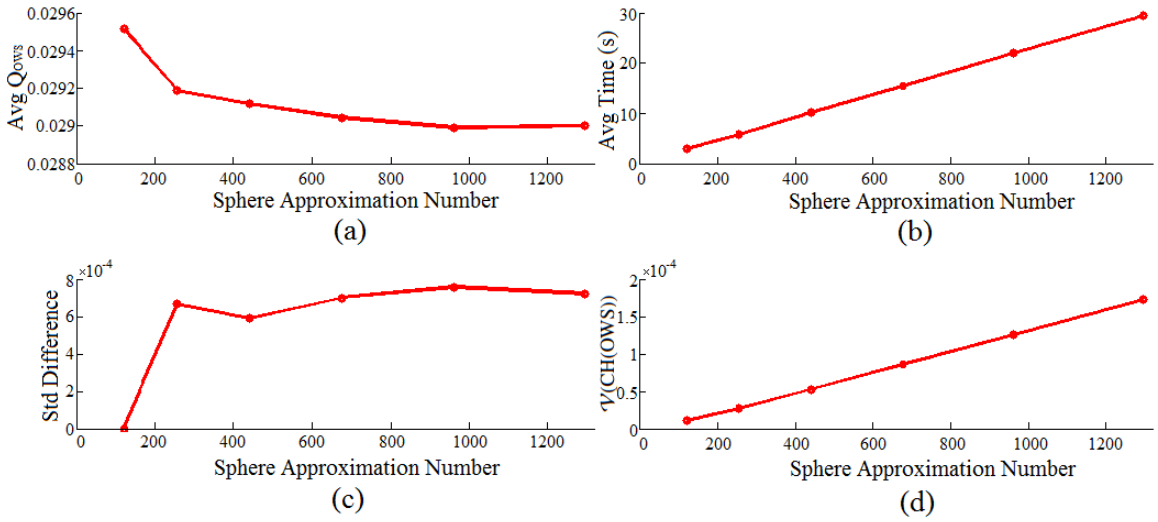



Figure 4.10: (a) shows the effect of the sample number on the unit sphere used to construct OWS with respect to average $Q_{OWS}$ calculated through 1000 samples. (b) shows the average time used to evaluate $Q_{OWS}$ for each case. (c) shows the standard deviation of the difference between all cases and the first case where the unit sphere is sampled with 121 vectors. (d) shows the number of vertices of the convex hull of the OWS with respect to the sample number of the unit sphere used to construct the OWS.

will show next, PQHOWS provides an algorithm whose performance is comparable to the current implementations for the grasp wrench space metric, and thus enables the use of this more powerful tool for grasp planning.

We next consider two planners to determine force closure grasps on some objects, three of which are displayed in Figure 4.12. The other two objects we consider (not displayed in the figure) are a sphere and a cube.

The first planner we consider works like the GraspIt! planner [66], i.e., it places the hand at a random position in proximity of the object and then closes the fingers to determine if the resulting contact points produce a force closure grasp or not. With this algorithm we determined 1000 grasps and we evaluated each of them using both the grasp wrench space metric and the object wrench space metric. For both algorithms we used the current implementation (QuickHull, BruteForce) and our algorithms PQHGWS and PQHOWS to compare the time. Figure 4.13 shows the results (displayed time is averaged over 1000 grasps, and the top figures display standard deviations) Figures 4.13.a and 4.13.b confirm that PQHGWS and PQHOWS largely outperform their counterparts. Figure 4.13.c compares the performance of PQHOWS with the algorithm computing the grasp wrench space metric with QuickHull. The figure shows that PQHOWS is comparable[4] to QuickHull in terms of the time spent to evaluate a grasp, and it can therefore be readily integrated into existing planners without altering their performance while providing a significantly better grasp

[4]The reader should consider that our PQHOWS implementation is not optimized yet, so small differences in performance can be eliminated.

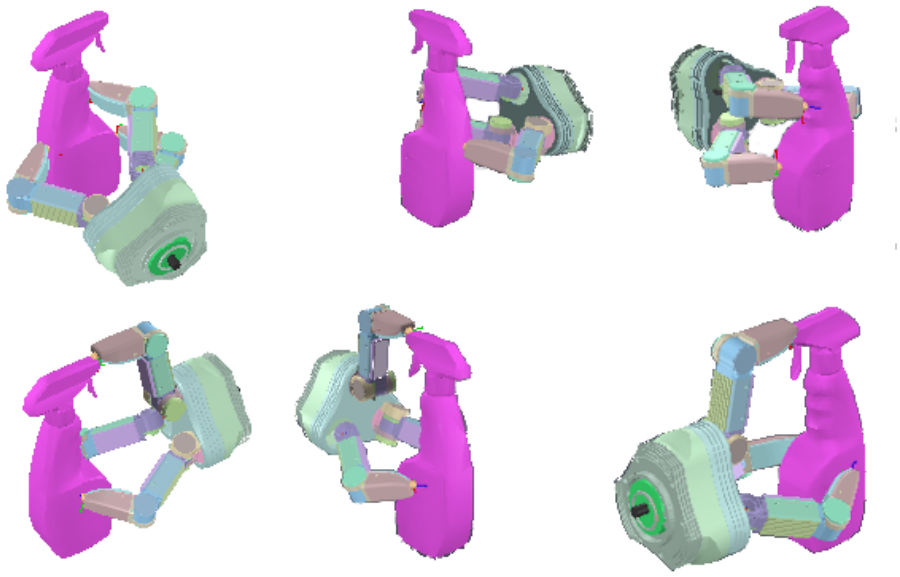


Figure 4.11: Top figures: best grasp determined using the grasp wrench space quality measure. Bottom figures: best grasp determined using the object wrench space quality measure. Each grasp is shown from three different view points to outline the differences.

quality measure. For completeness Figure 4.13.d also compares the the performance of PQHGWS with PQHOWS. To put this chart into perspective, it is important to recall that the two algorithms compute two different metrics, with the object wrench space metric being more complex. Therefore it is not surprising that PQHGWS outperforms PQHOWS.

Next, we incorporated PQHGWS into a grasp planner we recently developed [58]. In essence the algorithm performs a search for the best grasp exploring the surface of the object and guiding its search using a gradient-descent like approach informed by the quality metric (the details of the planner are provided in chapter 7). Our original implementation relied on QuickHull to compute $Q_{GWS}$, and to perform this comparison we replaced it with PQHGWS. Table 4.2 illustrates the results obtained when the planner seeks a grasp with four contact points and the friction cone is approximated with $k = 24$ edges. Each test case refers to a different object, and averages over 100 different planned grasps are displayed.

The second and third column show the overall planning time spent when QuickHull or PQHGWS are used, respectively. The fourth column shows the ratio between the two. Similarly, the fifth and sixth columns show only the time spent for grasp quality evaluation and the seventh column displays their ratio. As expected, this second ratio is higher since it considers only the time spent for grasp quality evaluation, whereas in the previous case, the overall planning time is shown. The table shows that by just substituting QuickHull with PQHGWS, speedups larger than 20 can be immediately obtained.

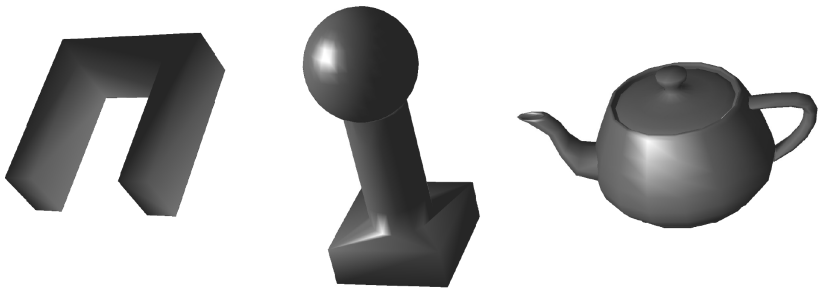


Figure 4.12: Three of the objects used to assess the impact of PQHGWS and PQHOWS on grasp planning. Left to right: C-shape, joystick and teapot.

| | Planning Time | | | Grasp Quality Evaluation | | |
|---|---|---|---|---|---|---|
| Testcase | QuickHull | PQHGWS | Speedup | QuickHull | PQHGWS | Speedup |
| Cube | 219.713 | 10.326 | 21.28 | 218.034 | 8.647 | 25.26 |
| Sphere | 163.625 | 7.678 | 21.31 | 162.182 | 6.235 | 26.01 |
| C shape | 270.357 | 15.921 | 16.98 | 268.439 | 14.003 | 19.17 |
| Joystick | 309.247 | 13.895 | 22.26 | 306.965 | 11.613 | 26.43 |
| Teapot | 309.629 | 10.376 | 29.86 | 307.204 | 7.951 | 38.64 |

Table 4.2: For five different test cases, the table displays the planning time spent when QuickHull or PQHGWS are used. The last three columns display just the time for grasp quality evaluation. Time is expressed in seconds.

## 4.4 Conclusions

In this chapter we have presented two algorithms to efficiently compute two grasp quality evaluation metrics proposed in literature. Both algorithms build upon the common intuition that the metrics can be obtained computing a partial quick hull in the six-dimensional space of wrenches. The first algorithm, PQHGWS, computes the widely used metric proposed by Ferrari and Canny more than twenty years ago. It is based on a modified criterion for expansion for the QuickHull algorithm and we have experimentally shown that is substantially faster than the original. The second algorithm considers a metric based on the object wrench space that was proposed in the past but rarely used in practice. This metric is defined only for force closure grasps, and we demonstrated that our method yields a performance comparable to the one obtained for the Ferrari and Canny metric. Therefore, thanks to our algorithm this metric can now be used in practice, whereas, in the past, it was deemed too demanding for practical use. Implementations for both algorithms are freely available.

In this chapter, we also provided two methods to deal with non-force closure grasps. The first one computes the distance of the convex hull to the origin for the case where the origin is not inside the convex hull. This method is beneficial for grasp planners using an optimization approach, such that the distance of the convex hull to the origin can be used to greedily drive the planner from non-force closure grasps towards force closure grasps. Although the current grasp measured is

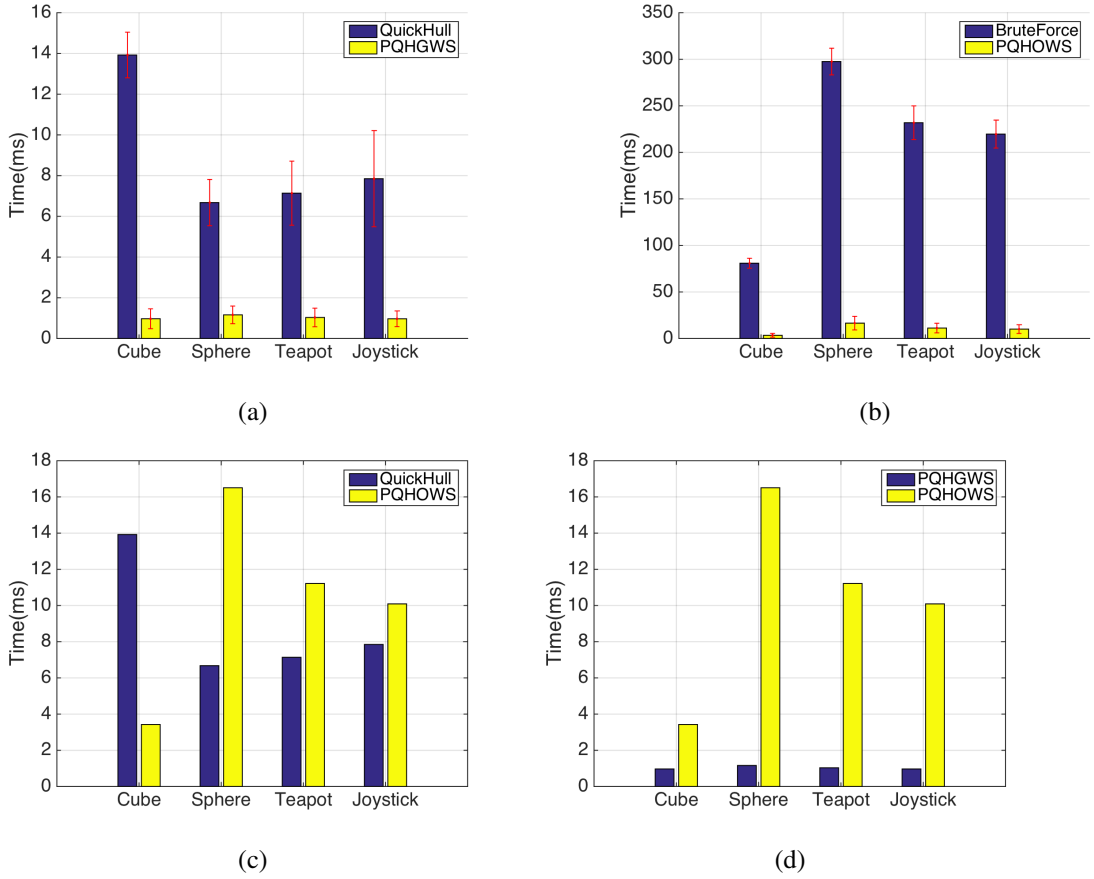



Figure 4.13: Performance comparison between different grasp quality evaluation algorithms.

not force closure, by approaching the direction that returns a closer distance in the next step drives the planner towards a local optimum grasp. The other method we proposed to deal with non-force closure grasps is by quickly rejecting them. This method fits well in sampling based grasp planners, where the current measure would not affect the grasp selected in the next step. Since the sampling based planner samples randomly, quickly rejecting non-force closure grasps improves the overall performance. Both methods are affected by the growth of the convex hull, which can be easily combined with the partial quick hull algorithms for GWS and OWS.

In this chapter we have focused on the grasp wrench space and the object wrench space, but a third paradigm was proposed in literature, namely the task wrench space. We have not considered the TWS metric because to date there is no efficient and practical way to determine the TWS in a general setting. However, it should be noted that the algorithm presented in section 4.2 could be equally applicable if the TWS is provided. In fact, one can just substitute OWS with TWS in algorithm 2 without making further changes.

# Chapter 5

# Grasp Quality Evaluation with Whole Arm Kinematic Noise Propagation

The study of grasp quality metrics is deeply intertwined with the study of grasp planning algorithms. This connection stems from the fact that in most cases a robotic manipulator can grasp a given object in multiple ways. Given a set of possible solutions determined by the grasp planner, it is then natural to ask which one should be preferred, and the utility of grasp quality metrics becomes evident. While many grasp quality metrics have been proposed [29, 55, 76, 95], most contributions have ignored the morphology of the robot executing the grasp. For example, the Ferrari and Canny metric [29] just considers the contact points without incorporating kinematic constraints. Most other methods embrace a similar standpoint [55, 76, 95]. Moreover, most classical grasp quality metrics do not incorporate a fully realistic noise-model, accounting for the inevitable inaccuracies emerging when a grasp is executed. This problem is particularly relevant with the advent of platforms with passive joints (e.g., Baxter [34]), for which small deviations from the desired trajectory or desired final position and orientation of the end effector are unavoidable.

Noise modeling in mechanics has been well studied, e.g., in [72], where it is evidenced how noise in modeling robot arm mechanics derives from sources such as inaccuracies in the geometric models of links, backlash, nondeterministic errors due to friction, and quantization errors. Although many papers account for these noise sources through linearized error propagation models and Gaussian distributions, works like [72] model errors using the empirical error distribution. Along with error modeling, a rich literature exists in parameter identification, using either analytic approaches [36] or data-driven techniques [8, 97].

Noise modeling for grasping has rarely been considered together with the kinematic structure. For example, [51] accounted for Gaussian errors in the end-effector positions, friction coefficient and object shape, and formulated the problem using a probabilistic framework. Similarly, Allen and collaborators [106] investigated uncertainty in the object model, which led to the notion of *probabilistic force closure*. Furthermore, [63] applied *probabilistic force closure* and proposed a large-scale cloud-based approach for sampling perturbations of grasps, leveraging multi-armed bandits and deep learning to determine grasps with high probability of force closure. The approach we consider in this chapter is instead focused on the impact of noise on the grasp quality metric through the whole-arm kinematic structure.

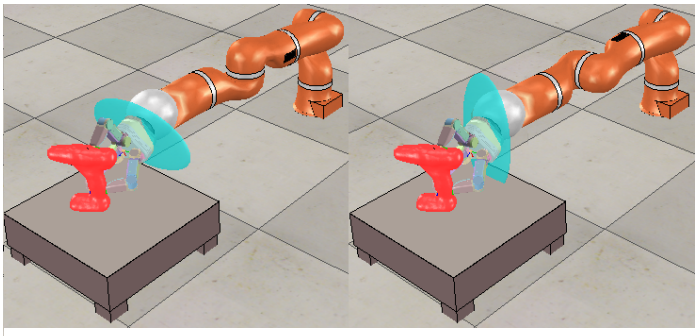



Figure 5.1: The above figures display two grasps with identical contact point configurations. Note however the relative rotation of the third joint by $\frac{\pi}{2}$. When imposing Gaussian Noise on the individual joints of the robot arm, the covariance ellipsoid of end-effector positions is also rotated by $\frac{\pi}{2}$, resulting in a drastically changed value for $P(FC)$.

In this chapter, we propose to explicitly account for noise in joint-angle positions, and to study the relationship between such noise and the definition of the grasp quality metric $Q$ defined by Ferrari and Canny in [29] and widely used in literature. We focus in particular on the probability of force closure measure $P(FC)$ [106] and the expected grasp quality $\underline{\mathrm{E}}[Q]$. The reader is referred to Figure 5.1 to consider the motivations for this work. It shows an example of two grasps with identical contact configuration at the finger tips, but with a differing third joint angle. With a classical modelling of these grasps, without noise, or assuming independent and identically distributed noise in end-effector position, these configurations would be considered equivalent. However, we observe that by considering Gaussian perturbations in joint-angles, significant differences between these configurations are noted. For example, our experiments show that the grasp shown on the left has a significantly lower probability of force closure and expected Ferrari and Canny grasp quality compared to the grasp on the right. The same results are observed throughout the numerous experiments we present in this manuscript. We argue that noise-models that fully incorporate the kinematic structure and limitations of a robot should be used to more reliably predict the success of a proposed grasp. The main goal of the present chapter is to highlight the importance of noise in joint-configurations for the purpose of grasp quality evaluation and synthesis. While error propagation is a well-studied problem in mechanics and robot arms [27], the impact of errors in joint-angles has not been integrated with the main-stream robotic grasp quality evaluation literature. We in particular present a sampling based approach to study the impact of noise on the Ferrari and Canny grasp quality metric and on the probability of force closure.

The rest of this chapter is organized as follows. In section 5.1 we define the problem we consider and present the experimental methodology sustaining our study. Experiments and their results are illustrated in section 5.2, while conclusions and future work are discussed in section 5.3.

## 5.1 Problem Definition and Methodology

### Problem Definition

Consider a robot arm with $d$ degrees of freedom equipped with a multifingered robotic hand, like the one shown in Figure 5.1. The forward kinematics (FK) function, $f : R^d \rightarrow SE(3)$, maps a joint-configuration $\mathbf{q} = (q_1, \ldots, q_d) \in R^d$ to the position and orientation of a frame rigidly attached to the end effector (e.g., to the center of the palm of the hand.) An approach widely used in grasp planning (see e.g. [67]) determines the pose of the reference point, which then projects the contact points for the fingers, assuming they are closing until contact is made with the object being grasped. Knowing the mechanical structure of the hand and of the object being grasped, this projection is a straightforward computation. Hence in the following, we can treat $\mathbf{f} = f(\mathbf{q})$ as a grasp. The quality $Q$ of the resulting grasp can then be calculated using one of the aforementioned metrics. In this work we assume the joint angles of the arm with $d$ DOF, are set to a desired target value, $\mathbf{q_0}$, but superimposed noise exists at each joint. Specifically, noise is modeled as vector $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_d)$, where the $\varepsilon_i$s are independent random variables with known distributions. For a realistic setup accounting for the mechanical limitations governing each joint, we consider each $|\varepsilon_i|$ to to be bounded by $\varepsilon_{max}$. Moreover, we assume each random component to have 0 mean. Distributions like the the truncated Gaussian[1] and the uniform distribution are obvious candidates to model this type of noise.

For the given setup, instead of reaching the target configuration $\mathbf{q_0}$, the arm will end up at $\mathbf{q} = \mathbf{q_0} + \varepsilon$. As a result, the grasp quality $Q$ is a random variable. We are therefore interested in the dependency between the random variable $Q$, the noise vector $\varepsilon$, and the kinematics of the robot arm. To be specific, for the grasp quality metric we consider the well known Ferrari and Canny metric that measures the size of the largest wrench along all directions that can be resisted by the grasp. For the case where the fingers fail to make contact with the object, or if the grasp does not achieve force closure, the value of the metric is undefined. To assess the effect of noise on $Q$, we consider the formerly mentioned probability of force closure $P(FC)$ and the expected quality metric $\underline{\mathrm{E}}[Q]$. To study the properties of $Q$ as a random variable, we consider different noise distributions and arm configurations achieving the same end-effector pose. The goal of our analysis is to identify variations in grasp robustness caused by variations in arm configurations, and reason about how we can computationally select the best arm configuration to achieve a higher $P(FC)$ for a target end-effector pose.

### Sampling based approximation

Given a target grasp configuration, $\mathbf{q_0}$, determined, for example, with a grasp planner such as GraspIT!, we study the resulting grasp quality of the random variable $Q(f(\mathbf{q_0} + \varepsilon))$, where we explicitly outline the dependency of $Q$ on both the forward kinematics map $f$ and the noise $\varepsilon$. To this end, we generate a finite number of samples $\varepsilon^1, \ldots, \varepsilon^n \in R^d$ and let $X_i = Q(f(\mathbf{q_0} + \varepsilon^{\mathbf{i}}))$. We then

[1] This is a modified Gaussian distribution with 0 mean and whose density function is set to zero outside $[-\varepsilon_{max}, \varepsilon_{max}]$ and then normalized to integrate to 1. We use $\mathcal{N}(0, \sigma^2, \varepsilon_{max})$ to indicate this distribution.

compute the $n$-sample empirical estimators (mean, standard deviation, co-variance, skewness):

$$\underline{\mathrm{E}}[Q, \mathbf{q}_0] \simeq \frac{1}{n}\sum_{i=1}^{n} X_i = \overline{Q}$$

$$\sigma \simeq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{Q})^2}$$

$$Cov[Q, \mathbf{q}_0]_{s,t} \simeq \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{Q})_s(X_i - \overline{Q})_t$$

$$\underline{\mathrm{M}}_3[Q, \mathbf{q}_0] \simeq \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - \overline{Q}}{\sigma}\right).^3$$

## Analytical approximation

As mentioned earlier, FK computes the pose of the end-effector as a function of the joint angles using the kinematic equations of a robot. For a given arm configuration $\mathbf{q_0}$, the corresponding end-effector pose can then be written as $\mathbf{p}_e = f(\mathbf{q_0})$. When noise is added in each joint, the end-effector pose is instead $\mathbf{p}'_e = f(\mathbf{q_0} + \varepsilon)$. The difference between the actual pose of the end effector and the desired pose can be written as

$$\Delta\mathbf{p}_e = f(\mathbf{q_0} + \varepsilon) - f(\mathbf{q_0}).$$

This mismatch is due to the error $\varepsilon$, superimposed to the desired robot configuration, $\mathbf{q_0}$. Let $\mathbf{J}$ be the Jacobian of the forward kinematics function $f$. For small values of $\varepsilon$, the error can be approximated as

$$\Delta\mathbf{p}_e \approx \mathbf{J}(\mathbf{q_0})\varepsilon.$$

According to our assumptions, each component of the disturbance vector satisfies the inequality $|\varepsilon_i| \leq \varepsilon_{max}$. If $\mathbf{q_0}$ is a non-singular configuration we can then write:

$$\Delta\mathbf{p}_e^T(\mathbf{J}\mathbf{J}^T)^{-1}\Delta\mathbf{p}_e \leq d\|\varepsilon_{max}\|_2^2$$

This inequality represents an ellipsoid for the error distribution of the end-effector pose with bounded noise $\varepsilon$. We dub this ellipsoid the *noise ellipsoid*. The length of the axes and the orientation of the noise ellipsoid are determined by the eigenvalues and eigenvectors of matrix $\mathbf{J}\mathbf{J}^T$. The noise ellipsoid is however accurate only for noise vectors $\varepsilon$ with a small norm, due to the first order approximation we used.

Though the approximation of the noise ellipsoid does not provide a tight bound, we can still use it to formulate the problem of computing an arm configuration to grasp an object that is robust to configuration disturbances. That is to say that among the various possible solutions provided by inverse kinematics in the case of a redundant manipulator, we can use robustness to noise as a selection criterion. To formalize the problem, we introduce the concept of *high probability force closure region* as follows.

**Definition 11.** *Let* $\mathbf{f}_e \in SE(3)$ *be an end-effector pose that achieves a force closure grasp. The* high probability force closure region *associated to* $f_e$ *with confidence* $\zeta$ *HPFCR(*$\mathbf{f}_e, \zeta$*) is defined as the neighborhood of* $\mathbf{f}_e$ *such that*

$$\forall \mathbf{f} \in HPFCR(\mathbf{f}_e, \zeta) \quad P_{FC}(\mathbf{f}) > \zeta$$

*where* $P_{FC}(\mathbf{f})$ *is the probability that* $\mathbf{f}$ *achieves force closure.*

Starting from this definition, we present a sampling based approach to approximate the high probability for closure region. The steps are as follows.

1. Determine a force closure grasp with grasp quality value $Q_{\mathbf{f}_e} > Q_{MIN}$, where $Q_{MIN}$ is a predetermined constant lower bound.

2. Given a bound $b$ and step size $s$, sample the grasps around $\mathbf{f}_e$ within the cubical region with edge size $2b$ using step size $s$. Let $P_{ALL}$ be the set of all such grasps.

3. Evaluate grasps in $P_{ALL}$ and let $P \subset P_{ALL}$ be the set of grasps that are force closure with quality larger than a threshold $t_q$.

4. Calculate the six-dimensional ellipsoid $E$ fitting $P$ with confidence percentage $c$. The confidence percentage refers to the percentage of points in set $P$ that are within $E$. In the following we set $c = 70\%$, but the algorithm is not too sensitive to this value.

5. Compute $P(E)$, i.e., the probability of force closure of $E$ as the ratio between the number of grasps in $E \cap P_{ALL}$ that are force closure and the total number of grasps in $E \cap P_{ALL}$. If $P(E) < \zeta$, increase $t_q$ and go back to step 3. Otherwise, we terminate and set $E$ to be HPFCR($\mathbf{f}_e, \zeta$). Note that the center of $E$ is usually shifted from the origin. In order to improve the probability of force closure, we use the center of $E$ to correct the force closure grasp we used in the first step.

Step 2 in the above procedure is the most critical because the end-effector pose is a six-dimensional vector and therefore brute force enumeration generates about $(\frac{2b}{s})^6$ grasps that must be evaluated. This problem, however, is mitigated in two different ways. First, this entire process is done offline in a precomputation step. Second, rather than enumerating all possible grasps in the regularly spaced grid, it is possible to generate just a subset of fixed size. Both of these expedients are explained and illustrated in the following.

After obtaining the high probability force closure region, the next step is to use it to measure the quality of an arm configuration, i.e., the ability of an arm configuration to yield a grasp with high probability force closure despite joint-level noise. The following definition formalizes this idea.

**Definition 12.** *Let* $E = HPFCR(\mathbf{f}_e, \zeta)$ *be the high probability force closure region for a grasp* $\mathbf{f}_e$ *and confidence* $\zeta$*. By construction, E is a six-dimensional ellipsoid, and let* $\mathbf{E}$ *be the associated square matrix representing it. For a given arm configuration* $\mathbf{q}_a$ *such that* $\mathbf{f}_e = f(\mathbf{q}_a)$*, let* $\mathbf{J}_a$ *be the Jacobian*

*matrix, and let* $e_1, \ldots, e_6$ *be the eigenvalues of the matrix* $d\varepsilon^2 \mathbf{E}^{-1}(\mathbf{J}_a \mathbf{J}_a^T)$. *The grasp quality with respect to arm configuration* $\mathbf{q}_a$ *is*

$$Q_{arm} = \frac{1}{\Sigma_i(\sqrt{e_i} - 1)^2}.$$

The rationale for this definition is as follows. $\sqrt{e_i}$ is the axis length of the ellipsoid determined by $d\varepsilon^2_{max}\mathbf{E}^{-1}(\mathbf{J}_a\mathbf{J}_a^T)$. Therefore $(\sqrt{e_i} - 1)^2$ measures the mismatch against the unit sphere along the $i-th$ dimension. The matrix $J_a J_a^T$ represents the noise ellipsoid at the end-effector of the given arm configuration and linear transformation $E^{-1}$ maps this matrix to a space where the high probability force closure region is a unit sphere. Therefore, $Q_{arm}$ measures the error between the ellipsoid determined by $d\varepsilon^2_{max}E^{-1}(J_a J_a^T)$ and the unit sphere. Defined this way, $Q_{arm}$ indicates how well the noise ellipsoid and the high probability force closure region are aligned, with higher values representing more robustness to noise. Since, for a given grasp, only $\mathbf{E}^{-1}$ is needed, we thereby store this information along with the grasp after the process of calculating the high probability force closure region.

The overall pipeline for our entire system is shown in Figure 5.2, and is divided into an offline and an online stage. For a given object, during the offline process, grasps with good quality are calculated along with their associated high probability force closure region. Each grasp is stored with its corresponding $\mathbf{E}^{-1}$ matrix in a database for online use. During the online process, the relative pose between the arm and the object to be grasped is determined, usually through by a vision pipeline as we did in [61]. Next, using IKFAST we obtain a set of candidate arm configurations, and determine which one to use after ranking them using the $Q_{arm}$ metric we just introduced.

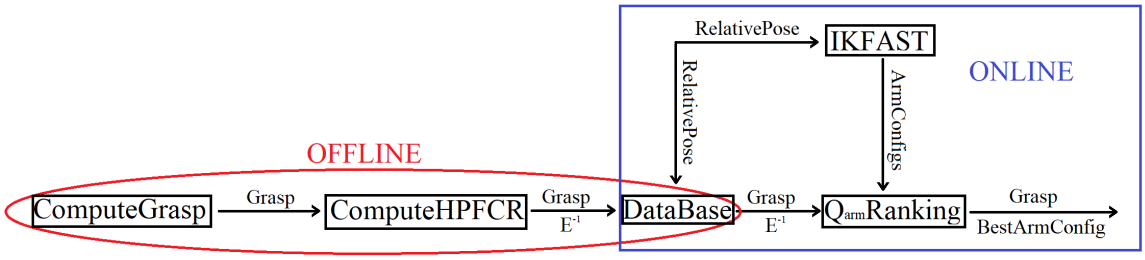



Figure 5.2: The framework of our entire system.

Algorithm 3 sketches the online part of the algorithm. The input is the target grasp pose, $\mathbf{p}_r$, and the matrix, $\mathbf{E}^{-1}$, associated with the grasp. The output is the best arm configuration, $\mathbf{q}_a$. The online part of the algorithm is efficient because it relies just on one call to IKFAST and several matrix multiplications.

**Algorithm 3** $Q_{arm}$ ranking algorithm

**Input** : $\mathbf{p}_r, \mathbf{E}^{-1}$
**Output**: Best arm configuration $\mathbf{q}_a$
$Q \leftarrow \text{IKFAST}(\mathbf{p}_r)$
**for all** $\mathbf{q}_j \in Q$ **do**
    $\mathbf{J}_j \leftarrow Jacobian(\mathbf{q}_j)$
    $e_i \leftarrow eigenvalues(d\varepsilon^2_{max}\mathbf{E}^{-1}(\mathbf{J}_j\mathbf{J}_j^T))$
    $Q_{arm_j} \leftarrow \frac{1}{\Sigma_i(\sqrt{e_i}-1)^2}$
$\mathbf{q}_a \leftarrow \{\mathbf{q}_j | j = \arg\max_j Q_{arm_j}\}$

### Methods comparison

In this section, we provide two ways to approximate the noise distribution at the end-effector, i.e., the sampling based approach and the analytical based approach. The sampling based approach is capable of providing a rather accurate distribution. In order to maintain accuracy, more time is required to process enough samples, whereas, the analytical based approach only relies on the calculation of the noise ellipsoid, making it more efficient. The major calculation of the noise ellipsoid is based on the computation of the Jacobian matrix, which is usually calculated upfront. However, the distribution given by analytical based approximation is inaccurate. We will show in the experiment section, the noise ellipsoid is only useful to predict how the noise distribution is aligned. When relating these two methods to grasping, the sampling method is more accurate, which is capable of providing a tight estimate of the end-effector distribution. However, this method is limited in practice due to its inefficiency in calculating the bound. On the other hand, the analytical based approach can not provide a tight bound, but it can be used to guide the online process to quickly select the best arm configuration based on the predetermination of the high probability force closure region.

## 5.2 Experiments

Here, we first provide an illustration of the performance of empirical estimators of mean, variance and skewness on end-effector position and showcase the differences in noise profiles that can occur, even for simple planar linkages. Then, we consider sampling based aproximation of grasp quality for a 7-DOF Kuka Leightweight (LWR) robot arm. Next, we show how analytical based approximation is used to estimate the end-effector pose. Finally, we provide an example of the calculation of the high probability force closure region and show the performance of the analytical based approach in simulation.

### 5.2.1 Empirical estimation of noise on end-effector position

We start by considering the simple planar 2 and 3 link arms depicted in Figure 5.3. We apply zero mean Gaussian Noise with a variance of $\sigma^2 = 0.0175^2$, and display two nominal arm positions

with an identical end-effector pose $(x, y) = (2, 2)$, as well as a visualization of the end-effector position for $10,000$ samples from the noise distribution.

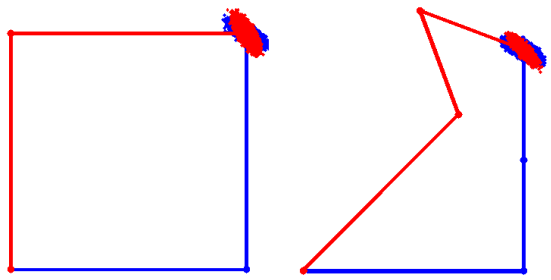


Figure 5.3: A two and three link planar robot arm with mean joint configurations and samples of end-effector poses under Gaussian Noise. Observe that while in both cases the nominal configuration $\mathbf{q}_0$ would bring the end effector to the same position, because of the noise, we have different empirical means and covariances.

The distributions of the end effector positions differ not only in spatial orientation, but also in spread. Their empirical covariances $\Sigma_1, \Sigma_2$ differing in Frobenius norm by 0.000437 for the two link arm 0.0014 for the three links arm. To better understand the convergence of these sampling based estimators for the end-effector position, we sampled 10,000 samples from $\mathcal{N}(0, \sigma^2)$ for $\sigma^2 \in \{0.0033^2, 0.0133^2, 0.0233^2, 0.0333^2\}$ in 10 trials. The top panel in Figure 5.4 displays the convergence of the mean of these 10 runs for the estimators of mean, covariance and skewness to a 'ground truth' given by the mean estimate of 100,000 samples. As we can observe, the number of required samples to obtain satisfactory convergence heavily depends on the variance of the noise profile. The bottom panel in the same figure displays the same experiment for the Kuka LWR arm. The charts show that despite the different number of degrees of freedom, 10,000 samples is sufficient to numerically determine the empirical estimators.

### 5.2.2 Grasp Quality and Noise

We next study the dependence of grasp quality on nominal grasp configuration under noise. After describing our experimental setup, we consider the convergence of sampling based grasp quality estimators followed by experiments illustrating interesting cases of the dependence of probabilistic grasp success on variance, object shape and joint configuration.

#### 5.2.2.1 Experimental setup

We used the VRep simulator [83] to simulate a 7 degree of freedom leightweight KUKA arm with a Schunk Dexterous hand with three fingers and 7 DOF displayed in Fig. 5.1. We determined a nominal pre-grasp joint configuration using a grasp planner we developed following the same ideas used in GraspIT! [67], i.e., we sample hand poses around the object to be grasped and through physical simulation we determine if the grasp obtained closing the fingers from the corresponding pose results in a force closure grasp. If this is the case the grasp configuration is retained, otherwise it is discarded. Figure 5.5 illustrates this process. All computations are performed using VRrep's built

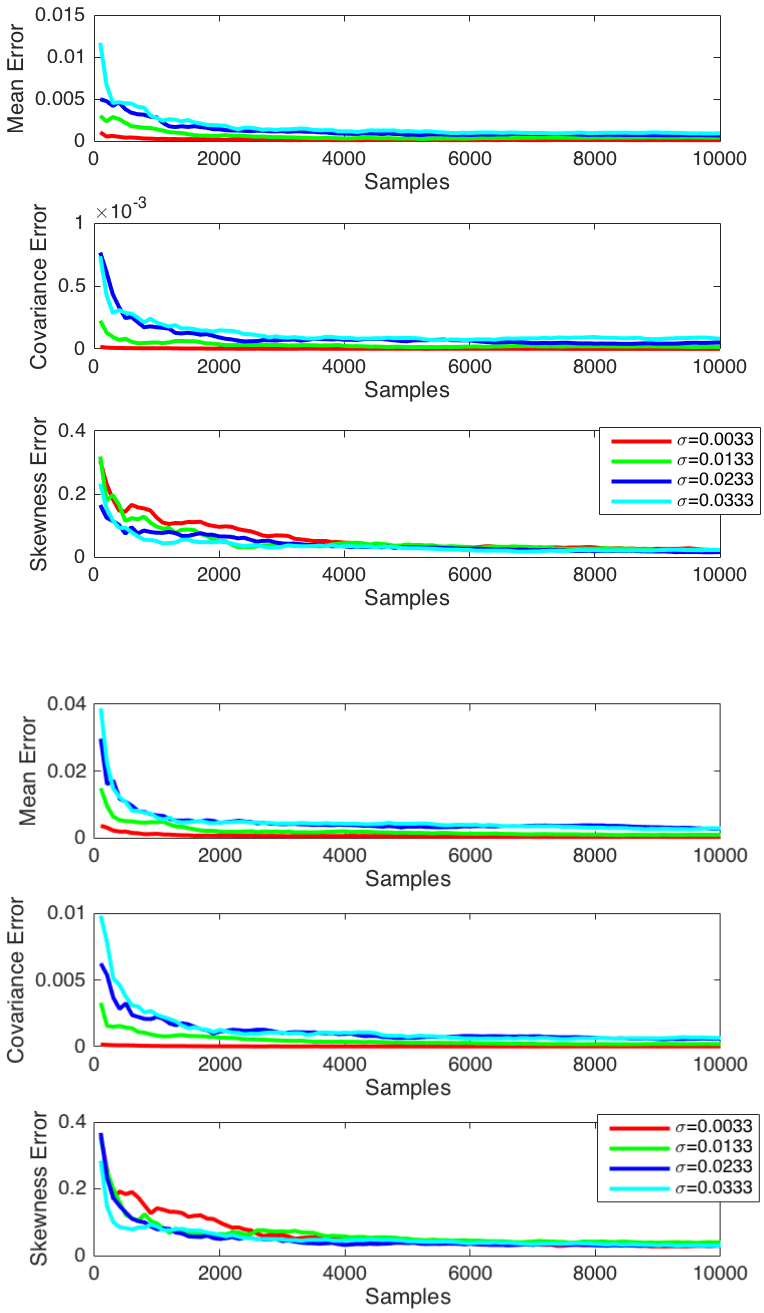



Figure 5.4: $L_2$ error convergence of mean position, covariance error (measured in Frobenius norm), and skewness under isotropic Gaussian Noise as the number of samples is increased. The top three figures display results for the 2 link arm, while the bottom three display results for the Kuka LWR. We observe that the standard deviation influences the initial convergence rate, but in all cases 10,000 samples seem sufficient to empirically determine the estimators.

in features for collision detection, and the method eventually returns a set of pre-grasp configurations that will give force closure grasps.

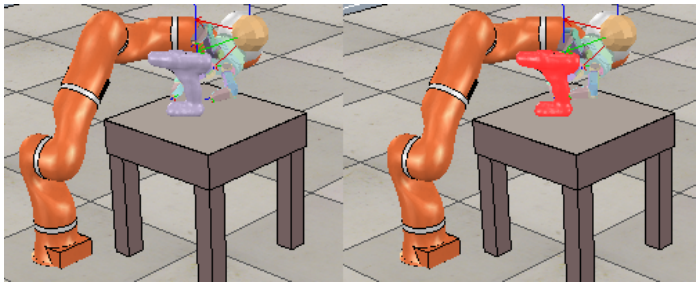


Figure 5.5: The left figure displays the arm and hand in a pre-grasp configuration. Pre-grasp configurations are generated randomly. Next, the fingers are closed and contact points with the surface of the objects are determined as shown on the right figure. If the resulting set of contact points gives force closure the grasp is retained, otherwise it is discarded.

At run time, to generate random samples for a nominal joint-configuration, we added isotropic Gaussian Noise $\mathcal{N}(0, \sigma^2)$ to each joint of the robot arm and executed the auto-close procedure. Due to noise, the grasp may fail for various reasons. First, the fingers may miss the object during the auto-close step, and therefore enough contact points cannot be established to restrain the object. In addition, there are cases in which all fingers make contact with the object, but the resulting placements still does not yield a force closure configuration. Both these instances will be indicated as *failures* in the following, because due to noise during the execution, a planned grasp does not achieve its desired force closure objective. If instead all fingers establish contact with the object and these points indeed yield force closure, we compute the Ferrari and Canny grasp quality measure. For our experiments, we used the objects displayed in Figure 5.6.

Figure 5.6: Objects used in experiment

#### 5.2.2.2 Empirical estimation of noise on grasp quality and probability of force closure

Paralleling the experiments we presented in section 5.2.1, we start by assessing the impact of noise on grasp effectiveness. In particular we empirically estimate the probability of force closure, i.e., the probability of obtaining a successful grasp and the grasp quality metric when a successful grasp is established. Figure 5.7 show the empirical estimation for the probability of force closure $P(FC)$ (bottom two charts) and the grasp quality metric (top two charts). The curves were obtained

averaging five different grasps for the spray flask object and show a slower convergence rate when compared with Figure 5.4. This is true for both the grasp quality metric $Q$ and $P(FC)$[2].

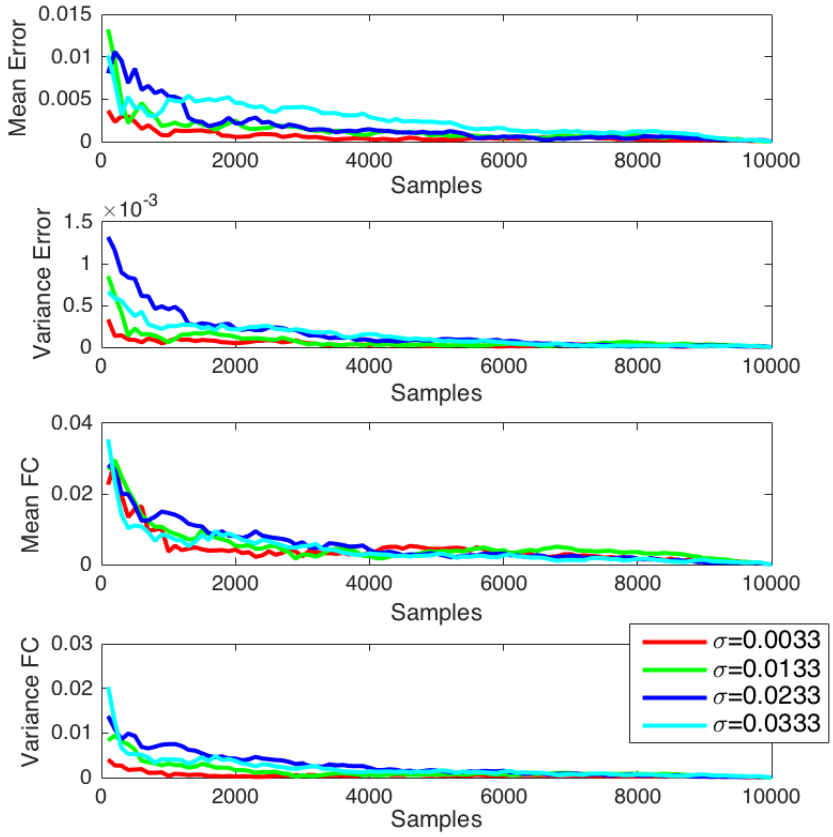



Figure 5.7: Empirical estimation of the convergence rate for the estimators of the Ferrari and Canny grasp quality metric (top charts) and probability of force closure (bottom charts).

#### 5.2.2.3 Impact of arm configuration

One of the limitations of current grasp metrics is that they do not consider the arm configuration used to implement a target grasp. However, when noise is explicitly modeled, significant differences may emerge when comparing the robustness of different arm configurations achieving the same contact points. To study this effect, we start by considering two objects grasped with the same contact points but different arm configuration. Figure 5.8 shows the two objects and the variability for the grasp. Colored *sticks* are used to display the normal to the object surface at each

[2]Note that since $P(FC)$ is the probability of a binary indicator random variable, the mean of $FC$ is equal to $P(FC)$.

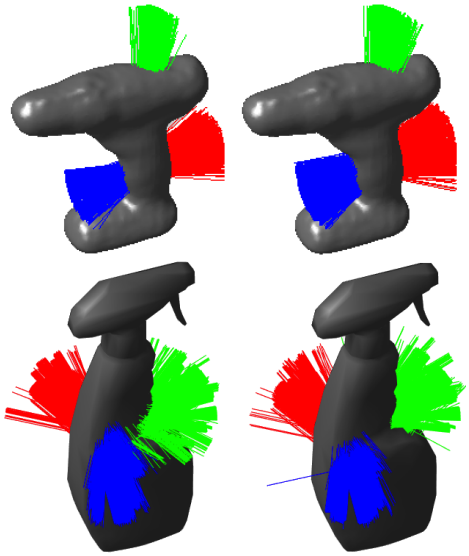


Figure 5.8: Distribution of grasp contact points on two objects where two different arm configurations are used to implement the same grasp.

contact point, with different colors used for the three fingers. The figures display the results obtained with 10,000 samples. For the drill object we determine that the mean grasp quality metric is comparable in the two cases (0.0982 versues 0.0910) and the variance is comparable too (0.0025 versus 0.0028). However, there is a remarkable difference in the probability of force closure $P(FC)$. The left configuration achieves a value of $P(FC) = 0.50$, whereas the right configuration has a value $P(FC) = 0.31$. Similar observations are made for the spray flask, where the right configuration has $P(FC) = 0.57$ and the left one has $P(FC) = 0.42$.

#### 5.2.2.4 Impact of noise on grasp quality

Finally, we evaluate the impact of noise on $P(FC)$ and the grasp quality metric for different objects, grasps, and noise levels. We start considering the spray flask object with five different grasps and four different noise levels. Figure 5.9 shows the distribution of normals to the contact points using the same coloring used in Figure 5.8. Each row corresponds to a different grasp, and each column to a different variance in the joints, i.e., $0,0033^2$, $0.0133^2$, $0.0233^2$, $0.0333^2$ (left to right). Table 5.10 displays the numerical results for these grasps, i.e., probability of force closure $P(FC)$ and the mean and variance of the Ferrari and Canny grasp quality metric for the successful grasps. Results are obtained using 10,000 samples. From the table we can observe that $P(FC)$ appears to be more sensitive to variations in the joint noise, as measured in terms of the variance $\sigma^2$. Grasp quality varies too, although the variation is more modest and at times even non-monotonic.

| $\sigma^2 = 0.0033^2$ | | | $\sigma^2 = 0.0133^2$ | | | $\sigma^2 = 0.0233^2$ | | | $\sigma^2 = 0.0333^2$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean | Variance | $P(FC)$ | Mean | Variance | $P(FC)$ | Mean | Variance | $P(FC)$ | Mean | Variance | $P(FC)$ |
| 0.026383 | 0.000683 | 0.5275 | 0.071357 | 0.003324 | 0.3094 | 0.071521 | 0.002820 | 0.2158 | 0.069315 | 0.002447 | 0.1972 |
| 0.161483 | 0.002088 | 0.4763 | 0.105348 | 0.004519 | 0.5033 | 0.094063 | 0.003341 | 0.3777 | 0.091105 | 0.003066 | 0.3040 |
| 0.055606 | 0.000868 | 0.6537 | 0.069675 | 0.002674 | 0.3228 | 0.073776 | 0.002924 | 0.2264 | 0.072395 | 0.002666 | 0.1850 |
| 0.067810 | 0.000094 | 0.6038 | 0.070409 | 0.001154 | 0.2673 | 0.066890 | 0.001361 | 0.1899 | 0.067309 | 0.001477 | 0.1625 |
| 0.141029 | 0.001225 | 0.5084 | 0.100313 | 0.002700 | 0.5749 | 0.089174 | 0.002718 | 0.3995 | 0.087669 | 0.002671 | 0.3422 |

Table 5.1: Probability of force closure, mean and variance of the grasp quality metric for the grasps in Figure 5.9. Grasp quality metric is computed only for successful grasps.

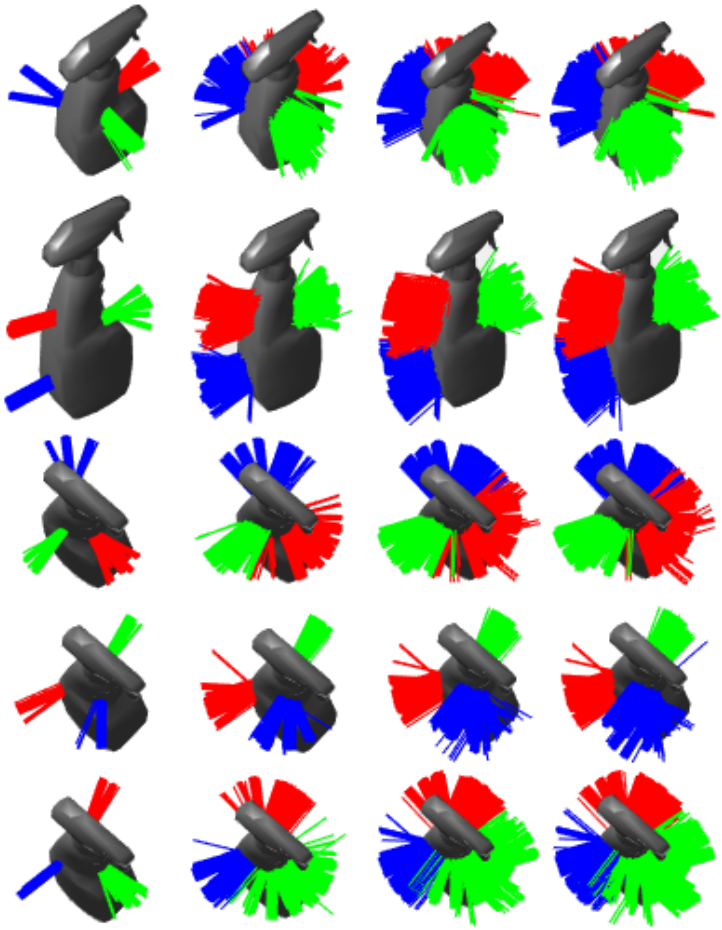


Figure 5.9: Distribution of the normals to the contact points for different grasps over the spray flask object.

For the next experiment, we considered ten different grasps and evaluate the effect of noise in terms of P(FC) and grasp quality for the successful grasps for the same four objects shown in Figure 5.6. Results are displayed in Figures 5.10 and 5.11. Figure 5.10 displays the result

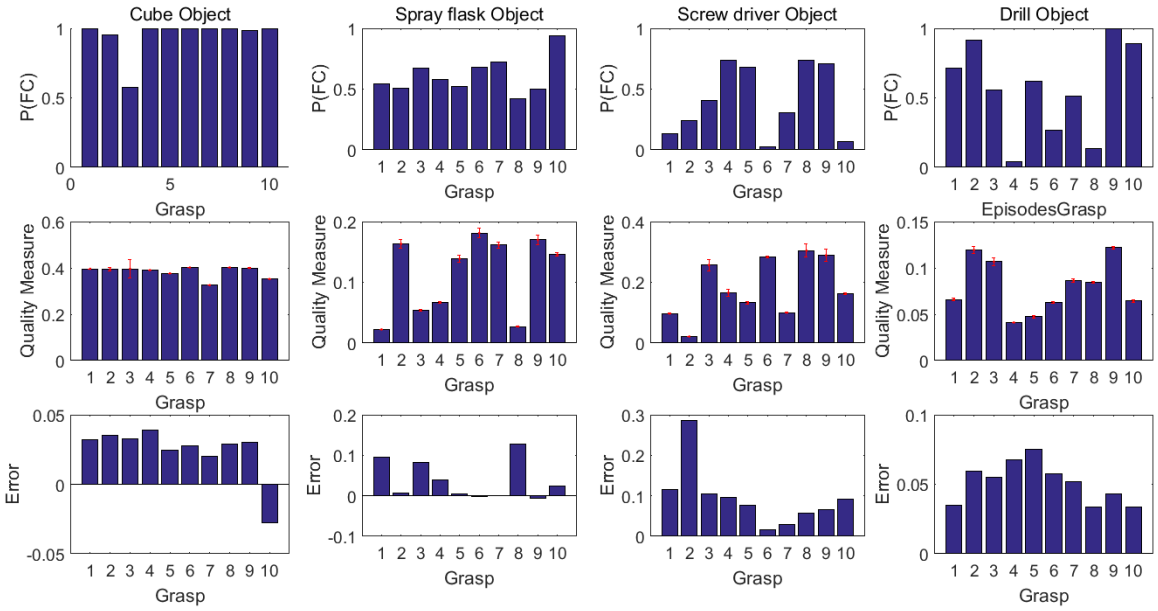



Figure 5.10: Success rate, quality measure, and error for ten different grasps with different contact points and different arm configurations.

for grasps configurations consisting of different contact points and different arm configurations. Figure 5.11, instead, considers the same contact points, but with different arm configurations due to different placements of the objects with respect to the arm. In both cases joint angles were affected by Gaussian Noise $\mathcal{N}(0, 0.0033^2)$. As shown below in Figure 5.11, $P(FC)$ is high for a simple object like the cube, but dramatically varies for more complex objects. The *quality measure* is the average value for the Ferrari and Canny metric and is limited to the grasps for which force closure is obtained. In the two figures we can see more variation for the first case that involve both the contact points and the arm configuration. Finally *Error* is the difference between the theoretical grasp quality metric predicted by the planner without considering noise sources, and the average grasp quality metric experimentally determined. As shown, this value is almost always positive, indicating that, in most cases, noise alters the contact points in a way that negatively impacts the quality measure, although in some cases this observation is not valid, i.e., noise actually ends up generating contact points with a better quality.

### 5.2.3 Analytical based estimation of noise on end-effector pose

The noise ellipsoid produces an ellipsoid approximately aligned with the sampled end-effector distribution when a bounded noise is applied at each joint. Figure 5.12 shows the result of a simple experiment for a planar three-link arm. The noise applied to each joint is drawn from a uniform distribution with support $[-0.01, 0.01]$. The top subfigure shows the ratio between the eigenvalues defining the noise ellipsoid and the sampled end-effector position distribution with the same end-effector position. With over 1,000 samples we can observe that the ratio between the maximum and

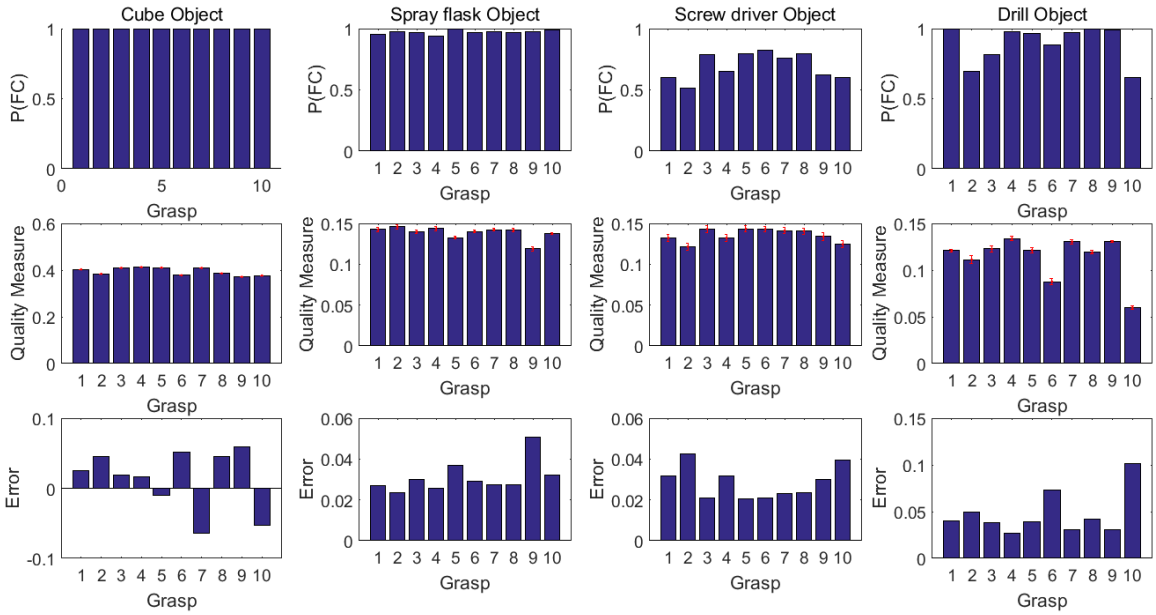



Figure 5.11: Success rate, quality measure, and error for ten different grasps with same contact points but different arm configurations.

minimum eigenvalue remains close to 1. The bottom subfigure shows, instead, the 2-norm of the difference between the eigenvector matrices. The small values indicate the substantial alignment between the noise ellipsoid and the sampled end-effector position, i.e., the fact that the approximation error introduced by the first order expansion, is limited.

Similarly, Figure 5.13 shows the comparison of the noise ellipsoid and the actual sampled end-effector distribution for a KUKA LWR arm. Data was gathered in simulation, thus allowing for flexibility in noise generation while relying on a high fidelity model for the arm. The noise applied at each of the seven joints is drawn from a uniform distribution with support $[-0.01, 0.01]$. The left subfigure shows the mean and variance of the scale between the eigenvalues, while the right subfigure shows the determinant of the difference in the matrix containing eigenvectors. Both figures confirm that the error is small.

Finally, Figure 5.14 contrasts the sampled end-effector position distribution, when noise drawn from a uniform distribution with support $[-0.01, 0.01]$ is applied to all joints. For each shown configuration, 10,000 samples were generated to determine the distribution of the end-effector position. The figure confirms that the sampled end-effector distribution is well aligned with the noise ellipsoid. For the 7 DOF KUKA arm, the noise ellipsoid therefore provides a good indication of the actual sampled noise distribution.

#### 5.2.3.1 High Probability Force Closure Region

Our grasp quality measure $Q_{arm}$ relies on the high probability force closure region. Figure 5.15 shows a SDH hand grasping a bottle (this study is instead done using the VREP simulator),

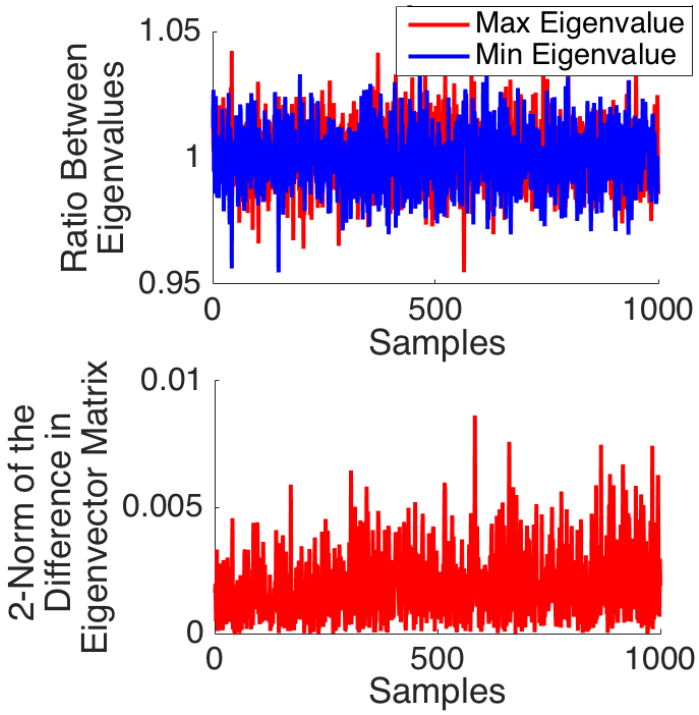



Figure 5.12: The top subfigure shows the ratio between the eigenvalues of the noise ellipsoid and the sampled end-effector position distribution for a 3 link planar arm. The red line represents the ratio of maximum eigenvalue and the blue line represents the ratio of minimum eigenvalue. The bottom subfigure shows the 2-norm of the difference in the matrix representing all eigenvectors.

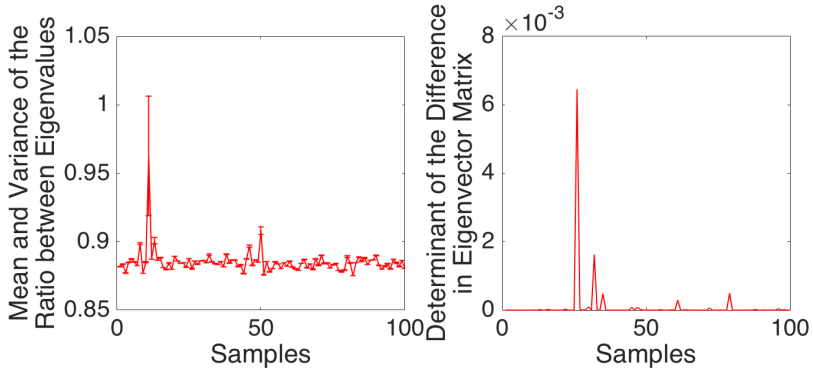



Figure 5.13: The left subfigure shows the mean and variance of the scale between the eigenvalues of the noise ellipsoid and the sampled end-effector position distribution for KUKA LWR. The right subfigure shows the determinant value of the difference in the matrix representing all eigenvectors.

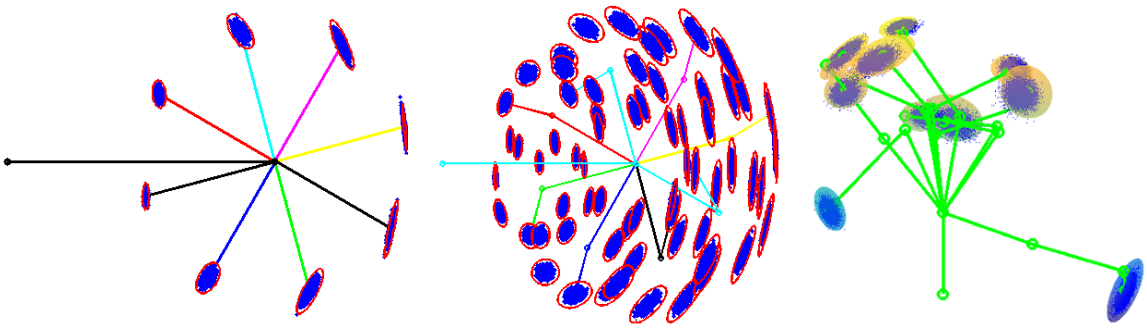


Figure 5.14: Sampled end-effector position distribution with noise shown in blue and noise ellipsoid for a 7 DOF KUKA arm (shown in colored ellipsoid).

whereas Figure 5.16 shows the corresponding high probability force closure region projected in the position domain and the orientation domain. All points shown as either red dots or blue dots represent force closure grasps, but red dots are the points within the ellipsoid associated with the high probability force closure region. To be more specific, the high probability force closure region is sampled around the grasp with the bound $b = 0.03$ and step size $s = 0.06$. The confidence percentage $c$ was set to 70% to filter out points too far away from the center of the ellipsoid $E$. The quality threshold was selected with $t_q = 0.2144$, which is equal to half of the maximum quality value over all samples. This value is selected to achieve a force closure probability of more than 50% within the high probability force closure region. As shown in Figure 5.16, the ellipsoid in the orientation domain is comparably larger than the ellipsoid in the position domain. Blue dots in the left figure are not evenly distributed, as can be observed, noting the lack of points in the left part. These observations confirms that the high probability force closure region is biased towards good grasps, so choosing the associated arm configuration will benefit the final outcome.

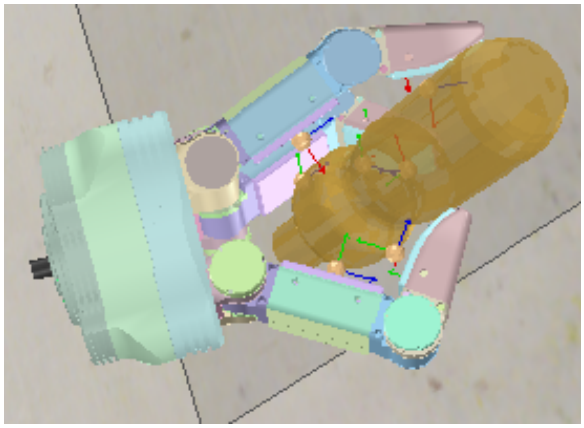


Figure 5.15: Example grasp on a bottle object.

Figure 5.17 shows the relationship between the force closure probability and the volume of the ellipsoid $E$ with respect to the quality threshold $t_q$. $t_q$ is sampled from 10% of the maximum

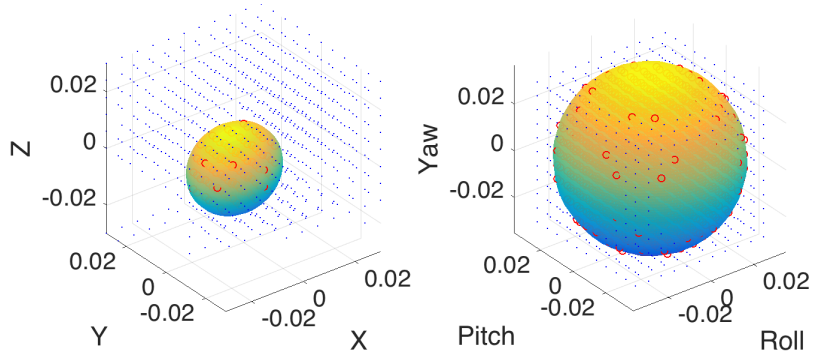



Figure 5.16: The High Probability Force Closure Region projected in $XYZ$-coordinate system (left) and $RPY$-coordinate system (right).

quality up to 90% of the maximum quality (that in this specific case is 0.4288). As $t_q$ increases, the force closure probability also increases whereas the volume of $E$ shrinks as expected. The ideal probability force closure region would be the largest region enclosing the probability threshold. Thus, our iterative method that increases $t_q$ at each iteration is sufficient to locate the ideal probability force closure region.

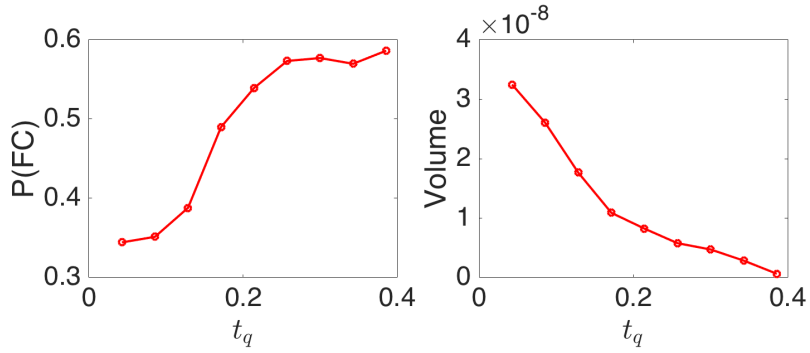



Figure 5.17: Trend of force closure probability (left) and volume of fitting ellipsoid (right) with respect to $t_q$.

To accelerate the speed to compute the high probability force closure region, we can first investigate the variance of the end-effector distribution in both the position and orientation domains. For example, the end-effector distribution of the KUKA LWR arm distribution varies more in the position domain compared to the orientation domain. Therefore, we can decrease the size of the bound $b$ for the orientation domain to improve the efficiency of the algorithm.

However, since the high probability force closure region is either generated using fixed uniform sampling or random sampling, some limitations may apply. Figure 5.18 shows two examples of the high probability force closure region in 2D. The left example represents a worst case for using fixed uniform sampling while the right example represents a worst case for random sampling. Both examples use the confidence percentage $c = 0.75$ and the probability force closure values are

64.86% and 56.25% from left to right. As for the left example, only if the region of the end-effector noise distribution is larger than the high probability force closure region, will we be able to include all of the force closure grasps. That is to say, if we apply relatively small noise, then we might not be able to obtain any force closure grasps. For the right example, since the probability force closure value is calculated by the percentage of force closure grasps among all grasps within the high probability force closure region, there might exist a huge portion of the high probability force closure region outside bound $b$ that has not been explored. In this case, the actual probability force closure value for the entire high probability force closure region might be significantly smaller than the calculated value, i.e., 56.25%.

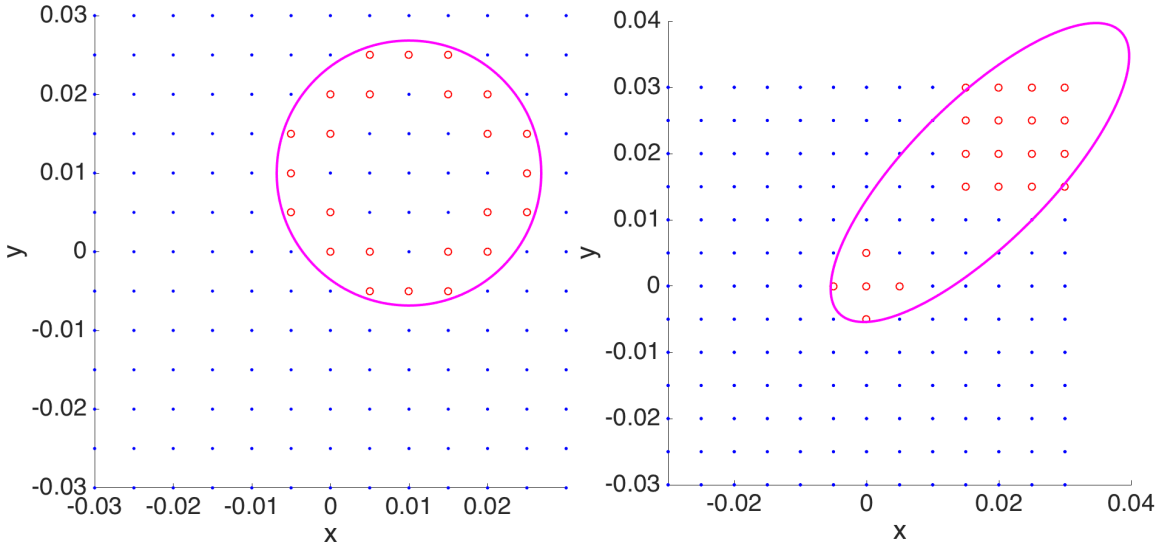



Figure 5.18: Two examples of HPFC in worst case scenario. The blue points are non-force closure grasps, red points are force closure grasps, and the ellipsoid is the calculated high probability force closure region.

#### 5.2.3.2 Validation in Simulation

To perform an end-to-end validation of our method in simulation, we generate the end-effector noise distribution by applying noise drawn from a truncated Gaussian distribution at each joint and for each end-effector pose we compute the corresponding grasp quality. In particularly, we compare the the best arm configuration and the worst arm configuration selected by $Q_{arm}$ optimizing $P(FC)$ as defined before and $\underline{\mathrm{E}}[Q]$ defined as the expected force closure quality among all force closure grasps. Figure 5.19 shows the comparison between the best arm configuration (in red) and the worst arm configuration (in blue) for five different objects under different noise averaging over 10 different grasps. The $P(FC)$ and $\underline{\mathrm{E}}[Q]$ values of the best configuration for all objects are in average higher than the value for the worst arm configuration, especially when the noise is large. The high probability force closure region is computed by sampling within a certain range. For a relatively small noise, most arm configurations might be included inside the high probability force closure

region, therefore the $P(FC)$ shows less change among all arm configurations. The difference is more significant when the noise is large, such that a more aligned noise distribution against the high probability force closure region has a higher chance to be force closure. Similar results were obtained for other objects.

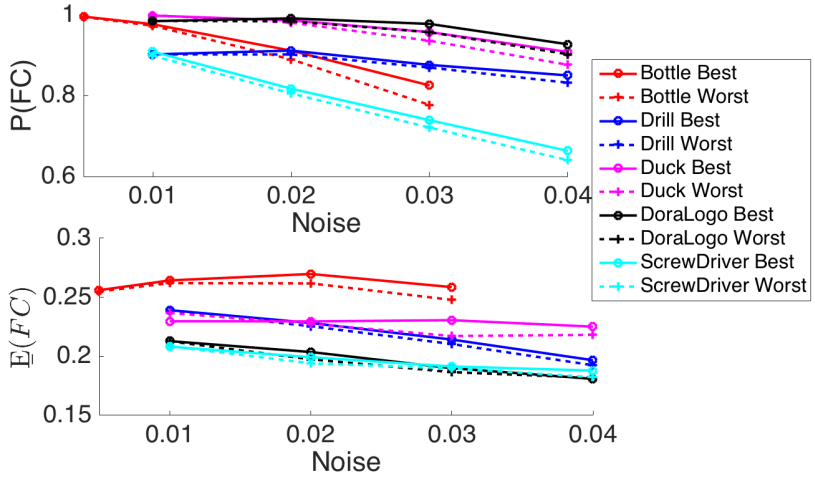



Figure 5.19: $P(FC)$ and $\underline{\mathrm{E}}[Q]$ for 5 different objects under different noise boundary. For each value $\varepsilon$ for the noise bound, the superimposed noise is a truncated Gaussian $\mathcal{N}(0, (\frac{\varepsilon}{3})^2, \varepsilon)$.

## 5.3 Conclusions and future work

In this chapter we have formulated two frameworks to study the interplay between grasp quality evaluation functions, noise, and the mechanical structure of the robotic arm. This work fills a gap in the grasp evaluation literature because grasp quality metric studies have mostly evaluated grasps as a set of deterministic contact points. The first framework used a sample based approach to explicitly point out the importance of considering the mechanical structure of the robotic arm with respect to applied joint noise. Unsurprisingly, our experiments show that the probability of grasp quality is strongly affected by this aspect. However, the sampling based approach only illustrated the importance of the problem but is very inefficient for online use. As an extension, we proposed an analytical based approach to solve this problem efficiently, which made it applicable for online use. The analytical method is based on two important concepts, i.e., the high probability force closure region, and a new grasp quality metric $Q_{arm}$, to explicitly consider the structure of the arm and disturbances when evaluating different ways to implement a target grasp. The outcome of this approach will be the arm configuration that provides a relatively high staleness when actuating the associated grasp. This metric can be integrated with existing grasp quality measures to quantify the quality of the entire robotic system.

In this chapter we have mostly concentrated on the noise affecting the robot joints and how it impacts the pose of the end effector and the success rate of a grasp or its quality. One could argue that in some robots this noise may be negligible, though the recent advent of low cost platforms

with passive joints like Baxter, demonstrate the opposite. Irrespective of that, our study outlines the effects of a mismatch between the expected relative pose between the robotic hand and the object to be grasped. Hence, our same conclusions extend to the case where the noise affecting the robot is negligible, but there is uncertainty in the pose of the object being grasped.

## Acknowledgments

We thank Jeff Mahler at UC Berkeley for providing the meshes for the objects used in the experiments and Florian Pokorny and Ken Goldberg at UC Berkeley for useful discussions regarding this work.

# Chapter 6

# Grasp Quality Metric Improvement Considering Hand Configuration and Target Object

In this chapter, we focus on embedding hand features and local geometry information into the grasp quality metric. The methods we propose are all based on the friction cone. Our main focus is on hand configuration and how the hand configuration affects the maximum force, which can be applied at each contact. We formulate the maximum force as a variable that scales the unit friction cone at each contact. This scaled friction cone will then shape the grasp wrench space and produce more robust grasp candidates. Interestingly, we found that we can also use the friction cone to add information of an object's local geometry. We are particularly interested on concave objects because we believe that having contact at the concave portion of the object will provide additional support to restrain the object. This effect can be addressed by expanding the friction cone along the direction of the additional supports reflecting the local geometry of the object.

Let us first start with two examples. First lets look at Figure 6.1. This figure displays the same gripper having two grasps with identical contact points on the same object. Using any metric related to the grasp wrench space would give you identical results in quality. However, due to a different hand configuration, the maximum force, which can be applied on the object, is also different. In section 6.1, we will discuss in detail, how the maximum finger force is calculated at each contact. In general, the grasp with higher force applied at each contact is considered to hold the object tighter, i.e., the ability to resist disturbance force is better. This feature should be taken into account for grasp quality measure.

Now, lets consider the second example shown in Figure 6.2. The leftmost figure shows a concave object with the blue dots indicating two contact points for a possible grasp. Similarly, the rightmost figure illustrates a convex object and the blue dots also indicate two contact points for a grasp. Most grasp quality metrics would rank these two grasps as equivalent because, as shown in the middle figure, locally, the two surfaces coincide and, therefore, the same grasp wrench space is generated. Real world experience and common sense, however, indicate that the one on the left is more stable, i.e., it will be capable of resisting disturbance wrenches of a higher magnitude (think

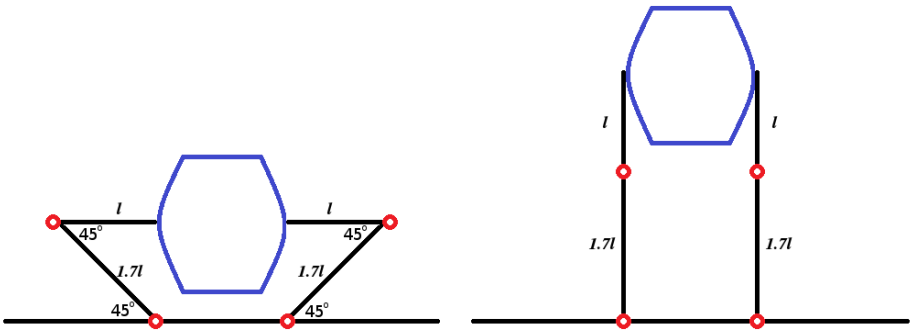



Figure 6.1: The above figures display two grasps with identical contact point configurations.

for example of a disturbance force acting vertically on the middle of the lower edge of both objects). This effect is magnified if we also consider that, in practice, contacts hardly occur at just one point.

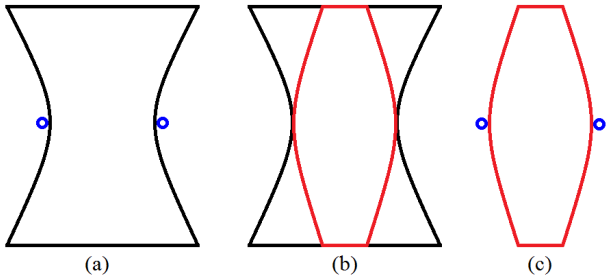



Figure 6.2: Subfigure (a) shows two grasping points on the surface of a concave planar object, whereas subfigure (c) considers a convex planar object. Subfigure (b) in the middle show that the normals at the grasping points are identical.

These two simple examples motivate the work present in this chapter. We start with revising the definition of the friction cone with the objective of better evaluating the ability to restrain objects featuring negative curvature boundaries, i.e., concave objects, and taking into account the effect of finger force. This approach overcomes classical methods that exclusively consider normals at the contact point, but ignore the maximum possible force at each fingertip and the curvature of the local neighborhood to the contact point.

The rest of the chapter is structured as follows. In section 6.1 we formalize the concept of the hand force dependent grasp quality metric and the curvature dependent grasp quality metric. We sketch how this can be used for grasp planning. The approach is demonstrated in simulation and on a real robotic arm with a multifingered robotic hand in section 6.2. Finally, we draw conclusions and outline future work in section 6.3.

## 6.1 Grasp Quality Metric Improvement

In this section we will develop the mathematical foundations for the method we propose. The section is organized into two parts. In this first part, we will focus on hand configuration. We will address the fundamental way to derive the maximum finger force applied at each contact and formulate the friction cone to take this information into account for the grasp quality measure. In the second part, we will propose our method to connect grasp quality and local geometry of the object being grasp that exist in a negative curvature feature. This connection is also established through the friction cone. To explain the concepts, we use pictures displaying planar objects, but the method we develop aims at the three-dimensional scenario.

### 6.1.1 Grasp Quality Metric with Hand Configuration

To start with, we address this problem on a dexterous hand with a limited configuration space for each finger. In particular, we will start with the most commonly used finger type, which only has two phalanges and limits each finger to move in a 2D plane.

Let's consider the setup shown in Figure 6.3 with such a basic configuration.

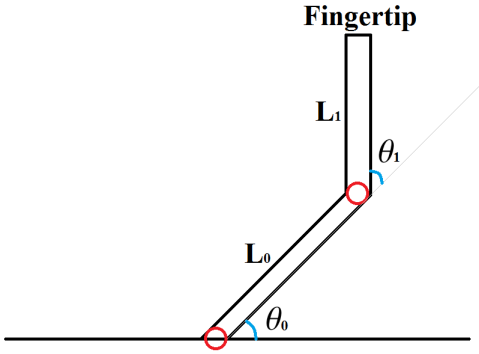



Figure 6.3: Simple two link finger model.

Assume the joint angles are $\theta_i$ for joint i. $L_i$ denotes the length of the phalanges connected to joint i. The fingertip pose $p_{ee}$ can then be derived as:

$$p_{ee} = \begin{pmatrix} L_0 \cos\theta_0 + L_1 \cos(\theta_0 + \theta_1) \\ L_0 \sin\theta_0 + L_1 \sin(\theta_0 + \theta_1) \end{pmatrix} \tag{6.1}$$

The corresponding Jacobian matrix, defined as $J = \frac{\delta p_{ee}}{\delta \theta}$, can be expressed using the following equation:

$$J = \begin{pmatrix} -L_0 \sin\theta_0 - L_1 \sin(\theta_0 + \theta_1) & -L_1 \sin(\theta_0 + \theta_1) \\ L_0 \cos\theta_0 + L_1 \cos(\theta_0 + \theta_1) & L_1 \cos(\theta_0 + \theta_1) \end{pmatrix} \tag{6.2}$$

Assume the torque applied at each joint is represented as $\mathbf{T} = [\tau_0, \tau_1]^T$ and the force applied at the contact is $\mathbf{F}$. The relation between $\mathbf{T}$ and $\mathbf{F}$ is therefore:

$$\mathbf{T} = J^T \mathbf{F} \tag{6.3}$$

The derivation of the equation 6.3 is straightforward. In general, work is the dot-product of a force and a displacement or a torque with an angular displacement. According to the principle of virtual work, assume these displacements become infinitesimally small, then:

$$\mathbf{F}\delta p_{ee} = \mathbf{T}\delta\theta \tag{6.4}$$

Since by the definition of the Jacobian matrix, where $J = \frac{\delta p_{ee}}{\delta\theta}$, equation 6.4 may also be written as:

$$\mathbf{F}^T J\delta\theta = \mathbf{T}^T\delta\theta \tag{6.5}$$

Which yields to equation 6.3.

So if we want to know the end effector force range we can use the inverse Jacobian matrix:

$$\mathbf{F} = (J^T)^{-1}\mathbf{T} \tag{6.6}$$

Assume that the maximum torque provided by each joint $\tau_{max}$, such that the amount of torque applied at each joint is within the range $[-\tau_{max}, \tau_{max}]$. Then the force ellipsoid can be derived to represent all possible forces that can be applied at each contact. Following equation 6.6, the force ellipsoid is then represented as:

$$\mathbf{F}^T JJ^T\mathbf{F} = c\tau_{max}^2 \tag{6.7}$$

where $c$ is the number of joints of each finger. Note that the force ellipsoid for a finger moving in 2D is, indeed, an ellipse instead of an ellipsoid.

This problem can be easily generalized to 3D. However, if the finger can move in 3D, then the finger itself forms a simple manipulator. The force applied at contact is therefore generalized to

$$\mathbf{F} = (J^T)^{+}\mathbf{T} \tag{6.8}$$

where $(J^T)^+$ is the pseudo-inverse of the transpose of the Jacobian matrix. After obtaining the force distribution at each finger contact, we can start to formalize it and embed it into the grasp quality metric using the friction cone.

To prevent slipping on the surface, only the forces within the friction cone should be considered. Assume that $f_j(\mathbf{p})$ denotes the force generated by the hand at contact point $\mathbf{p}$ maximized along $j$th direction inside the contact friction cone. We can therefore write $f_j(\mathbf{p})$ as

$$f_j(\mathbf{p}) = f_j(\mathbf{p})^N + f_j(\mathbf{p})^T \tag{6.9}$$

where $f_j(\mathbf{p})^N$ and $f_j(\mathbf{p})^T$ is the decomposition of $f_j(\mathbf{p})$ along the surface of its normal direction and its tangential direction. $f_j(\mathbf{p})^T$ will be cancelled by the friction. Therefore the force we take into account is $f_j(\mathbf{p})^N$. The magnitude of this force is, thereby, the actual force that affects the contact. Therefore, the maximum force $f_H$ applied at contact point $\mathbf{p}$ with friction coefficient $mu$ can be represented by

$$f_H(\mathbf{p}) = max_j \left\{ f_j(\mathbf{p})^N | f_j(\mathbf{p})^T \leq \mu f_j(\mathbf{p})^N \right\} \tag{6.10}$$

Figure 6.4 shows an 2D example for the forces at contact point $\mathbf{p}$. The black ellipse captures the possible force a 2D finger can apply. The lines in red are the boundary for the friction cone. The

area in green is the intersection of the friction cone and possible finger force, which shows the valid finger force that can be applied. The line in blue captures the maximum finger force $f_H(\mathbf{p})$, it has the maximum projection along the normal direction of the friction cone. The only difference for the 3D case with a 2D finger is that the friction cone is represented as a cone in 3D instead of a cone in 2D.

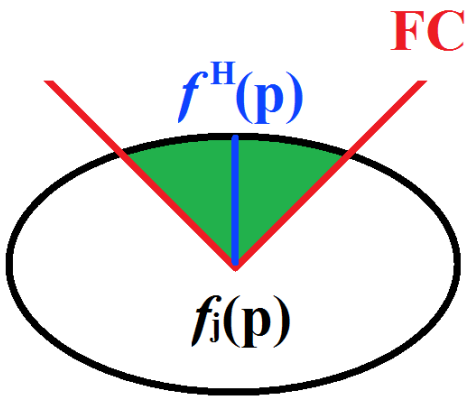



Figure 6.4: 2D example that locates $f_H(\mathbf{p})$

In order to reflect this information in the grasp wrench space, we can simply scale the friction cone at contact point $\mathbf{p}$ with $f_H(\mathbf{p})$ instead of using a unit friction cone. Assume that $f_1$ is the unit normal vector of the friction cone and $f_2$ and $f_3$ forms the tangent plane with respect to $f_1$. Therefore we define the hand friction cone $F^H(\mathbf{p})$ as

$$F^H(\mathbf{p}) = \left\{ \mathbf{f} \mid \sqrt{f_2^2 + f_3^2} \leq \mu f_H(\mathbf{p}) f_1 \right\} \tag{6.11}$$

The newly defined $F^H(\mathbf{p})$ captures the effect of the maximum finger force and can be applied directly to generate the grasp wrench space. The grasp wrench space generated using the hand friction cone will be based on the hand force, which will guide the grasp quality measures using the grasp wrench space.

### 6.1.2 Grasp Quality Metrics with Negative Curvature

From an analytic perspective, differential geometry offers pertinent tools and concepts [48], although, as we will outline in the next section, objects are most often represented using triangular meshes and they, therefore, feature many local non-differentiable patches. Moreover, as we explained in the previous section, in practical scenarios, most representations rely on discrete models (e.g., discretized friction cones). Therefore it will be necessary to eventually reconcile the continuous models and representations with their discrete counterparts. For the time being we assume that the boundary of the object being grasped can be decomposed as a finite collection of surfaces and that grasping points are placed at differentiable points. That is to say that if $\mathbf{p} \in \mathbb{R}^3$ is a contact point on the surface of the object, then a neighborhood of $\mathbf{p}$ exists such that the points $\mathbf{x}$ on the surface

satisfy the equation $g(\mathbf{x}) = 0$, where $g : \mathbb{R}^3 \to \mathbb{R}$ is a suitable function twice differentiable in $\mathbf{p}$. Moreover, points inside the object are such that $g(\mathbf{x}) < 0$. To begin with, consider Figure 6.5. The three blue dots display three possible contact points on the surface of an object. According to our previous considerations, point number 1 and number 3 lie in locally convex and concave patches on the surface. Therefore point number 3 is more valuable than point number 1 in terms of its ability to resist an external wrench, e.g., a wrench due to an external lateral force. Point number 2 represents an intermediate situation, whereby an orthogonal force applied there would help resist forces pushing the object up, but not down.

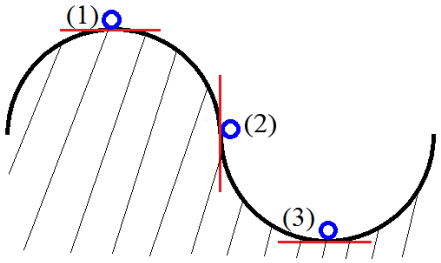



Figure 6.5: Three different conditions for contact points on a curved surface bounding the object from above.

Next, consider Figure 6.6. All cases depict contact points in locally convex patches on the surface. However, from left to right, the local curvature increases, and then, intuitively, one would prefer the rightmost contact point for its ability to restrain a disturbance wrench.

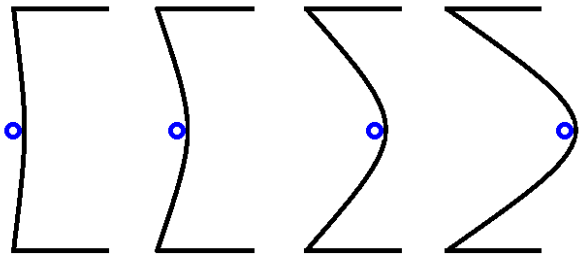


Figure 6.6: Contact points in blue on the black surface with different ratios of curvature.

Starting from the above observations, we then aim at the analytic definition of a quality metric incorporating them. The idea is to *dilate* the friction cone $F(\mathbf{p})$ at the contact point considering the local curvature on the surface at the contact point. According to our assumptions, the Hessian matrix of the function $g$ at $\mathbf{p}$ exists. Let $\mathbf{H}(\mathbf{p})$ be such matrix. The local concavity or convexity of $g$ in $\mathbf{p}$ can be determined from the properties of $\mathbf{H}(\mathbf{p})$. If $g$ is locally convex, then the friction cone remains unchanged, as suggested by the first contact point in Figure 6.5. If $g$ is instead locally concave, then the friction cone should be expanded, as for the third contact point in Figure 6.5. Such expansion should not be isotropic (i.e., uniform), but rather dependent on the local curvature. In other words,

the friction cone should be expanded more towards the directions in which $g$ grows more, and vice versa. Different surface curvature measures have been proposed in differential geometry. However, aiming at a definition that can be turned into an easily computed method, we instead estimate the curvature, and then the expansion of the friction code considering *directional slices* of function $g$. Let $\mathbf{v}$ be a unit length vector on the tangent plane $\mathcal{T}_\mathbf{p}$, and let $g_\mathbf{v}$ be the unidimensional function obtained, evaluating $g$ along the direction identified by $\mathbf{v}$. For a given small constant $h$, we therefore define

$$\nabla g'_\mathbf{v} = \frac{g'(\mathbf{x} + h\mathbf{v}) - g'(\mathbf{x})}{h} \tag{6.12}$$

where $g'$ is a well defined derivative of the single variable function $g$ evaluated along $\mathbf{v}$. From an algorithmic standpoint this can be achieved considering a few directions on the plane tangent to $g$ in $\mathbf{p}$ and then computing the $\nabla g'_\mathbf{v}$ along these directions. Directions can be uniformly spaced or randomly selected. The same idea can be applied when $g$ is locally concave, convex, or neither convex nor concave (e.g., a saddle point). In each case the friction cone $F(\mathbf{p})$ is expanded exclusively in the directions along which $\nabla g'_\mathbf{v}$ is positive. This idea can be formalized as follows. For a given force $\mathbf{f}$, let $\mathbf{v_f}$ be the unit length vector lying in $\mathcal{T}_\mathbf{p}$ and having the same direction defined by $f_2$ and $f_3$. In the frame $\mathbf{t}, \mathbf{u}, \mathbf{w}$, $\mathbf{v_f}$ is then[1]

$$\mathbf{v_f} = \begin{bmatrix} 0 & \dfrac{f_2}{\sqrt{f_2^2 + f_3^2}} & \dfrac{f_3}{\sqrt{f_2^2 + f_3^2}} \end{bmatrix}. \tag{6.13}$$

We then define the expanded friction cone, namely object friction cone, $F^O(\mathbf{p})$ as

$$F^O(\mathbf{p}) = \begin{cases} \mathbf{f} \mid \sqrt{f_2^2 + f_3^2} \leq \mu f_1 & \text{if } \nabla g'_{\mathbf{v_f}}(\mathbf{p}) \leq 0 \\ \mathbf{f} \mid \sqrt{f_2^2 + f_3^2} \leq (\mu + \zeta) f_1 & \text{if } \nabla g'_{\mathbf{v_f}}(\mathbf{p}) > 0 \end{cases} \tag{6.14}$$

Where $\zeta = k \nabla g'_{\mathbf{v_f}}(\mathbf{p})$ and $k$ is a fixed parameter $k > 0$. Note that in this definition we did not write the condition $f_1 \geq 0$ in the interest of space, but this should nevertheless be assumed.

The newly defined $F^O(\mathbf{p})$ formally captures the cases intuitively discussed in Figure 6.5 and 6.6, i.e., it expands $F(\mathbf{p})$ only along directions of negative curvature, and it moreover performs an anisotropic expansion, i.e., the cone is grown more in the directions of larger negative curvature. Note that from a geometric standpoint, $F^O(\mathbf{p})$ is still a cone, however, its base is no longer circular.

From a practical perspective, as we mentioned in the previous section, the friction cone $F(\mathbf{p})$ is most often represented by a regular pyramid. Moreover, as explained in the next section, $\nabla g'_{\mathbf{v}_f}$ is evaluated only along a finite number of directions. A convenient approach is to evaluate it only along directions orthogonal to the pyramid edges, and to then expand only the edges associated with directions revealing negative curvatures.

[1] Recall that the first component is along the orthogonal axis $\mathbf{t}$.

### 6.1.3 Combining the Hand Friction Cone and Object Friction Cone

In previous sections, we proposed 2 kinds of a modified friction cone; one taking into account the hand information and the other one considering the object local geometry. Furthermore, these two friction cones can be simply combined since that scaling of the friction cone, along its normal direction, is not in conflict with expanding the friction cone along the tangent direction. Therefore, we define a new combined friction cone $F^C(\mathbf{p})$ as

$$F^C(\mathbf{p}) = \begin{cases} \mathbf{f} \mid \sqrt{f_2^2 + f_3^2} \leq \mu f_H(\mathbf{p}) f_1 & \text{if } \nabla g'_{\mathbf{v_f}}(\mathbf{p}) \leq 0 \\ \mathbf{f} \mid \sqrt{f_2^2 + f_3^2} \leq (\mu + k\nabla g'_{\mathbf{v_f}}(\mathbf{p})) f_H(\mathbf{p}) f_1 & \text{if } \nabla g'_{\mathbf{v_f}}(\mathbf{p}) > 0 \end{cases} \tag{6.15}$$

This newly defined friction cone combines maximum hand force and object local geometry, which, will provide a tighter and more comprehensive grasp quality measure.

## 6.2 Experiments and Results

### 6.2.1 Preliminary Results Considering Hand Configuration

To illustrate the effect of using the hand friction cone instead of the unit friction cone, we first consider a simple planar grasp. In order to quantify the grasp, we use the quality measure formerly introduce, namely the Ferrari and Canny metric (indicated as $Q_{GWS}$ or *Quality*). The object to be grasped is a rectangle where the hand is constructed by a palm with two fingers, each with two joints. We control the hand to always make the same contact with respect to the object, while changing the hand configurations from fully bent to fully stretched. Figure 6.7 shows the simple plot for four sample cases, where, from left to right, the hand is stretched out, holding the rectangle object shown in cyan further away from the palm. The red ellipse represents the force ellipsoid and the green part of this ellipsoid is the intersection of the force ellipsoid with the friction cone. Notably, the resultant hand friction cone, shown in black, is shrinking as the hand is stretched out. Similarly the corresponding quality measure of $Q_{GWS}$, using the hand friction cone, is also decreasing. On the other hand, if we used the unit friction cone, $Q_{GWS}$ will remain constant for all four cases. Furthermore, Figure 6.8 captures the change of the scalar value (in red) used for the hand friction cone and the corresponding quality (in blue) with respect to the hand configuration from fully bent to fully stretched. Note that the curves are following a similar trend for the scalar and quality since the grasp wrench space used to calculate the quality is scaled entirely by the corresponding scalar. However, this phenomenon will only hold when the scalar for all fingers are the same.

Note that the Jacobian matrix for the hand configuration representing fully bent and fully stretched is singular, which is the case we want to avoid. Therefore, the experiments in this subsection only approached fully bent and fully stretched without actually taking these two situations into account.

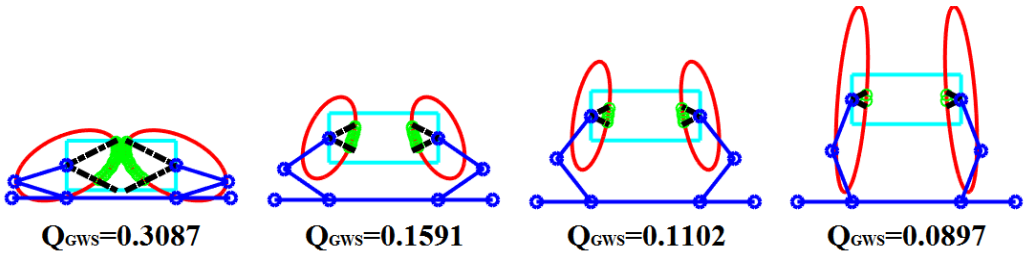



Figure 6.7: An example of a two finger hand (shown in blue) grasping a planar object (shown in cyan) at the same contact position with different hand configurations. The force ellipsoid is represented in red, the discretized points (shown in green) are the intersecting points between the force ellipsoid and the friction cone, and finally, the cone (shown in black) is the hand friction cone.

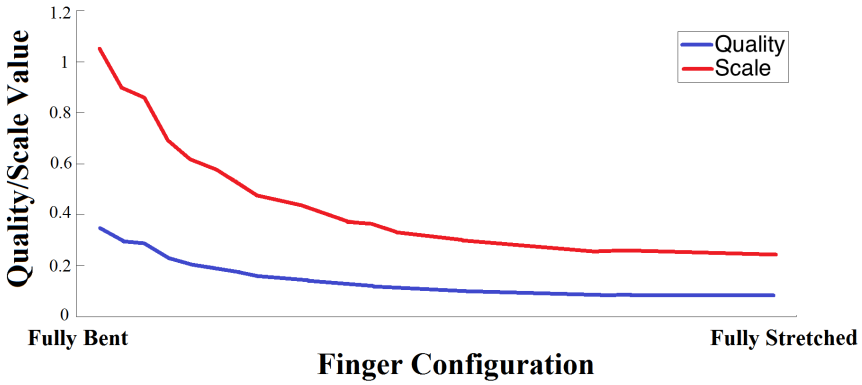



Figure 6.8: The scale of the friction and the corresponding quality measure with respect to the hand configuration. From left to right, the hand is changing from fully bent to fully stretched.

### 6.2.2 Preliminary Results Considering The Target Object

Revisiting Figures 6.5 and 6.6, our first experiment shows the effect of the revised definition of the friction cone on the value of the grasp metrics. In particular, we start considering the dependence on the ratio of negative curvature. In the following we consider two quality measures formerly discussed, namely the Ferrari and Canny metric (indicated as $Q_{GWS}$ or *Quality*) and the volume of the grasp wrench space (indicated as $V_{GWS}$). The reason to also consider this second metric will become clear later on in this section. We consider objects similar to Figure 6.2(a), where the curve is generated by the function $x_{-}(y) = -\alpha y^2 - c$ and $x_{+}(y) = \alpha y^2 + c$. By varying $y$ within $[-y_0, y_0]$, we obtain an object with the center of mass located at $(0, 0)$. We fix our grasp point $\mathbf{p}_1$ at $(-c, 0)$ and $\mathbf{p}_2$ $(c, 0)$. Therefore, we use the expanded friction cone $F^O(\mathbf{p_1})$ and $F^O(\mathbf{p_2})$ to calculate the grasp wrench space.

The top row in Figure 6.9 shows an example with $c = 2$, $k = 0.05$ and $\alpha = 0, 2, 4$. We approximate $\nabla g'_{\mathbf{v_f}}(\mathbf{p})$ with $h = 0.001$, so that $\nabla g'_{\mathbf{v_f}}(\mathbf{p}) = 2\alpha$. The Ferrari and Canny quality and grasp wrench space volume are indicated below each object. The examples show that in this case both metrics are increased as the curvature of the object increases and the friction cone is correspondingly

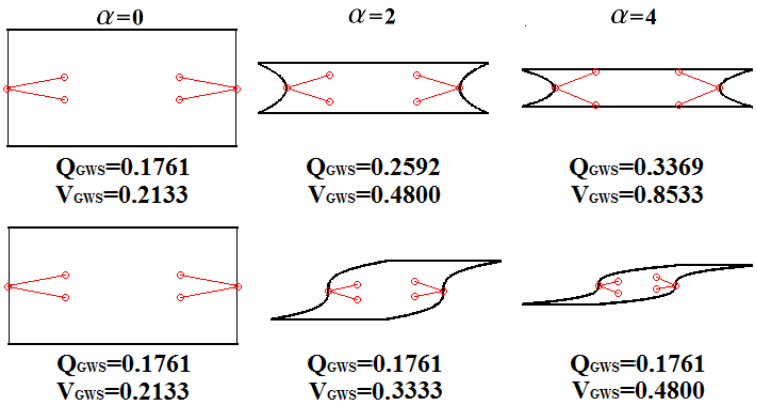



Figure 6.9: 2D example of objects with different contacting curvature. Red line shows the expanded friction cone at the contact point.

expanded. Figure 6.10 displays the relationship between the Ferrari and Canny grasp quality and the curvature.

Next, we consider a situation similar to point (2) in Figure 6.5. In this case the curve is generated by function $x_{-}(y) = \alpha y^3 - c$ and $x_{+}(y) = \alpha y^3 + c$, with $y$ within $[-y_0, y_0]$. The center of mass is still located at $(0, 0)$ and we fix our grasp points at $\mathbf{p}_1$ at $(-c, 0)$ and $\mathbf{p}_2$ $(c, 0)$. Then, we use the expanded friction cone $F^O(\mathbf{p_1})$ and $F^O(\mathbf{p_2})$ to calculate the grasp wrench space. The bottom row in Figure 6.9 shows some examples with with $c = 2$, $k = 33.33$ and $\alpha = 0, 2, 4$. We approximate $\nabla g'_{\mathbf{v_f}}(\mathbf{p})$ with $h = 0.001$, so that $\nabla g'_{\mathbf{v_f}}(\mathbf{p}) = 0.003\alpha$. As for the previous examples, the values for both the Ferrari and Canny and $V_{GWS}$ metrics are shown. These last examples show why we considered $V_{GWS}$, too. In this case the Ferrari and Canny metric does not change because it is defined by the worst case disturbance. In this case, the asymmetric expansion of the friction cone helps in resisting disturbance forces, but is neutral with respect to a disturbance torque rotating the object counter clockwise. This is consistent with the metric definition, but is, indeed, one of the weaknesses of this metric, i.e., it is defined by worst case scenarios that may hardly occur in practice. On the contrary, the metric considering the volume of the grasp wrench space grows, indicating that the grasp can resist a wider range of force disturbances. Note that this would not happen with the classic definition of friction cone.

### 6.2.3 Results on Combined Friction Cone

In section 6.2.1 and 6.2.2, we separately showed the impact of both modified versions of the friction cones. However, as mentioned earlier, these two ways of modifying the friction cone can be combined following equation 6.15. Figure 6.11 shows the effect of the combined friction cone. From left to right, the figure follows 6.7, where the hand is stretching out, and, from top to bottom, the figure is following Figure 6.9, where the negative curvature at each contact is increasing. However, we ignore the fact that for some cases the object is in collision with the hand, because we only want to illustrate the impact of the combined friction cone. As obtained directly from Figure 6.11, the friction cone is shrinking from left to right and is expanding from top to bottom, while the

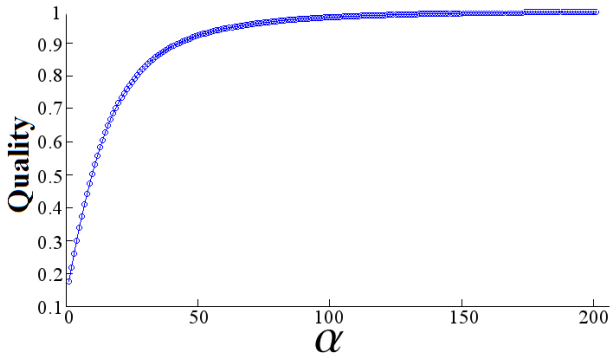



Figure 6.10: The relationship between grasp quality and the curvature of the surface for setup shown in Figure 6.9.

quality is decreasing from left to right and increasing from top to bottom.

### 6.2.4 Impact on Grasp Planning

In this section we discuss, in detail, the impact of using the hand friction cone instead of a unit friction cone on grasp planning. Note that the negative curvature feature can be used directly to construct a grasp planner, and will be introduced in chapter 7. Therefore in this section, we only focus on the effect of the hand.

The grasp planner we used to compare the effect of the hand friction cone and the unit friction cone is based on a random planner similar to the approach presented in GraspIt! [67]. The hand we used in this experiment is the Schunk Dextrous Hand (SDH) with its model shown in Figure 6.12. This hand is a fully actuated hand that has three fingers and each finger has two joints controlled independently using gear boxes. During the planning process, the SDH hand is randomly placed with respect to a target object. We apply a random configuration on the upper joint of each finger and close the hand to locate contact points. Once the contact points are determined, we calculate the Jacobian matrix at each contact point with respect to the base of the finger to determine the scale value used for the hand friction cone. Finally, we calculate the $Q_{GWS}$ quality value of the grasp using both the hand friction cone and the unit friction cone. Figure 6.13 shows the quality difference measure by $Q_{GWS}HFC - Q_{GWS}UFC$ for three different objects for over 400 runs. Refer to 6.14 for the models. The difference can be either positive or negative, indicating that some grasps are preferred by $Q_{GWS}$ using the hand friction cone and some are preferred by $Q_{GWS}$ using the unit friction cone.

Figure 6.14 shows the best grasp, i.e., the grasp that has the highest quality measure by both $Q_{GWS}$ using the hand friction cone, as in the top row, and by $Q_{GWS}$ using the unit friction cone, as in the bottom row, for 3 different objects. The major difference between the best grasps is that the grasps selected by $Q_{GWS}$, using the hand friction cone, are making contact closer to the bottom of the fingers, which generates a larger contact force. On the other hand, the best grasps selected by $Q_{GWS}$, using the unit friction cone, are only affected by the contact location regardless to the hand. This result matches the results presented in section 6.2.1.

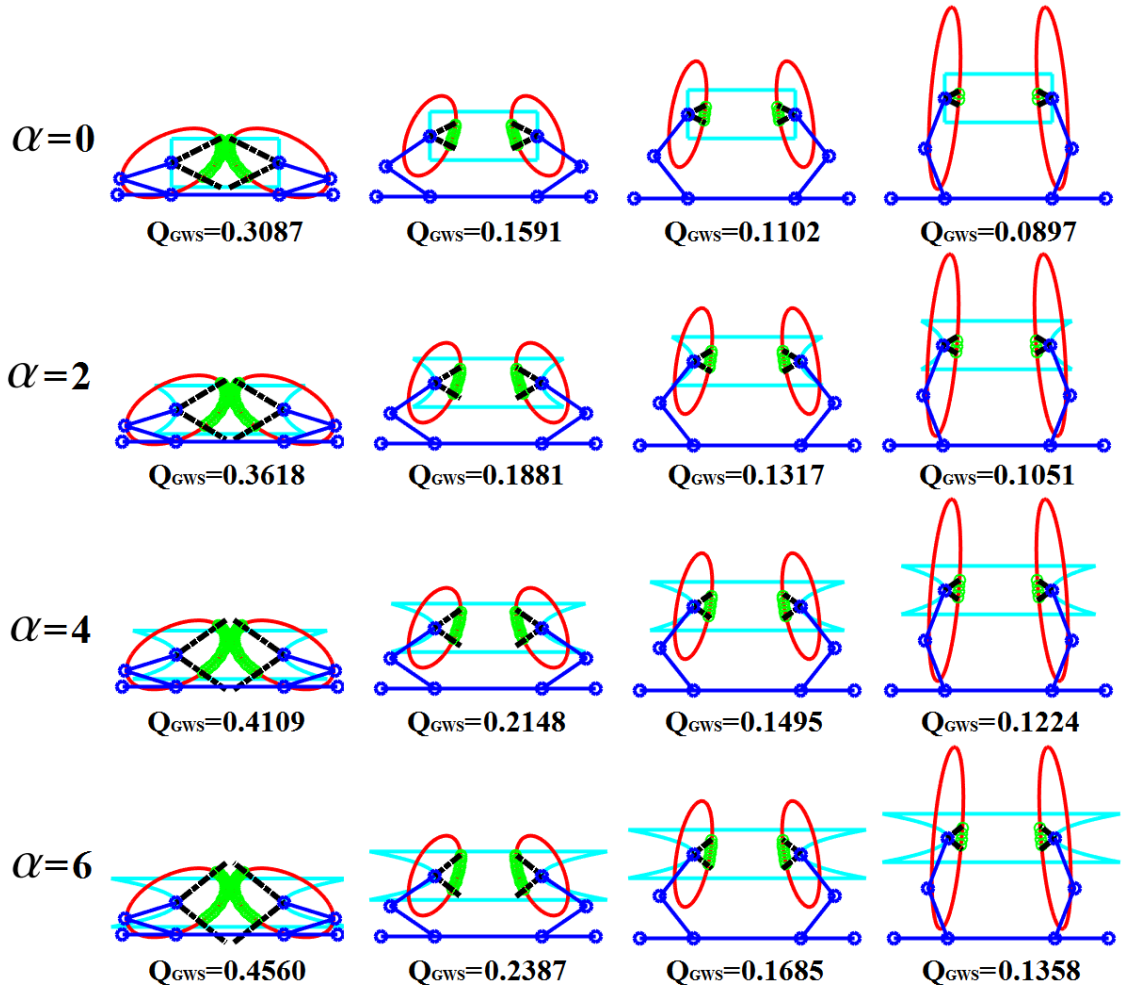



Figure 6.11: An example of a hand (in blue) with two fingers, each with two link grasping objects shown in cyan. The force ellipsoid is represented in red, the discretized points (shown in green) are the intersecting points between the force ellipsoid and the friction cone, and finally, the cone shown in black is the combined friction cone.

## 6.3 Conclusions

In this chapter, we addressed the effect of the hand configuration with respect to grasp quality. By limiting the maximum torque at each joint of the hand, we can use the Jacobian matrix, with regards to the hand configuration, to calculate the maximum force, which can be applied at each contact point. This maximum force guides the scale of the unit friction cone in order to connect the hand configuration with the grasp wrench space. We proposed the hand friction cone to represent the effect of hand configuration and used it to derive the quality of the grasp. We also considered the problem of grasping objects with negative curvature. More specifically, we showed that grasping at

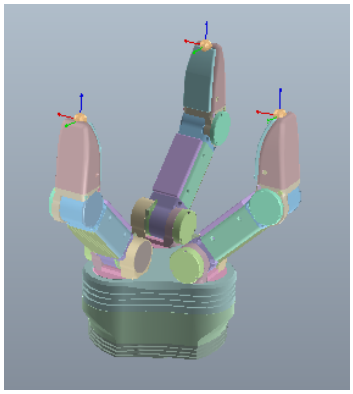


Figure 6.12: The model for Schunk Dextrous Hand.

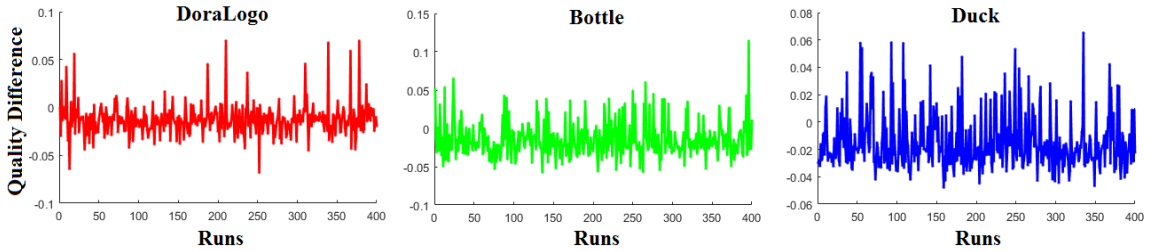



Figure 6.13: The quality difference between $Q_{GWS}$ for the hand friction cone and the unit friction cone for random grasps on 3 different objects: a DoraLogo, a bottle and a duck.

negative curvature points are, in general, more stable compared to other kinds of surface geometry. We formulated the effect and gain of negative curvature contact and connected this feature with grasp wrench space through a modified version of the friction cone.

Both the negative curvature feature on the target object and the hand configuration affect the representation of the friction cone. However, the negative curvature feature is modifying the friction cone by expanding the unit friction cone along its tangent direction while the hand configuration is scaling it along its normal direction. We noticed that it is possible to combine them. The resultant version of the combined friction cone will contain both the negative curvature feature and the hand configuration. Therefore, we proposed a modified grasp quality metric that accounts for finger forces and local curvatures and have shown that it overcomes many of the problems associated with commonly used metrics.

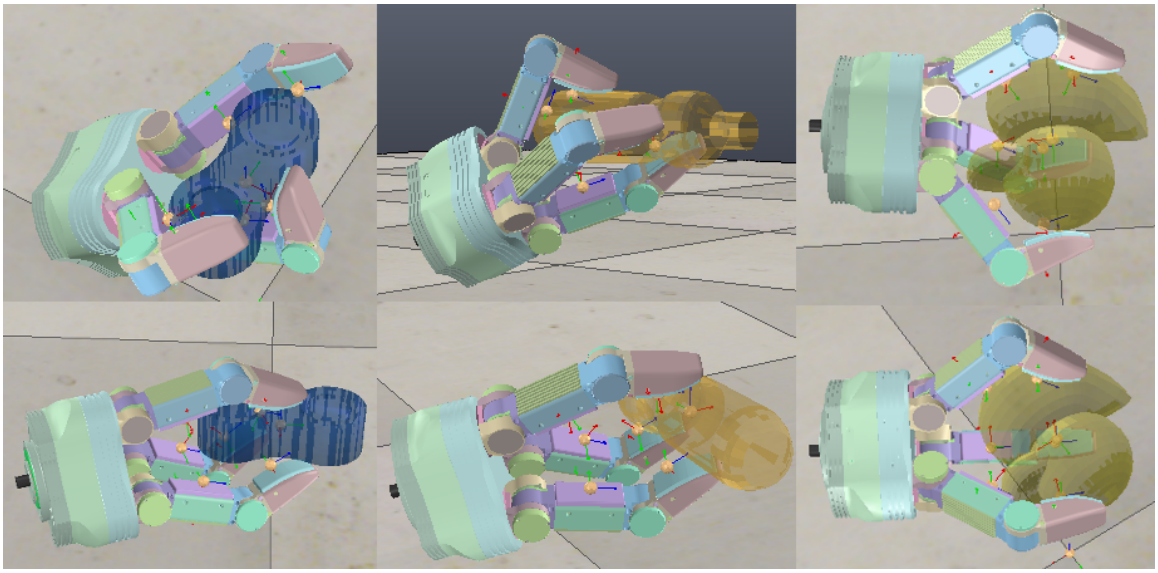


Figure 6.14: The best grasps selected by $Q_{GWS}$ using the hand friction cone, shown in the top row, and by $Q_{GWS}$ using the unit friction cone, shown in the bottom row, for 3 different objects: a DoraLogo, a bottle and a duck.

# Chapter 7

# Grasp Planner Development

In this chapter we propose three grasp planners we developed. In general, grasp planners can be classified into two categories; namely, model-less grasp planners and model-based grasp planners. Model-less grasp planners directly process captured data, i.e., from vision or touch, and locate features for the actuator to grasp. The model-based grasp planners take advantage of the object model and plan grasps upfront. The three planners we introduce in this chapter cover both types of planners: one model-less and two model-based. The model-less planner was based on the negative curvature feature introduced in chapter 6. One of the model-based planners also takes advantage of the negative curvature and calculates candidate grasps using the object geometry. These two planners will be introduced in section 7.1. The other model-based planner uses an optimization approach and moves the contact globally on the surface of the object to find local optimal grasps. This planner will be discussed in 7.2. The quality metric we used in this chapter is $Q_{GWS}$. However, this can be substituted with any metric, including the metrics we proposed in chapter 6.

## 7.1 Grasp Planner using Negative Curvature

In this section we present two grasp planners using the grasp quality metric we presented in section 6.1.2 to inform its search through the space of possible grasps. The model based planner follows the pipeline shown in Figure 2.2, where, in the precomputation step, all grasp candidates are computed with respect to the negative curvature features. The other model-less planner is developed according to the pipeline shown in Figure 2.4. This planner passes the sensorial data to a feature locator, which locates negative curvature features directly from the input point cloud. While the underlying principles are aiming to validate the algorithm on our existing robotic hardware, some implementation choices are made considering the hardware we will use. In particular, we focus on the multifingered Dora Hand produced by Dorabot, Inc., whose CAD rendering is shown in Figure 7.1. Note that while the fingers can be closed, they cannot be moved around the palm of the hand, and therefore the relative position of their first joint remains constant. The finger on the left of the figure is indicated as *finger 1* in the following.

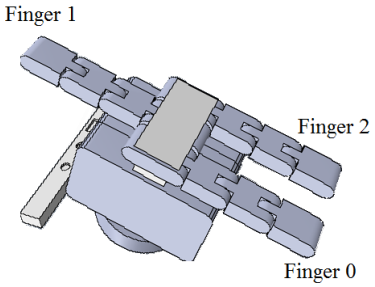



Figure 7.1: Dora Hand model.

### 7.1.1 Model Based Grasp Planning

As formerly noted in [24], in most scenarios, objects are represented using triangular meshes. This is often true when also considering objects with curved surfaces since they can be modeled as a large collection of small size triangles. In the following, we hypothesize that the objects' surface is represented by meshes of triangles, and it is, therefore, necessary to appropriately adapt the general concepts we developed, assuming differential surfaces.

The planner starts locating all edges whose adjacent faces form a concave region. Figure 7.2 shows a side view of two adjacent faces forming in a convex (left) and a concave (right) region. We can identify whether the region is convex or concave by calculating the dot product of two vectors. The first vector, $\overrightarrow{N}$, is the sum of the normal vectors of these two adjacent faces, $\overrightarrow{N_1}$ and $\overrightarrow{N_2}$. The other vector, $\overrightarrow{V'}$ is formed by the midpoint point of the vertices of these two adjacent faces that are not on the common edge, $V_1$ and $V_1'$, and one of the vertices on the common edge, $V_0$ or $V_2$. Note that in our case of triangular mesh representation, the normal vector of each face is always pointing towards the outside of the object. If the region formed by two adjacent faces is convex, the midpoint point of $V_1$ and $V_1'$ must be inside of the object, and the dot product of $\overrightarrow{N}$ and $\overrightarrow{V'}$ will be negative. On the other other hand, if the dot product is positive, then the region is concave. A special case arises when $\overrightarrow{N_1} = \overrightarrow{N_2}$, i.e., the two adjacent faces are on the same plane. In this case, the dot product will be 0 which also indicates these two faces forms a region that is neither strictly convex nor concave. Figure 7.3 shows some examples, with the common edges of two adjacent faces forming a concave region shown in red, while the normals of the two adjacent faces are shown in green and blue.

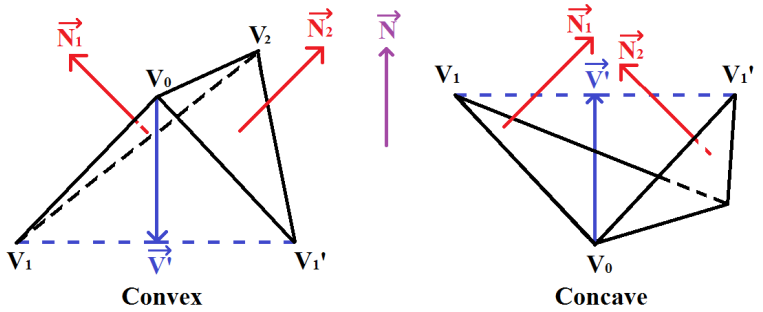



Figure 7.2: Side view of two adjacent faces forming convex and concave region.

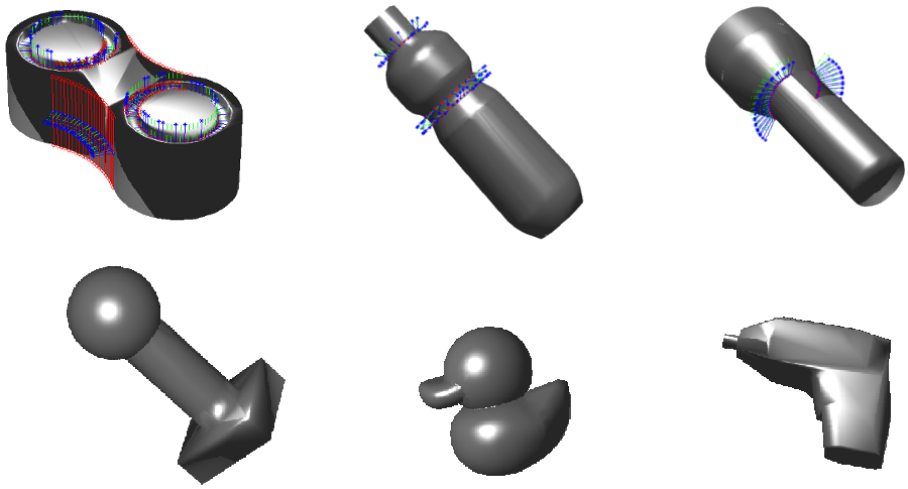


Figure 7.3: Examples of the objects we used in simulation. The top row also shows the negative curvature attached to edges in red lines. Blue and green lines are the normal vector of the two adjacent faces connected by the negative curve edge.

The planner we developed is inspired by the approach presented in GraspIt! [67], i.e., it randomly generates hand positions around the object to be grasped and then determines the contact points by simulating finger closure. Eventually, the grasp with the highest value for the grasping metric is returned. This idea is adapted to the hand we consider as follows. First, a point of negative curvature is determined on the triangular mesh. A hand configuration is then generated by approaching the fingertip of finger 1 towards the negative curvature point. We here refer to the fingertip as the center of the top phalanx. If both sides of finger 1 make contact with the object and the face normal at each contact forms a concave region, we move on to the next step. Otherwise, this negative curvature point is rejected. Note that when we first locate a point of negative curvature, the size of the negative curvature is relevant to the face size. Since the size of each face varies, not all negative curvature points will generate a valid grasping point on the negative curvature. Next, the other two contact points are determined by projecting where the other two fingers will make contact when closed. At this point the quality of the grasp configuration is determined. The process is then repeated multiple times, and at each iteration a new point with negative curvature is randomly chosen. Figure 7.4 shows some example grasps computed by the planner we just described. Note that the finger 1 always makes contact on the surface on negative curvature.

After obtaining grasp candidates, they are measured by the grasp quality metric $Q_{GWS}$. We then construct a database with grasps that have a quality value higher than a predefined threshold. We set the threshold to be 80% of the highest quality calculated for this target object. The database we constructed stores three sets of information: object ID, the relative pose of the hand with respect to the object and hand configuration. The object ID is used to determine whether the grasp is matched to the detected object. The relative pose of the hand, with respect to the object, is used to determine the grasp and check if it is feasible to grasp from. The hand configuration stores, in detail, how the joints of the hand is positioned when the object is grasped. Both relative pose and

hand configuration are useful to compute a pregrasp pose.

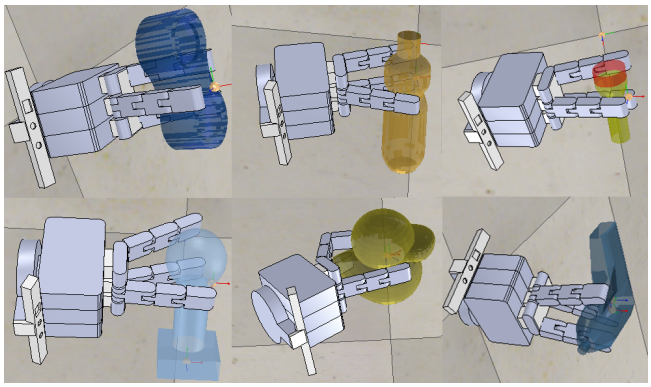


Figure 7.4: Example output from the model-based grasp planner for six different objects.

### Perception Pipeline Using a Model-less Approach

For a typical model-based grasp planner, precise information, including an object's ID and pose, are given to the database. This information is usually handled by human knowledge instead of sensor data captured in real-time. However, in order to develop an autonomous framework, we use a model-less approach based on vision to obtain the object's ID and pose. This vision pipeline containing object segmentation, recognition and pose estimation, is used in order to identify the object to be grasped and determine the pose of the object. Segmentation is done by Locally Convex Connected Patches (LCCP) algorithm described in [20]. This algorithm decomposes the scene into a voxel grid based adjacency-graph of surface patches. A novel combination of simple criteria, which operate on the local geometry of these patches, is then used to classify edges in the graph as convex or concave. As a result, the graph is segmented into locally convex connected subgraphs, which represent object parts with high accuracy. A descriptor for 3D point cloud data, called the Viewpoint Feature Histogram (VFH) [84], was used for object recognition and pose estimation. VFH encodes geometry and viewpoint, which is used as a distinctive signature that allows simultaneous recognition of the object and its pose. However, the pose estimated by this algorithm is not very accurate. Another algorithm, called Iterative Closest Point (ICP) algorithm [9], is then used to produce an accurate pose. ICP matches the segmented point cloud with a prestored point cloud representing the object we recognized using VFH. This algorithm works at the scale of points and tries to minimize the sum distance between all matching point pairs. Due to these reasons, ICP is known as one of most accurate algorithms for pose estimation. Once these visions related processes are finished, we will know the model of the object along with its pose in the environment. Thus, we can query grasps with high quality measure from the database we previously constructed to execute on real robots.

## 7.1.2 Model-less Grasp Planner

In general, negative curvature is a valuable feature that can be determined directly using point clouds. We hereby developed an algorithm which takes a point cloud captured in real time

as an input and outputs a pose array listing of all possible fingertip poses at a point with negative curvature.

The input point cloud is raw data, as shown in 7.5(a), which is too noisy to locate negative curvatures. Therefore we first pass the point cloud through some pre-process steps in the following order using Point Cloud Library (PCL) [85]:

- Remove error data, which are either NaN or having 0 depth value. An example of this step is shown in Figure 7.5(b).
- Remove outliers caused by noisy measurements, based on the computation of the distribution of a point to its neighbor's distances. The output of an example of this step is shown in Figure 7.5(b). [1]
- Smooth and resample noisy data using Moving Least Squares(MLS) surface reconstruction method. Figure 7.5(c) shows an example of the smoothing step. [2]

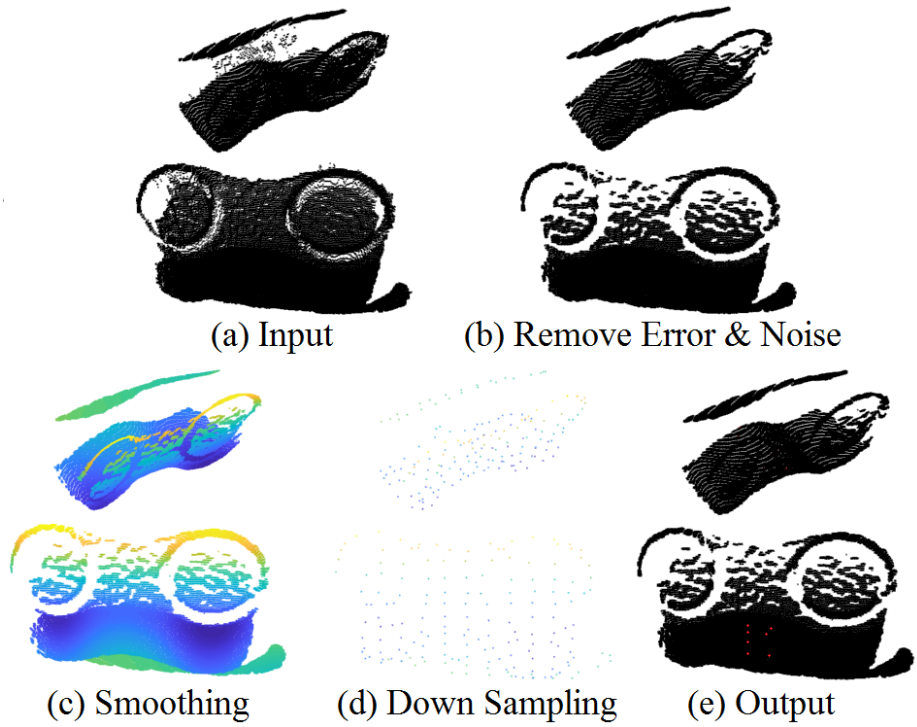



Figure 7.5: An example of step by step point cloud processing demonstration. The input point cloud is shown in (a). (b) shows the filtered point cloud after removing error and noise data. A smoothed point cloud is then shown in (c). (d) shows the resultant down-sampled point cloud. (e) shows the output point cloud with negative curvature points shown in red.

[1] Refer to http://pointclouds.org/documentation/tutorials/statistical_outlier.php for a detailed instruction.
[2] Refer to http://pointclouds.org/documentation/tutorials/resampling.php for a detailed instruction.

After these preprocessing steps, we obtain a cleaner and less noisy point cloud for us to start with, namely $P_0$. A step by step point cloud demonstration is shown in Figure 7.5, where the preprocessing steps are shown from subfigures (a) to (c).

After preprocessing, an algorithm focused on points instead of the entire point cloud is developed in order to locate negative curvature points. Furthermore, a hand configuration that makes contact between one of the fingertips and a negative curvature point is calculated considering the model of the hand. This algorithm follows the steps below:

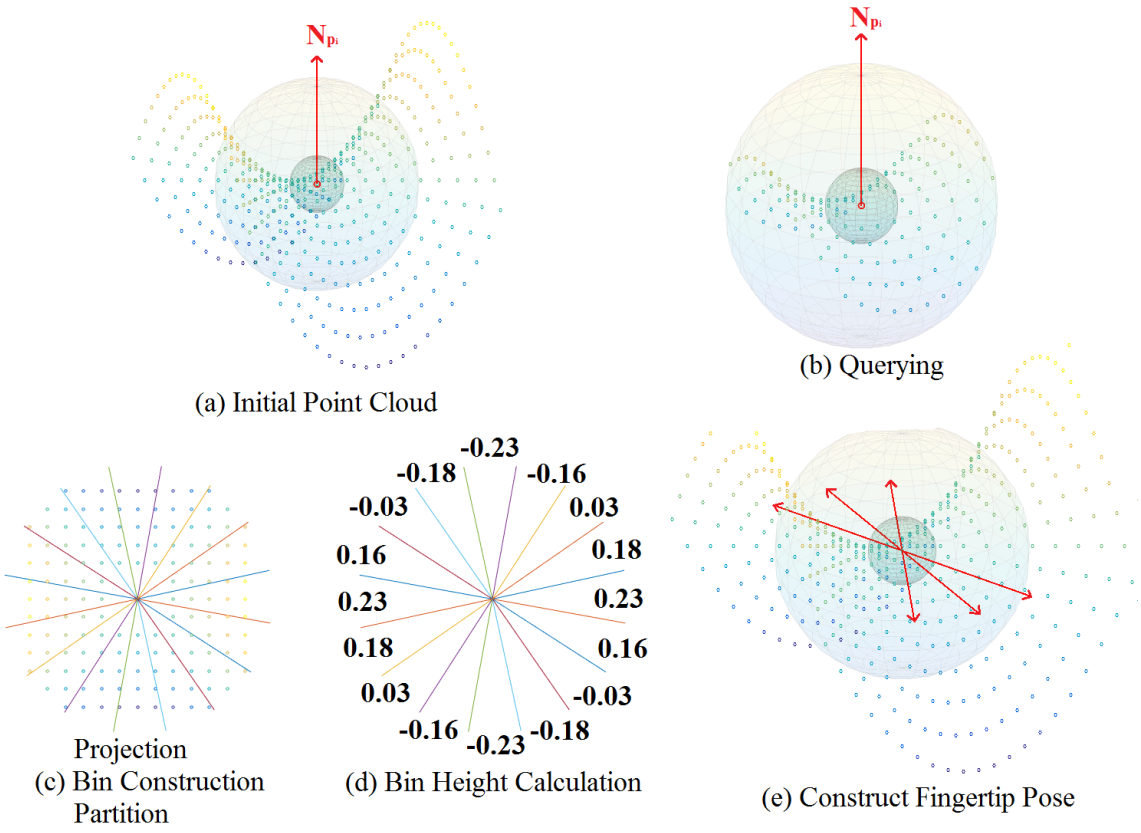



Figure 7.6: An example of step by step demonstration of our algorithm that constructs fingertip pose from the initial point cloud. The initial point cloud is shown in (a). (b) shows the result after querying. (c) shows the partitioned point cloud and the constructed bin separated by colored lines. The corresponding partition of the points into the bins are also shown. (d) shows the height calculation of each bin with positive values representing a negative curvature along the bin direction and negative values representing a positive curvature along the bin direction. The output fingertip pose (in red arrow) is shown in (e).

**Normal Calculation**: Calculate the surface normal for all points in $P_0$ with the viewing point set to the camera. The calculation of surface normal with respect to a point is calculated using Principal Component Analysis (PCA) of a covariance matrix created from the nearest neighbors of

this point. The nearest neighbors are set within a range defined by the user. Note that since we are dealing with a point cloud, we do not need to consider the special case where points on the boundary does not have normal.[3]

**Down Sampling**: Down sample the point cloud using a voxelized grid approach to obtain a subset $P_d$ with a limited amount of points. The outcome of this step is shown in 7.5(d). [4] This down sampling step is necessary because processing all points within the point cloud to locate negative curvature is very inefficient. A region of the concave feature usually contains a set of points, and any point from this set is sufficient to represent the entire set.

**Querying**: For all points $p_i$ in $P_d$, query all points $p_{ij}$ in $P_0$ such that $r_1 \leq \|p_i - p_{ij}\|_2 \leq r_2$, i.e., querying point from $P_0$ within a spherical ring for all points in set $P_d$. $r_1$ and $r_2$ are constants relevant to the size of the fingertip. This step is shown in Figure 7.6(b). The reason to set a lower bound is that if the point very close to $p_i$ is some noisy data, then it will have a huge impact in the partition step, which will produce wrong results. Also, time will be saved by processing less points. We set an upper bound with regards to the fingertip size because we are only looking for local negative curvature points, which are not affected by the points further away.

**Projection**: Project $p_{ij}$ for all $j$ to the tangent plane of the normal of $p_i$. Therefore, the tangent plane projection will be bounded by a circle with radius $r_2$, shown in Figure 7.6(c).

**Bin Construction**: Equally divide the tangent plane determined by $p_i$ into $4s$ bin, namely $b_k$, $k \in [0, 4s]$ shown in Figure 7.6(c), where $s$ is a constant integer. We chose the number of bins to be a multiple of 4 in order to simplify later steps.

**Partition**: We now project the normalized vector $\frac{\overrightarrow{p_{ij}-p_i}}{\|p_{ij}-p_i\|_2}$ for all $j$ on the normal of $p_i$. Assume $\overrightarrow{d_0}$ is an arbitrary direction within the tangent plane determined by the normal of $p_i$. The bin index $k$ is determined by the angle $\alpha$ between $\overrightarrow{p_{ij} - p_i}$ and $\overrightarrow{d_0}$, where $k = \lfloor \frac{2\alpha s}{\pi} \rfloor$. The projection length $pl_{ij}$, as defined in Lemma 9 in chapter 4, is calculated with respect to $\frac{\overrightarrow{p_{ij}-p_i}}{\|p_{ij}-p_i\|_2}$. This step is also shown in Figure 7.6(c). The reason we used normalized vector $\frac{\overrightarrow{p_{ij}-p_i}}{\|p_{ij}-p_i\|_2}$ instead of $\overrightarrow{p_{ij} - p_i}$ is because we are focused on the curvature with regards to the size of bins rather than each points.

**Bin Height Calculation**: For each bin $b_k$, calculate the average value of the projected length $l_k$, where $l_k = Avg(\{pl_{ij} | pl_{ij} \in b_k\})$. This step is shown in Figure 7.6(d).

**Locate Points with Negative Curvature Feature**: If $l_l < l_k < l_u$, $l_l < l_{k+2s} < l_u$, $l_{k+s} < 0$ and $l_{k+3s} < 0$, where $l_l$ and $l_u$ are the lower and upper threshold that we chose to set for the curvature value and $0 \leq l_l < l_u \leq 1$, then we define this point as a point that has the negative curvature feature along the direction defined by $l_k$ and $l_{k+s}$. Otherwise, this point is not a point with the negative curvature feature along the direction defined by $l_k$ and $l_{k+s}$. In the case of grasp planning, our goal is to gain support for each finger. This requires two conditions. The first one is the support must be along both sides of the finger and the second one is we must be able to place a finger at this point. In order to satisfy both conditions, we need to look at two directions orthogonal to each other in the tangent plane, where along one direction it forms a negative curvature and along its orthogonal direction, the curvature is non negative.

[3]Refer to http://pointclouds.org/documentation/tutorials/normal_estimation.php for a detailed instruction.

[4]Refer to http://pointclouds.org/documentation/tutorials/voxel_grid.php for a detailed instruction.

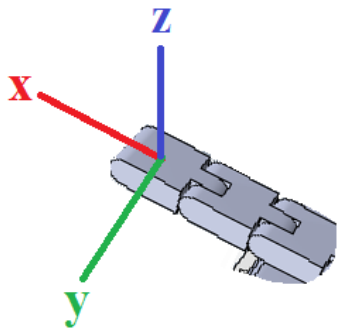



Figure 7.7: Coordinate system at the fingertip.

**Construct Fingertip Pose**: After determining the direction of a negative curvature point, we can align one finger of our gripper. Figure 7.7 shows the coordinate system at the fingertip. The output poses of the fingertip will then have the translation equal to the position of $p_i$; the z-axis of orientation aligned with the normal of $p_i$, facing towards the object; the y-axis aligned with the direction defined by $b_k$ and $b_{k+2s}$ and x-axis aligned with $b_{k+s}$ and $b_{k+3s}$. An example is shown in Figure 7.6(e). Note that for every negative curvature point, the fingertip pose is generated in pairs, where the y-axis of the fingertip can be aligned with both directions defined by $b_k$ and $b_{k+2s}$.

**Construct Hand Pose**: After obtaining the fingertip pose, the next problem is how to set the hand pose. This step is purely based on the robotic hand being used. For our case, DoraHand is used to perform in real robot experiment. The hand pose, with regard to this hand, is set to a pose facing towards the object and placed with an offset equal to the gripper length along the direction determined by $b_{k+s}$ and $b_{k+3s}$, with respect to the negative curvature point. The left column of Figure 7.8 shows some examples of where the hand is placed. In general, as long as one of the fingertips can be place on the negative curvature point, it would be a valid hand pose to set.

### Algorithm Sketch

Putting all the previous steps together we obtain algorithm 4. The input of the algorithm is the raw point cloud $P_{raw}$ and parameters $r_1$, $r_2$, $s$, $l_l$ and $l_u$, defined in the previous steps. The output of the algorithm is the pose array that stores the end-effector pose with respect to the hand. The stored information guarantees that one of the fingertips of the hand will be in contact with a point having a negative curvature feature. Line 3 to 5 are the preprocessing steps on the raw point cloud. Line 6 to 21 are the main steps to locate a point with negative curvature. Line 22 to 24 calculates the corresponding end-effector pose with respect to each negative curvature point. The time complexity of the algorithm is directly related to the size of the raw point cloud. The size of $P_d$ and the size of $P'_i$ is comparably smaller than the size of the raw point cloud.

**Algorithm 4** Negative curvature feature locator algorithm

**Data** : $P_{raw}$, $r_1$, $r_2$, $s$, $l_l$, $l_u$
**Result** : $PA_{eff}$
$P_{filtered} \leftarrow RemoveErrorData(P_{raw})$
$P_{filtered} \leftarrow RemoveOutlier(P_{filtered})$
$P_0 \leftarrow Smooth(P_{filtered})$
$N \leftarrow NormalCalculation(P_0)$
$P_d \leftarrow DownSample(P_0)$
**for** $p_i \in P_d$ **do**
  $P_i \leftarrow \{p_{ij} | r_1 \leq \|p_i - p_{ij}\|_2 \leq r_2\}$
  $P'_i \leftarrow ProjectAlongN_i(P_i)$
  $b \leftarrow ConstructBin(4s)$
  Set $\overrightarrow{d_0}$
  **for** $p_{ij} \in P'_i$ **do**
    $\alpha \leftarrow GetAngle(\overrightarrow{p_{ij} - p_i}, \overrightarrow{d_0})$
    $k \leftarrow \lfloor \frac{2\alpha s}{\pi} \rfloor$
    $pl_{ij} \leftarrow ProjectionLength(\frac{\overrightarrow{p_{ij} - p_i}}{\|p_{ij} - p_i\|_2})$
    Partition $pl_{ij}$ to $b_k$
  **for** $k \leftarrow 1$ to 4s **do**
    $l_k \leftarrow Avg(\{pl_{ij} | pl_{ij} \in b_k\})$
  **for** $k \leftarrow 1$ to s **do**
    **if** $l_l < l_k < l_u$ and $l_l < l_{k+2s} < l_u$ and $l_{k+s} < 0$ and $l_{k+3s} < 0$ **then**
      $p_{finger} \leftarrow ConstructFingerPose(p_i)$
      $p_{hand} \leftarrow ConstructHandPose(p_{finger})$
      $PA_{eff} \leftarrow PushBack(p_{hand})$

Figure 7.8 shows the outcome of our planner. The left side shows the real time captured point cloud in Rviz[5]. The red arrows are the pose array containing feasible poses for the hand. The axis in RGB is the one we selected to perform grasping. The right side of the figure shows the actual grasp executed on the robot. The run time is limited by the amount of points in the input cloud. Although we had embedded a down-sampling technique to decrease the number of points to be processed, when a huge point cloud with rich environmental information is fed into the system, it will still be time consuming. A possible way to improve the calculation speed for this case would be designing a point cloud filter based on the environment setup and remove as much environment information as possible, keeping just the object's point cloud. Environment, for most cases, are modelled upfront for collision checking, ensuring the safety and correctness of the real robot actuation. This information can be used directly to construct the point cloud filter.

[5]Refer to http://wiki.ros.org/rviz for more information

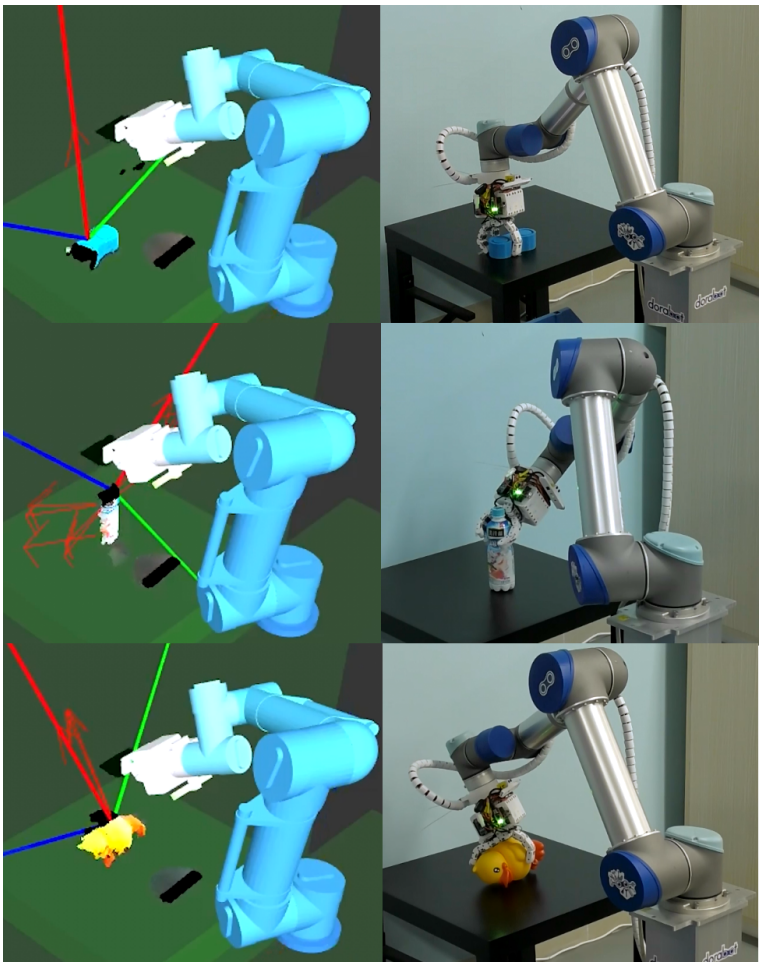


Figure 7.8: Input point clouds captured in real time showing on the left and the resultant grasp of the real robot shown in the right for three objects.

### 7.1.3 Negative Curvature Planner Comparison

The model-based grasp planner relies on a vision pipeline to recognize the object and obtain its pose. This is more time consuming compared to the model-less grasp planner which only needs to locate negative curvature points for a given point cloud. Also, the model-based grasp planner faces the limitation of the set of objects it is capable of dealing with. If the object is unknown, this planner will not be able to recognize the object and obtain grasp candidates. On the other hand, the model-less grasp planner can process any object as long as a negative curvature feature can be seen. Preprocesses such as filtering environment and segmentation will help to speed up the whole pipeline, although it is not necessary and we would still get valid results. The downside of this model-less planner is that we would not be able to guarantee a high quality grasp. One way to improve would be projecting the point to the plane determined by the negative curvature point's normal direction and negative curvature direction. Then we can calculate the quality for this planar

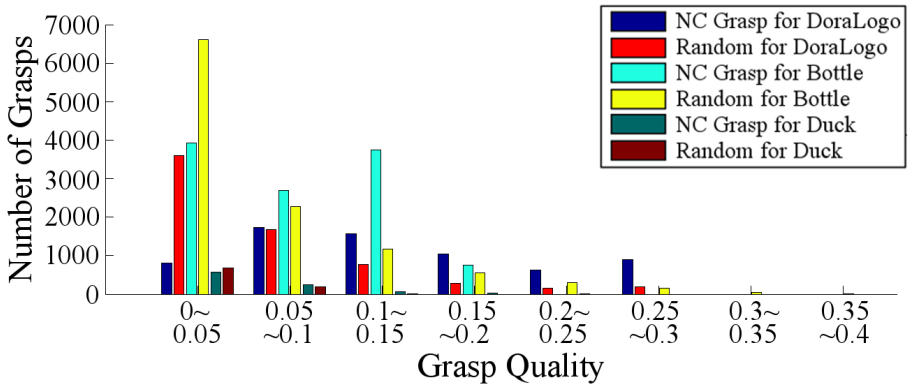



Figure 7.9: Grasp quality comparison with same amount of grasps for 3 different objects.

grasp. Although grasping at negative curvature points would be beneficial for the quality, it is not possible for all objects with this feature to ensure a force closure grasp. Consider the cap of a bottle that forms a negative curve. Even grasping at this featured point is totally valid, but since it is too off to one side of the object, a huge torque will occur, which the hand will be unable to resist.

In conclusion, if we are constrained with only a limited amount of objects, then the model-based grasp planner would be more stable. However, the model-less grasp planner is more adaptive and would be a good start when facing more general object setup.

### 7.1.4 Experiments and Results

In this section we present some results illustrating the metric we proposed in chapter 6. Both planners were also validated on a commercially available robot and the comparison result between these two planners will also be given.

#### 7.1.4.1 Grasp Planning Comparison

To show how the grasp planner we developed takes advantage of negative curvatures, we compare it with a baseline random grasp planner, as it is often done in literature [58, 75]. In the following, our planner is referred to as *NC* planner, where *NC* stands for negative curvature. The random grasp planner was implemented by simply giving a random pose of the object with respect to the hand, then closing the hand based on its hardware structure and measure the quality of the grasp. This is basically the GraspIt! approach.

We chose the 3 objects shown in Figure 7.12, i.e. a duck, a bottle and an eight-shaped object representing the logo of Dorabot, Inc., (referred to as DoraLogo in the following). Figure 7.9 shows the comparison results between our grasp planner and the random grasp planner with both of them generating the same amount of grasps. Negative curvatures on given objects are limited to a fix amount, so that grasp candidates are also limited for the NC grasp planner. In order to show a more meaningful comparison, we generate the same amount of grasps using a random grasp planner. The time spent to generate grasps using our NC grasp planner and the random planner are shown in Table 7.1. Clearly, our NC grasp planner is more efficient in generating grasps with higher quality.

| Object | DoraLogo | | Bottle | | Duck | |
|---|---|---|---|---|---|---|
| Grasp planner | NC | Rand | NC | Rand | NC | Rand |
| Avg Time(s) | 4.974 | 8.735 | 0.832 | 5.523 | 2.535 | 13.715 |

Table 7.1: Comparison result for time to generate grasps in the simulation.

For fairness, it is also important to notice that the NC grasp planner has some limitations. The first drawback is that our NC grasp planner offers no advantages if the object to be grasped has no negative curvature areas. Secondly, due to the limited amount of finger placement on negative curvatures, the database may only contains grasps from certain directions. However, this might not be enough depending on the placement of the object. In addition, being biased towards areas of negative curvature might end up in low quality grasps if the center of mass of the object is far from these areas.

### 7.1.5 Real Robot Experiment

#### 7.1.5.1 Model-Based Grasp Planner

We conclude the validation of the method we proposed using a Dorabot mobile manipulator (see Figure 7.10). This robot features an omni-directional mobile base with 360 degree coverage by a lidar sensor, and it includes a lifter, a UR5 robot arm, and a reconfigurable dexterous robot hand with an eye-in-hand RGBD vision sensor. The platform can be controlled using ROS. The hand is designed in a modular fashion, and all flanges in the fingers are the same module. Each joint may be under actuated or not. Each finger can have any number of phalanges, and a hand can have any number of fingers. Every finger can bend in any direction, and all phalanges are equipped with tactile sensors and a joint angle sensor. The hand can then function in multiple ways, from a parallel jaw gripper to an anthropomorphic mode. The software pipeline is shown in Figure 7.11, and our grasp planar results intervenes in the third step—retrieving the grasp from the database. In this section, we show the real robot performance of grasping objects with and without negative curvature. The objects we used to perform our test are those shown in Figure 7.12.

We ran 10 grasp tests on each object with the database generated by the NC grasp planner and a random grasp planner. Successful runs are determined by fully grasping the object from the bin, picking it up, and dropping the object at a predefined location, while failure is defined as not being able to complete the whole process. Failure is typically caused by the object sliding during motion or being unable to determine an appropriate grasp configuration. Figure 7.13 shows successful grasps for each object with our NC grasp planner whereas Table 7.2 summarizes the overall result.

Figure 7.10: Robot hardware.

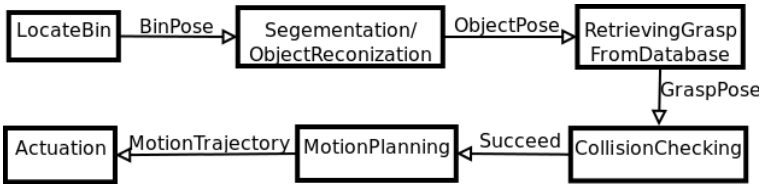



Figure 7.11: Software pipeline.

Figure 7.13: Grasp example with real robot.

The NC grasp planner clearly outperforms the grasps from the baseline random grasp planner. Since sliding may occur while the object is moving, grasping on a negative curve can better restrain the object, so we get fewer failures caused by sliding.

Figure 7.12: Objects used in real robot example.

| Object | DoraLogo | | Bottle | | Duck | |
|---|---|---|---|---|---|---|
| Grasp planner | NC | Rand | NC | Rand | NC | Rand |
| Success | 8 | 4 | 6 | 5 | 7 | 4 |
| Success (sliding) | 1 | 1 | 2 | 1 | 0 | 0 |
| Fail (sliding) | 1 | 2 | 0 | 1 | 3 | 4 |
| Fail (motion planning) | 0 | 3 | 2 | 3 | 0 | 2 |

Table 7.2: Comparison results from the real robot experiment.

#### 7.1.5.2 Model-less Grasp Planner

The setup of the real robot experiment for the model-less grasp planner is simpler compared to the experiment setup for the model-based grasp planner. We moved the robot down from its mobile base and placed it directly on the ground as shown in Figure 7.14. The objects were placed on a table top instead of a bin. Such setup can bypass the locate bin step. Also, since we are directly processing the point cloud, the segmentation and object recognition step is also redundant. Therefore, the pipeline for this setup is reduced to the diagram shown in Figure 7.15.

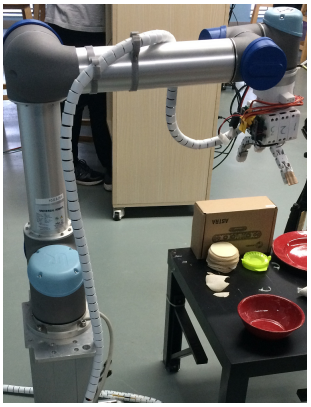


Figure 7.14: Robot hardware

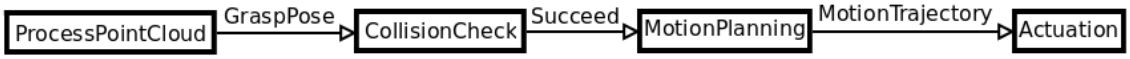



Figure 7.15: Software pipeline.

To address the adaptivity of the model-less planner, we perform the same test, i.e. run 10 grasp tests on each object, on 6 different objects. Three of them were used in the previous tests shown in 7.12, and three of them were newly introduced, shown in Figure 7.16. One thing needing clarification is that in these experiments we control the relative pose of the object against the camera in order to guarantee that the input point cloud contains negative curvature. The results of the test are shown in table 7.3. Table 7.3 has one more entry: fail with grasp planning. This item stands

Figure 7.16: Objects used in real robot example.

| Object | DoraLogo | Bottle | Duck | Goggle Box | Ball Gripper | Vase |
|---|---|---|---|---|---|---|
| Success | 9 | 7 | 5 | 6 | 6 | 5 |
| Success (sliding) | 0 | 1 | 0 | 2 | 1 | 1 |
| Fail (sliding) | 1 | 1 | 2 | 2 | 2 | 3 |
| Fail (motion planning) | 0 | 0 | 2 | 0 | 0 | 1 |
| Fail (grasp planning) | 0 | 1 | 1 | 0 | 1 | 0 |

Table 7.3: Test results of real robot experiment for the model-less grasp planner.

for when the planner failed to locate a negative curve feature directly from the input point cloud. Therefore it fails with grasp planning.

As shown in table 7.3, the success rate for the model-less planner is comparable to the results using the database, shown in table 7.2. It is also shown that the model-less performs well with unknown objects. However, there are still some limitations using the model-less planner. The most significant one is the viewing angle. Since this method grasps objects based only on the point cloud, the viewing angle of the point cloud will have a major impact. If the input point cloud is captured from an angle that does not contain a negative curvature, then this method will definitely fail. On the other hand, if we use the model-based grasp planner, although the negative feature is not seen, it still works as long as the object is recognized.

## 7.2 Global Grasp Planning Using Triangular Meshes

In this section, we will introduce the grasp planner we built aiming to find local optimal grasps over the entire object. We assume that $\mathcal{B}$ is represented using a triangular mesh. This assumption is consistent with current practices in CAD software [24]. We will show how the planner is capable of exploring the whole object and how much we can gain compared to the base line random grasp method. A variation of this planner is also proposed to consider hand structure. These two versions share the same framework where the first one optimizes over the contact point and the other one

optimizes over the hand configuration.

### 7.2.1 Continuity

The objective function for grasp planner problems is the quality of the grasp. The variable that controls the optimization are the contact points. Although the quality measure we use here is computed through the convex hull, the quality Q is measured by a point on the convex hull, i.e., the closest point on the convex hull to the origin. This point is a linear combination of the approximated friction cone related to each contact. The reason gradient based approaches work in this case, is that the surface to place contact on is continuous, such that the gradient can be found. However, for the triangular mesh representation, the edge intersecting two triangles is not continuous. In this case, a strictly gradient based method will not fit if we want to optimize over all surfaces, but a sample based objective driven method will have the ability to perform such process, i.e., sample the points around current contacts, the one that returns the best quality (objective function value) is considered to be the direction with maximum gradient. Also, optimizing over all contacts at same time using this method can be difficult. Therefore, we use coordinate descent approach, which optimizes over one contact then moves to the next one. Since, by construction, we get a better objective function value, once it converges, it must have reached a local maximum.

The problem configuration is similar to the hill climbing problem. It is known that for such a problem, a global optimum is not guaranteed to be found. Many methods have been proposed to find better a local maximum such as random restarts, local beam, and so on. We use random restarts to help us solve the problem.

### 7.2.2 Proposed Approach

Let $\mathbf{T} = \{\mathcal{T}_1, \ldots \mathcal{T}_N\}$ be the set of $N$ triangles in the mesh. A grasp on $\mathcal{B}$ with $n$ contacts will then associate each finger with one of the $N$ triangles. In general, multiple fingers could be associated with the same triangle. Unless the object to be grasped is simple, this case will usually not result in a satisfactory grasp.

Our grasp planning method works as follows. A sequence of random grasps assigning each of the $n$ fingers to one of the triangles are tested on the object. This is an online process that will set a threshold $Thr$ for initial grasps. Then, we again run random grasps until the score is above some percentage of $Thr$. After that, an iterative improvement process runs as follows. The positions of $n - 1$ fingers are kept fixed, while the position of the remaining finger explores the points around it in order to improve a given score function $g$. During this phase the position of the finger is not bound to stay inside the triangle it started from, but it can cross the boundary and move to one of the neighboring ones. The ability to move from one triangle to another overcomes some of the major limitations of other methods. In particular we are able to cross *sharp* edges on the boundary of the object. When it is no longer possible to improve the score of the grasp by moving that finger, its position becomes fixed and a local improvement is then sought by moving another finger. This process is iterated until no further improvements are possible. Each of the steps necessary to implement this strategy is described in the following subsections.

## Contact point representation

At the core of the method we propose lies a parametric representation for a generic point inside an arbitrary triangle. Figure 7.17 illustrates the idea.

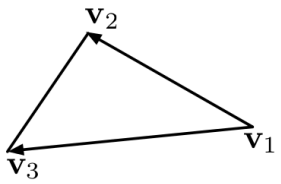



Figure 7.17: Any point inside the given triangle can be represented as a constrained linear combination of $\mathbf{v}_2 - \mathbf{v}_1$ and $\mathbf{v}_3 - \mathbf{v}_1$.

Indicating with $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ the coordinates of the three vertices of the triangle $\mathcal{T}$, referred to a given reference frame, the coordinates of a generic point $\mathbf{p}$ inside $\mathcal{T}$ can be written as

$$\mathbf{p} = \alpha_1(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.1}$$

subject to the constraints $\alpha_{1,2} \geq 0$ and $\alpha_1 + \alpha_2 \leq 1$. In the following, we indicate with $\mathbf{p}_i$ the contact point of the $i$th finger. We furthermore indicate as $\mathcal{T}^{(i)} \in \mathbf{T}$ the triangle in which $\mathbf{p}_i$ is located, and let $\alpha_1^{(i)}, \alpha_2^{(i)}$ the coefficients to obtain $\mathbf{p}_i$ using Eq. 7.1. We will also implicitly assume that the numbering of the vertices in $\mathcal{T}^{(i)}$ is unambiguously determined[6] and stored together with $\mathcal{T}^{(i)}$.

## Score function

A score function maps a set of contact points to a real value. We therefore write $g(\mathbf{p}_1, \ldots, \mathbf{p}_n)$. In the following works we will consider different score functions and details will be presented in the results section. The logic underlying the algorithm does not change as long as the score function is monotonic with grasp quality (the better the grasp, the higher the score.) In the following we will just write $g$ without explicitly specifying which one we are using.

In particular, the score function we chose to evaluate the grasps is calculated by the Partial QuickHull for Grasp Wrench Space algorithm we proposed in chapter 4. This score function is beneficial in many ways. First, the metric to evaluate force closure grasps is the most well known Ferrari and Canny metric [29], which aims to balance disturbances with the minimal effort. Second, to deal with non-force closure grasps, we can quickly calculate the distance from the convex hull to the origin, which can also be used to guide the exploration step towards a local optimal grasp. Finally, PQHGWS improved the calculation speed of the original method by a factor of 40 for grasps with 4 contacts and 8 edge approximations of the friction cone.

[6]Eq. 7.1 relies on a precise ordering of the vertices in the triangle. If the order is changed, the expression is still valid, but the coefficients change as well.

## Local improvement

Let us assume the *i*th finger is the one whose contact point position is being modified to increase the objective function while the remaining ones are kept fixed. The coordinates of contact point $\mathbf{p_i}$ can then be written using Eq. 7.1. Six different vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6$ are then computed as follows to generate six new candidate contact points:

$$\boldsymbol{\mu}_1 = (\alpha_1^{(i)} + S)(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2^{(i)}(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.2}$$

$$\boldsymbol{\mu}_2 = (\alpha_1^{(i)} - S)(\mathbf{v}_2 - \mathbf{v}_1) + \alpha_2^{(i)}(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.3}$$

$$\boldsymbol{\mu}_3 = \alpha_1^{(i)}(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} + S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.4}$$

$$\boldsymbol{\mu}_4 = \alpha_1^{(i)}(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} - S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.5}$$

$$\boldsymbol{\mu}_5 = (\alpha_1^{(i)} + S)(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} - S)(\mathbf{v}_3 - \mathbf{v}_1) \tag{7.6}$$

$$\boldsymbol{\mu}_6 = (\alpha_1^{(i)} - S)(\mathbf{v}_2 - \mathbf{v}_1) + (\alpha_2^{(i)} + S)(\mathbf{v}_3 - \mathbf{v}_1). \tag{7.7}$$

The parameter $S$ (step size) determines the magnitude of the local exploration and is iteratively altered during the computation, according to a schedule described later. The score function $g$ is then computed for all grasps $\mathbf{p}_1, \ldots, \mathbf{p}_{i-1}, \boldsymbol{\mu}_j, \mathbf{p}_{i+1}, \ldots \mathbf{p}_n$, $(1 \leq j \leq 6)$ obtained by changing $\mathbf{p}_i$ with $\boldsymbol{\mu}_j$. If no improvement is obtained, the local improvement for the *i*th finger is stopped. Otherwise $\mathbf{p}_i$ is substituted by the new point $\boldsymbol{\mu}_j$ achieving the highest value for the score function and the process continues.

It is evident that there exists combinations of values for $\mathbf{p}_i$ and $S$ such that one or more of the new candidate contact points $\boldsymbol{\mu}_j$ may fall outside the triangle $\mathcal{T}^{(i)}$ in which $\mathbf{p}_i$ is located. This is the case when $\alpha_{1,2}^{(i)} + S > 1$ or $\alpha_{1,2}^{(i)} - S < 0$, or $\alpha_1^{(i)} + \alpha_2^{(i)} > 1$. Assume $\boldsymbol{\mu}_j$ lies outside $\mathcal{T}^{(i)}$. Then, the segment between $\boldsymbol{\mu}_j$ and $\mathbf{p}_i$ must cross one of the edges of $\mathcal{T}^{(i)}$ and the neighboring triangle $\mathcal{T}_k$ can be efficiently determined, for example, using a doubly-connected edge list data structure [24] (see Figure 7.18).

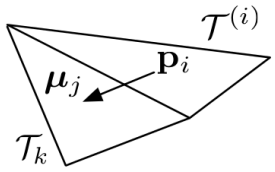



Figure 7.18: When $\boldsymbol{\mu}_i$ is outside $\mathcal{T}^{(i)}$, the neighboring triangle $\mathcal{T}_k$ can be efficiently determined.

In this case $\boldsymbol{\mu}_j$ is replaced with a random point inside $\mathcal{T}_k$. Then, if the score function obtained substituting $\mathbf{p}_i$ with $\boldsymbol{\mu}_j \in \mathcal{T}_k$ is the best among the new candidates, the *i*th finger moves to the new triangle, i.e., $\mathcal{T}^{(i)}$ is replaced by $\mathcal{T}_k$. $\mathcal{T}^{(i)}$ and $\mathcal{T}_k$ are not bound to lie on the same plane, but could be arbitrarily positioned (e.g., they could lie on two orthogonal planes.) This feature allows the local improvement phase to cross boundaries that normally limit the exploration range of gradient based methods relying on smooth surfaces.

### Step size update

Eqs. 7.2-7.7 depend on the step size parameter $S$. The parameter is initially set at 0.5. This choice is motivated by the constraints on $\alpha_{1,2}^{(i)}$ ensuring $\mathbf{p}_i$ is inside $\mathcal{T}^{(i)}$. A larger value would provide a too strong bias towards points $mu_j$ outside $\mathcal{T}^{(i)}$, while we strive to balance exploration inside and outside the triangle. As the process continues, $S$ decreases according to the formula ($S'$ is the new step)

$$S' = S\,\frac{v-b}{|v|}$$

where $v$ is the value of the score function for the newly determined best score and $b$ is the previous best value. Therefore, as the magnitude of the improvement decreases, the new step size decreases too and is eventually set to 0 when $v = b$.

### Global optimization

As mention earlier, the problem we aim to solve is similar to the hill climbing problem. There exist many algorithm to help solve such a problem and in particular, we use random restarts to help us solve the problem. One important observation for using random restarts is that if the starting point is poor, i.e., with low quality, then the local optimal solution from this starting point is also affected. Due to this observation, we decided to set a threshold for each starting point. This threshold is calculated by measuring a fix number of randomly generated grasps. After the threshold is obtained, each start point, also generated randomly, must satisfy this threshold in order to continue. At the end of this algorithm, the best grasp is selected to be the one with the highest measure over all runs.

### Algorithmic sketch

Putting all the previous steps together we obtain algorithms 5 and 6. Algorithm 5 is the high level control part. It first generates $\mathcal{K}$ random grasps and the best score is set to be the threshold $Thr$. Then we again generate random grasps until the score of the grasp is larger than $Thr$ and use algorithm 6 to improve the score and push it into local maximum. We do random restarts for this second part $\mathcal{R}$ times and the grasp with the highest score is chosen to be the final grasp. The process for generating random grasps is extremely rapid. So the first step which calculates $Thr$ will favor us in a way that we won't start the optimization process from a particularly disadvantageous initial grasp. Higher values of $\mathcal{K}$ are evidently desirable but come at the cost of a longer preprocessing step. In our experiments we set $\mathcal{K}$ to 90 and $\mathcal{R}$ to 10, but the overall performance is not too sensitive to these parameters, neither in terms of quality of the solution, nor in terms of time.

**Algorithm 5** Global optimization algorithm

**Data** : $\mathbf{T} = \{\mathcal{T}_1, \ldots \mathcal{T}_N\}, n$
**Result** : $\mathbf{p}_1, \ldots, \mathbf{p}_n$
$Thr \leftarrow -\infty$
**for** $i \leftarrow 1$ **to** $K$ **do**
    $(\mathbf{p_{r1}}, ..., \mathbf{p_{rn}} \leftarrow RandomGrasp$
    **if** $Q(\mathbf{p_{r1}}, ..., \mathbf{p_{rn}}) > Thr$ **then**
        $Thr \leftarrow Q(\mathbf{p_{r1}}, ..., \mathbf{p_{rn}})$
$Best \leftarrow -\infty$
$S \leftarrow 0.5$
**for** $i \leftarrow 1$ **to** $R$ **do**
    $Converged \leftarrow$ **false**
    **while** $Q(\mathbf{p_1}, ..., \mathbf{p_n}) < Thr$ **do**
        $(\mathbf{p_1}, ..., \mathbf{p_n}) \leftarrow RandomGrasp$
    **repeat**
        $RP \leftarrow RandomPermutation(1, ..., n)$
        **for** $j \leftarrow 1 ton$ **do**
            $\mathbf{p}_{R(i)}, v_i \leftarrow \text{Improve}(R(i), \mathbf{p}_1, \ldots, \mathbf{p}_n, S)$
            **if** $\max_i v_i > Best$ **then**
                $Best \leftarrow \max_i v_i$
                $S \leftarrow NewStep$
            **else**
                $Converged \leftarrow$ **true**
    **until** $Converged =$ **true**

**Algorithm 6** Local improvement step

$i, \mathbf{p}_1, \ldots, \mathbf{p}_n, S$
$\mathbf{p}_i, v$
**repeat**
    Compute $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6$ from $\mathbf{p}_i$ and $S$ as in Eq. 7.2-7.7
    **for** $j \leftarrow 1 to 6$ **do**
        **if** $\boldsymbol{\mu}_j \notin \mathcal{T}^{(i)}$ **then**
            Determine triangle $\mathcal{T}_k$ neighbor of $\mathcal{T}^{(i)}$
            $\boldsymbol{\mu}_j \leftarrow$RadomPoint in $\mathcal{T}_k$
    $\mathbf{p}_i \leftarrow \arg\max_{\boldsymbol{\mu}_j, \mathbf{p}_i} g(\mathbf{p}_1, \ldots, \boldsymbol{\mu}_j, \ldots \mathbf{p}_n)$
    **if** needed **then**
        Update $\mathcal{T}^{(i)}$
**until** $\mathbf{p}_i$ does not change anymore
$v \leftarrow g(\mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots \mathbf{p}_n)$

It is obvious that the algorithm eventually converges. This is true because for each step the score function improves, otherwise the whole process will terminate with a local optimal solution.

### Algorithm Limitations and Improvement

The algorithm we developed plans on the object model directly with no extra information taken into account. This type of algorithm often faces some limitations. The first major problem is that the output grasp configuration may not be applied by the robotic hand we are using. For example, if we are planning a grasp on a 1m×1m×1m box for a parallel jaw gripper, which can only open to 20cm, then no solution obtained by the planner will be valid for the gripper. This type of planner is often designed for a robotic hand with high degree of freedom on each finger such as the Shadow Hand or Schunk Hand. However, this problem can be overcome by setting the hand structure up as constraints during local improvement step. While we are exploring the surface of the object, instead of moving the contact with respect to the faces, we move the contact by changing the configuration of the finger. This improvement guarantees that every step is mapped to a valid hand configuration, but since we are constrained to explore the object surface with respect to the hand, we may lose the ability to plan over the entire object. The second problem is that the output grasp configuration may contain contacts that are not feasible in reality. For instance, for the object teacup shown in Figure 7.20, the inside of the spout is also included in the mesh model, but it is infeasible to grasp at that part of the object. One way to overcome this problem would be marking the infeasible part of the mesh, which is represented as a set of triangles. During the local improvement step, if the face, we moved the contact into along direction $\overrightarrow{v_d}$, is a marked triangle, we force that contact to keep moving along $\overrightarrow{v_d}$ until the face is not marked. Therefore, the infeasible regions are avoided while global exploration is still guaranteed.

### Local improvement for Schunk Dexterous Hand

To overcome one of the limitations mentioned above, we propose a local improvement algorithm designed for Schunk Dexterous Hand (SDH hand). This algorithm can substitute algorithm 6 and be put in use directly. The model of the SDH hand is shown in Figure 7.19. SDH hand is a fully actuated hand that has three fingers and each finger has two joints controlled independently using gear boxes. This hand also features a rotational joint that controls the rotation for finger 1 and finger 3 together with a 90 degree freedom. Using this joint, finger 1 and finger 3 can change from facing each other to facing finger 2.

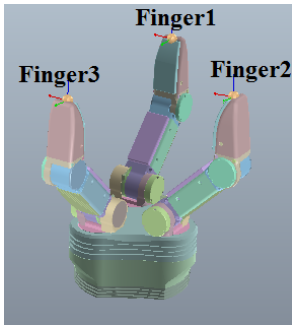



Figure 7.19: The model for Schunk Dextrous Hand.

Assume the lower joint, which connects the base of the hand with each finger, is denoted as $j_{il}$ for finger $i$. The upper joint, which connects the two links of the finger, is denoted as $j_{iu}$ for finger $i$. The finger rotational joint on the base that controls the rotation of finger 1 and finger 3 is denoted as $j_{br}$. Since we are planning for fingertip grasps, different types of joints can be understood differently. The upper joints control the distance between the fingertip and the base. The lower joint can be used to determine the exact contact point by performing a close action. The finger rotational joints determine the closing direction for finger 1 and finger 3. Based on this observation, the upper joint and the finger rotational joint are used as parameters for the hand configuration, while the lower joints are variables to be determined when the hand is in contact with the object. After the hand configuration is defined, in order to form a grasp, we also need to know the relative pose between the hand and the object. Assume the relative pose is denoted as a 6D vector $(x, y, z, roll, pitch, yaw)$, where the first 3 elements determines the relative position and the last 3 elements determines the orientation. Similar to Eq. 7.2-7.7, we define the equations 7.8-7.27 used as the local exploration of the object.

$$\boldsymbol{\mu}'_1 = j_{1u} + S \tag{7.8}$$
$$\boldsymbol{\mu}'_2 = j_{1u} - S \tag{7.9}$$
$$\boldsymbol{\mu}'_3 = j_{2u} + S \tag{7.10}$$
$$\boldsymbol{\mu}'_4 = j_{2u} - S \tag{7.11}$$
$$\boldsymbol{\mu}'_5 = j_{3u} + S \tag{7.12}$$
$$\boldsymbol{\mu}'_6 = j_{3u} - S \tag{7.13}$$
$$\boldsymbol{\mu}'_7 = j_{br} + S \tag{7.14}$$
$$\boldsymbol{\mu}'_8 = j_{br} - S \tag{7.15}$$
$$\boldsymbol{\mu}'_9 = x + S' \tag{7.16}$$
$$\boldsymbol{\mu}'_{10} = x - S' \tag{7.17}$$
$$\boldsymbol{\mu}'_{11} = y + S' \tag{7.18}$$
$$\boldsymbol{\mu}'_{12} = y - S' \tag{7.19}$$
$$\boldsymbol{\mu}'_{13} = z + S' \tag{7.20}$$
$$\boldsymbol{\mu}'_{14} = z - S' \tag{7.21}$$
$$\boldsymbol{\mu}'_{15} = roll + S \tag{7.22}$$
$$\boldsymbol{\mu}'_{16} = roll - S \tag{7.23}$$
$$\boldsymbol{\mu}'_{17} = pitch + S \tag{7.24}$$
$$\boldsymbol{\mu}'_{18} = pitch - S \tag{7.25}$$
$$\boldsymbol{\mu}'_{19} = yaw + S \tag{7.26}$$
$$\boldsymbol{\mu}'_{20} = yaw - S \tag{7.27}$$

The parameter $S$ and $S'$ (step sizes) determines the magnitude of the local exploration, which was previously defined. $S$ and $S'$ should be treated with different initial values since $S$ is the step size in

the angle domain and $S'$ is the step size in the distance domain. Eq. 7.8-7.27 represents all possible hand configuration changes with step size $S$ and $S'$. Different from the previous local improvement step, we are exploring the object with respect to the hand instead of the contacts.

After the hand configuration and hand pose is set, the next step is to locate the contacts $\mathbf{c_i}$. Since the model of the object and the model hand are determined, this step can be done by interpolating the lower joints. This is a common strategy used in simulation based grasp planners. After the contacts are located, assume its represented as $\mathbf{p}_i$, the score function $g$ is then computed. In total, we would have 20 different scores determined by $\boldsymbol{\mu}'_1$ to $\boldsymbol{\mu}'_{20}$. If all the scores are lower than the current score, i.e., the score computed without any contact change, the local improvement step is terminated. Otherwise, we move to the new configuration with the highest score. This algorithm is sketched in algorithm 7, where $C$ is the set of configurations containing the hand configuration and relative hand pose.

**Algorithm 7** Local improvement step for SDH hand

1: $C = \{j_{1l}, j_{2l}, j_{3l}, j_{1u}, j_{2u}, j_{3u}, j_{br}, x, y, z, roll, pitch, yaw\}$
2: $C, v$
3: **repeat**
4: Compute $\boldsymbol{\mu}'_1, \ldots, \boldsymbol{\mu}_20'$ from $S$ and $S'$ as in Eq. 7.8-7.27
5: **for** $j \leftarrow 1$ to 20 **do**
6: Compute contacts $\mathbf{c_j}$ by closing $j_{1l}, j_{2l}, j_{3l}$ with change $\boldsymbol{\mu}'_j$ in $C$
7: $C \leftarrow \arg\max_{\boldsymbol{\mu}'_j} g(\mathbf{c_j})$
8: **until** $C$ does not change anymore
9: $v \leftarrow g(C)$

Note that although this algorithm is exploring with 20 new options, it covers all the fingers. For the previous local planner algorithm, each finger is exploring with 6 options. This planner, built for the SDH hand, faces certain drawbacks. This planner can only plan for fingertip grasps. If the contact is not made at the fingertip, then it is not taken into account. Also, the model to represent the SDH hand is a simple skeleton which simplifies the calculation of contact points. Due to this reason, the resultant grasp might be in collision with the object. The calculation for contacts based on the skeleton does not have any collision checking. Thus, while interpolating the closing action, parts other than the fingertip might be in collision with the object. If collision checking is enabled, the parts that first collide with the object can then be used as contacts of the grasp. However, this planner can be combined with simulation software such as GraspIT! [67] or V-REP [83]. After the grasp is planned, we can use the collision checking enabled by the simulation software to verify the resultant grasp is feasible. We have developed the entire pipeline starting from planning the grasp till solving a feasible motion plan using V-REP. The result will be shown in the experiment section.

### 7.2.3 Experimental Results

In this section we present the results we obtained using the algorithm we proposed. The code is written in C++ using the Partial QuickHull for Grasp Wrench Space algorithm we proposed in chapter 4 and publicly available GSL Library. All the experiments were run in Linux under a 2.8GHz Intel i7 with 8 Gb RAM machine.

In the default case, the experiment were running for four contact points and the friction coefficient $\mu$ is set to 0.5. The friction cone is approximated with eight edges. Table 7.4 shows the five objects and its related number of triangle meshes we did our test on. Figure 7.20 gives a geometric view of the first three object.

| Object | # Tri |
|---|---|
| Joystick | 1134 |
| C shape | 96 |
| Teapot | 992 |
| Cube | 12 |
| Sphere | 960 |

Table 7.4: Number of triangles in the meshes

Figure 7.20: Three of the objects used to test the algorithm. In the following they are referred to as *joystick*, *c-shape* and *teapot*.

#### 7.2.3.1 Global Grasp Planner

The result of the first experiment is shown in Figure 7.21. It ran 100 episodes, each with *mathcalK*=90 and $\mathcal{R}$=10. The score on the y axis is the Q quality measure formally described. Similar trends for all five objects are shown in Table 7.5 and 7.6. Comparing Table 7.6 and 7.4, one can conclude that the run time is not strictly related to the number of triangle meshes of the object. In fact, the run time is related to the smoothness of the entire surface. For the five objects we tested on, the order of smoothness should be sphere, cube, joystick, teapot and C shape, which is similar to the timing order. Also from table 7.6, it is shown that the run time is hard to achieve in real time. As a matter of fact, our algorithm can run faster, having the advantage of multi-processor parallel computing. For the first K run, they do independent random grasp and for the last R run, they are also independent to each other. Thus, it is possible to use this algorithm in real time computation.

Figure 7.22 illustrates one of the major features of the algorithm we proposed; the ability to move the contact points between the different parts of an object.

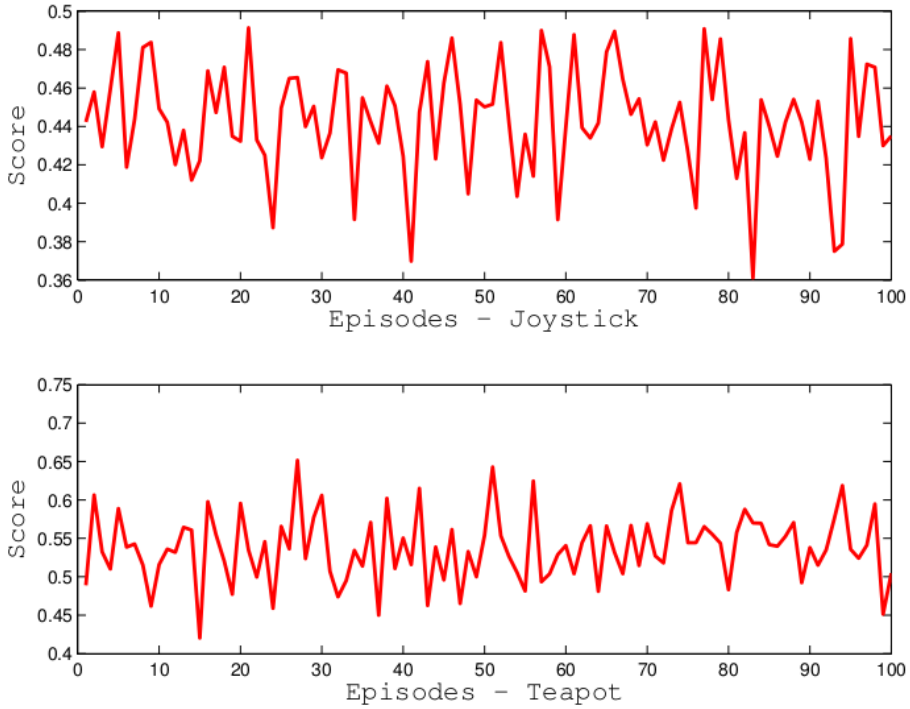



Figure 7.21: Performance on the Joystick and Teapot

#### 7.2.3.2 Global Grasp Planner for SDH Hand

Taking into account the hand structure for the SDH hand, we first show the result in Figure 7.23, which is similar to Figure 7.21. It also runs 100 episodes each with K=90 and R=10 using the local planner substituted by the local planner for SDH hand. The score on the y axis is the Q quality measure formally described. Similar trends for all five objects are shown in Table 7.7 and 7.8. The results in table 7.7 and 7.8 are similar to the result in table 7.5 and 7.6. However, the average time using the SDH hand planner is higher then the regular planner.

#### 7.2.3.3 Grasp Planner Comparison

The final experiment is to show the comparison between our two algorithms and the base line algorithm random grasp, as in Figure 7.24. The x-axis is the number of grasps evaluated, which can also be considered as the number of QuickHull functions. Since this is the most time consuming part of the algorithm, the x-axis also reflects time. The y-axis is the highest score trend. It is clear that our algorithm with and without the SDH hand, shown in green and red, reaches a higher score in less number of grasps evaluated compared to the randomized planner with and without the SDH hand, shown in cyan and blue. For the first 90 runs, four algorithms have a similar convergence rate which is obvious since all algorithms are running random grasp. After that, our algorithm improves

| Object | Avg. Q | Std. Q |
|---|---|---|
| Joystick | 0.4432 | 0.0281 |
| C shape | 0.4279 | 0.0263 |
| Teapot | 0.5377 | 0.0441 |
| Cube | 0.3619 | 0.0376 |
| Sphere | 0.6667 | 0.0260 |

Table 7.5: Grasp quality for different objects. Average and standard deviations are computed over 100 episodes

| Object | Avg. T | Std. T |
|---|---|---|
| Joystick | 8.48 | 7.6422 |
| C shape | 10.17 | 15.5095 |
| Teapot | 8.33 | 11.4297 |
| Cube | 8.09 | 9.8105 |
| Sphere | 3.26 | 0.8287 |

Table 7.6: Time spent for different objects. Average and standard deviations are computed over 100 episodes

faster in the optimization part. Even though we do multiple grasp evaluation in one restart, we can still beat the randomized algorithm. Between our algorithms, the quality of the one using SDH hand is lower than the one without SDH hand. This is clearly understandable since by using SDH hand, we added more constraints compared to the regular planner.

As clearly shown above, our algorithm beats the baseline algorithm in both score and time. As a matter of fact, our algorithm can run even faster, taking the advantage of multi-processor parallel computing. For the first $\mathcal{K}$ runs, they do independent random grasp and for the last R runs, they are also independent of each other. Thus, it is possible to use this algorithm in real time computation.

Figure 7.22: The figure shows the path followed by three contact points for object joystick (red line). Each contact point starts from the cyan location and ends in blue.

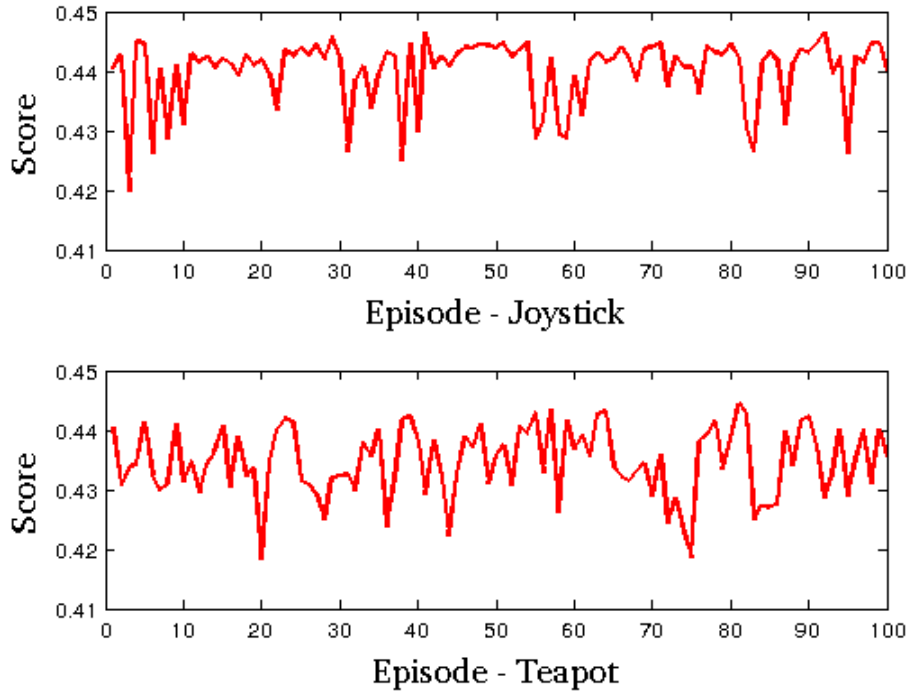



Figure 7.23: Performance on the Joystick and Teapot for SDH hand

#### 7.2.3.4 Validation using V-REP of the grasp planner based on SDH hand

Figure 7.25 shows the validation of our algorithm based on SDH hand in V-REP. Our algorithm is designed as a plug-in which computes grasp candidates for the given object for an SDH hand. After the grasp is computed, it is processed through collision checking to determine if there is any collision between the hand with the target object and the environment. If the only collision is between the fingertip of the hand and the target object, then the grasp is feasible and passed on to an IK solver to plan an arm configuration for the KUKA LWR arm. Otherwise, this grasp is rejected and the planner will plan the next grasp. At the bottom of the figure shows the grasp quality $Q_{GWS}$ for the grasp planned for all five objects. The range of the quality is from 0.0949 up to 0.4461.

| Object | Avg. Q | Std. Q |
|---|---|---|
| Joystick | 0.4399 | 0.0053 |
| C shape | 0.3097 | 0.1103 |
| Teapot | 0.4338 | 0.0056 |
| Cube | 0.4263 | 0.0255 |
| Sphere | 0.4518 | 0.0027 |

Table 7.7: Grasp quality for different objects using the SDH hand planner. Average and standard deviations are computed over 100 episodes

| Object | Avg. T | Std. T |
|---|---|---|
| Joystick | 11.4004 | 1.6270 |
| C shape | 10.3624 | 9.7375 |
| Teapot | 15.7029 | 5.3058 |
| Cube | 16.0449 | 11.2558 |
| Sphere | 18.0850 | 2.3751 |

Table 7.8: Time spent for different objects using the SDH hand planner. Average and standard deviations are computed over 100 episodes

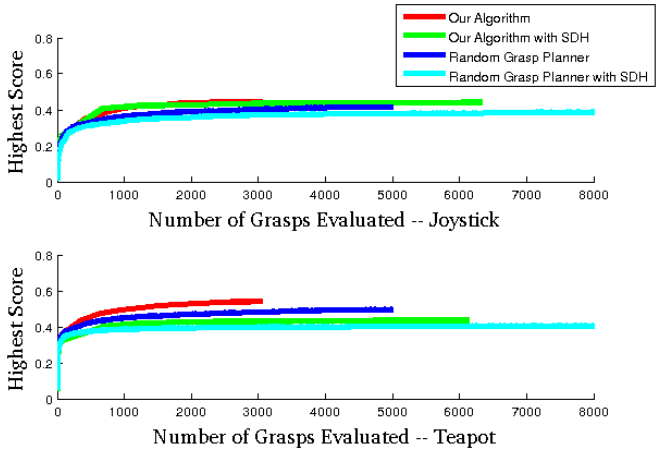



Figure 7.24: Comparison between the proposed algorithms and a randomized grasp planners.

This is reflecting one of the advantages of our planner, such that the threshold is set separately for different objects.

## 7.3 Conclusion

In this chapter, we proposed three grasp planner algorithms solving the grasp planning problem from different angles. The first two planners considered the problem of grasping objects with negative curvature and the related problem of evaluating grasp configurations where the fingers make contact in points of negative curvature. One of the planners is developed based on the geometric model of the target object, i.e., model-based grasp planner. This planner has been contrasted with a baseline planner using randomized grasps commonly used in literature and it was shown to be largely superior regarding both quality and time. Validation occurred both in simulation and on a mobile manipulator with consistent results. The other grasp planner based on negative curvature works directly with the point cloud to locate negative curvature features and compute grasps directly, i.e., model-less grasp planner. A vision pipeline to process the input point cloud and calculate points satisfying constraints defined for negative curvature feature was developed. This planner is efficient to grasp unknown objects with negative curvature features. Validation on real robot is

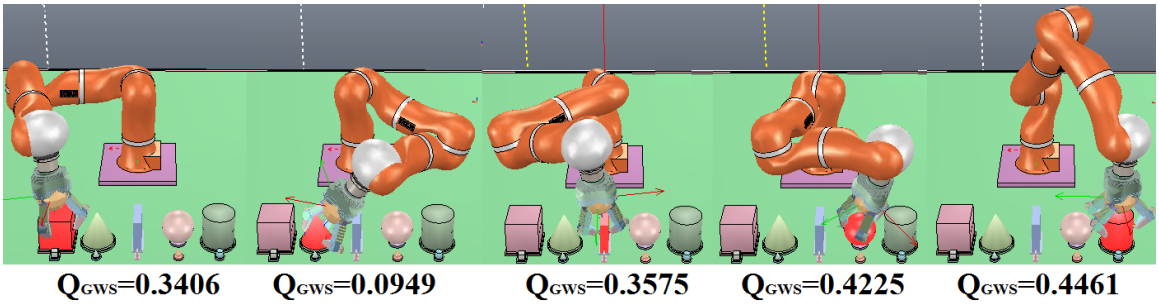



Figure 7.25: Example of a grasp planner based on the SDH hand visualized in V-REP. The corresponding IK solution is also solved for a KUKA arm.

provided, which shows some valuable results. The last grasp planner in this chapter plans grasps to obtain force closure on objects whose surface is represented using a triangle mesh. The algorithm can plan grasps for an arbitrary number of contact points and the result is obtained through an iterative process aiming at optimizing a commonly used grasp quality function. Besides the triangle mesh, the algorithm does not require any other preliminary information, like the initial placement of the contact points. Our simulations outline two major findings. First, the computational time grows very slowly with the number of triangles in the meshes and this enables the use of meshes with a large number of triangles to provide accurate approximations of objects featuring curved surfaces. We have also demonstrated that the algorithm is indeed global in the sense that during the exploration stage the contact points can move through different patches of the surface, thus overcoming one of the limitations of methods requiring smooth representations and imposing constraints on the area to be explored. The algorithm also clearly outperforms a baseline randomized planner. A variation of this planner was also presented. Instead of optimizing over the contacts, the new version considers the hand configuration, in particular, for the SDH hand. This modified version follows the same framework and overcomes the problem of connecting the planned grasp with an actual hand. The experiments for this planner based on the SDH hand is designed similar to the original version and the result agrees with our previous findings. This planner is also validated using a simulation software, V-REP, which enables the pipeline connecting grasp planning to motion planning.

# Chapter 8

# Conclusions and Future Work

In this dissertation, we focused on solving robot grasping related problems, in particular, computing force closure grasps with good quality. We tackled this problem mainly by improving grasp quality metrics and using these metrics to generate force closure grasps. Currently, grasp quality metrics are often based on a single feature, such as disturbance force rejection. Although some researchers try to combine multiple grasp quality metrics in serial or in parallel, each metric is still treated independently. Our goal is to improve what is usually considered when evaluating grasps and treat grasp evaluation as a property of the entire robot system. In this thesis we considered the effects of some of the most important features of the robot system and improved existing grasp quality metrics by taking these features into account. We also provided some insights for combining these metrics with probability and direct contact force. Our contributions are the following:

- Improved the calculation speed for two important quality metrics, i.e., the grasp wrench space metric and the object wrench space metric.
- Connected grasp quality measures with arm configuration using a probabilistic formulation.
- Connected grasp quality measures with local geometry information using the friction cone.
- Connected grasp quality measures with hand configurations using the friction cone.
- Developed a framework to combine the previous elements into a single grasp quality metric.
- Developed a global grasp planner that generates high quality force closure grasps using triangular meshes as object representations.
- Developed a model-based and a model-less grasp planner based on local geometry information.

The grasp wrench space metric is one of the most popular grasp quality measures and the object wrench space metric can be seen as a more precise, but time consuming version of the grasp wrench space metric. Our first effort, proposed in chapter 4, shows a significant speed improvement for both metrics, which made these metrics more appealing for online use. Moreover, these two

metrics are focused on resisting disturbance forces, which is the most important factor in obtaining stable grasps. Both metrics use the grasp wrench space, which is the key to adding in more features. In chapter 6, we proposed a modified version of the friction cone to embed local geometry information and hand configurations into the process of generating grasp wrench spaces. The resultant quality metric includes features like disturbance force rejection, hand structure and the local geometry of the object being grasped. In chapter 5, we combined arm information by adding probability to the grasp wrench space metric, which can also be substituted by the metric we had proposed in chapter 6. We also proposed a framework to rank arm configurations based on pre-sampled high probability force closure regions. We showed in simulation that using this framework will increase the stability of performing the grasp. A robot system for grasping is usually constructed by hand, arm, target object and the environment, where the environment is the key source to generating disturbance force. Clearly, the metric we proposed in chapters 5 and 6 is built upon all the important parts of the robot system. Although one could argue that this metric lacks enough features to represent the robot system, we hope it shows a direction and sets up a stepping stone for future improvements. Apart from our work in grasp quality metrics, we also developed multiple grasp planners in chapter 7 to use these metrics to generate grasps. One model-less grasp planner was developed to directly process the point cloud and efficiently calculate a grasp plan using the negative curvature feature. One model-based grasp planner also takes advantage of negative curvature feature and computes all grasps located at negative curvature point. The results of this planner are more robust and efficient compared to the baseline random grasp planner. Both planners were validated using a real robot and were more robust compared to the random grasp planner. The last planner we developed aims to globally identify local maximum grasp solutions for objects represented as triangular meshes. We also proposed a modified version of this planner that considers hand constraints during the planning process. Both versions show significant improvements in speed and quality compared to the random grasp planner.

The work presented in this thesis could be extended in various directions, e.g.:

- Develop a framework to embed more features into grasp quality metrics by considering the physical effects, such as influence in contact wrench, or by considering the probability of achieving a force closure grasp.

- Develop a simulation software that not only simulates geometrically but use well defined physical properties.

Friction cone plays a very important as it addresses the most direct physical effect at contact points. Referring to our work in chapter 6, we had successfully integrated local geometry of the target object and hand configuration into grasp quality metrics by modifying the friction cone. More features that affect contact wrench can also be interpreted using the friction following a similar strategy. For example, for the type of objects that deform when applying force at the contact point, the contact naturally forms a negative curvature region. So if we are measuring grasps for this type of object, we can take the level of the deformation of the object into account and quantifies the problem by the friction cone.

Another way to embed new features into grasp quality metrics is through adding force closure probability. As introduced in chapter 5, we provided a guide to measure the effect of arm configurations on grasp quality using the probability force closure grasps. As we pointed out in chapter 5,

grasp quality metrics can be considered as a random variable influenced by the entire hardware system. Any feature that brings uncertainty to this system can be integrated into grasp quality metrics by considering the probability of achieving a force closure grasp. For example, the vision pipeline of a real robot system is one of the major sources that cause uncertainty. We can investigate the distribution of the target object pose returned by the vision pipeline and follow a similar framework proposed in chapter 5 to quantify it.

As we previously discussed in chapter 2, one of the most deployed methods to solve grasping problems is based on the construction of a grasping database using a simulation software. However, the simulators being actively used in the grasping community suffer from a major drawback that affects the performance of grasping using a real robot. These simulators are limited to simulating behaviors geometrically instead of physically. For example, contacts between the hand and the object are simulated by sticking the object to the hand, the contact force is not considered as well as friction or object deformation. The gap between simulation and the real world has a huge impact on the success rate of grasping. If a simulator is capable to account all physical behavior occurred in the real world, then performing the solution calculated in the simulation will most likely succeed using a real robot.

# Bibliography


[1] J. Abel, W. Holzmann, and J. McCarthy. On grasping planar objects with two articulated fingers. *IEEE Journal on Robotics and Automation*, 1(4):211–214, 1985.

[2] J. Aleotti and S. Caselli. Interactive teaching of task-oriented robot grasps. *Robotics and Autonomous Systems*, 58(5):539–550, 2010.

[3] A. M Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[4] B. Balaguer and S. Carpin. Efficient grasping of novel objects through dimensionality reduction. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1279–1285, 2010.

[5] C. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[6] J. Barber, R. Volz, R. Desai, R. Rubinfeld, B. Schipper, and J. Wolter. Automatic two-fingered grip selection. In *Proceedings of the IEEE International Conferene on Robotics and Automation*, pages 890–896, 1986.

[7] J. Barber, R. Volz, R. Desai, R. Rubinfeld, B. Schipper, and J. Wolter. Automatic evaluation of two-fingered grips. *IEEE Journal on Robotics and Automation*, 3(4):356–361, 1987.

[8] P.R. Barragan, L.P. Kaelbling, and T. Lozano-Perez. Interactive bayesian identification of kinematic mechanisms. In *Proceedings of the IEEE International Conferene on Robotics and Automation*, pages 2013–2020, 2014.

[9] P. J Besl and N. D McKay. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[10] A. Bicchi and V. Kumar. Robotic grasping and manipulation. In *Ramsete: Articulated and mobile robots for services and Technology*, volume 270, chapter 4, pages 55–74. Springer, 2001.

[11] L. Birglen, T. Laliberté, and C. Gosselin. *Underactuated robotic hands*, volume 40 of *Springer Tracts in Advanced Robotics*. Springer, 2007.

[12] E. Boivin, I. Sharf, and M. Doyon. Optimum grasp of planar and revolute objects with gripper geometry constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 326–332, 2004.

[13] G. M Bone, A. Lambert, and M. Edwards. Automated modeling and robotic grasping of unknown three-dimensional objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 292–298, 2008.

[14] C. Borst, M. Fischer, and G. Hirzinger. Grasp planning: how to choose a suitable task wrench space. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 319–325, 2006.

[15] H. Bronnimann, B. Chazelle, and J. Matousek. Product range spaces, sensitive sampling, and derandomization. In *Proceedings of the IEEE 34th Annual Symposium on Foundations of Computer Science*, pages 400–409, 1993.

[16] B. Calli, M. Wisse, and P. Jonker. Grasping of unknown objects via curvature maximization using active vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 995–1001, 2011.

[17] T. M. Chan. Output-sensitive results on convex hulls, extreme points and related problems. *Discrete Computational Geometry*, 16:369–387, 1996.

[18] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Computational Geometry*, 10:377–409, 1993.

[19] S. L Chiu. Task compatibility of manipulator postures. *The International Journal of Robotics Research*, 7(5):13–21, 1988.

[20] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.

[21] K. Clarkson and P. Shor. Applications of random sampling in computational geometry. *Discrete Computational Geometry*, 4:387–421, 1989.

[22] M. R Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989.

[23] H. Dai, A. Majumdar, and R. Tedrake. Synthesis and optimization of force closure grasps via sequential semidefinite programming. In *International Symposium on Robotics Research*, pages 285–305, 2015.

[24] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 3rd edition, 2008.

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

[26] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 79, 2008.

[27] D. Dornfeld and M. M Helu. *Precision manufacturing*. Springer Science & Business Media, 2007.

[28] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2d objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 424–429, 1991.

[29] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2290–2295. 1992.

[30] D. Fischinger, A. Weiss, and M. Vincze. Learning grasps with topographic features. *The International Journal of Robotics Research*, 34(9):1167–1194, 2015.

[31] C. Goldfeder and P. K Allen. Data-driven grasping. *Autonomous Robots*, 31(1):1–20, 2011.

[32] C. Goldfeder, M. Ciocarlie, H. Dang, and P.K. Allen. The Columbia grasp database. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1710–1716, 2009.

[33] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information processing letters*, 1(132-133), 1972.

[34] E. Guizzo and E. Ackerman. The rise of the robot worker. *IEEE Spectrum*, 49(10):34–41, 2012.

[35] K. Hang, J. A. Stork, F. T. Pokorny, and D. Kragic. Combinatorial optimization for hierarchical contact-level grasping. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 381–388, 2014.

[36] R. He, Y. Zhao, S. Yang, and S. Yang. Kinematic-parameter identification for serial-robot calibration based on poe formula. *IEEE Transactions on Robotics*, 26(3):411–423, 2010.

[37] R. Hester, M. Çetin, C. Kapoor, and D. Tesar. A criteria-based approach to grasp synthesis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1255–1260, 1999.

[38] C. A. Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

[39] T. Iberall. The nature of human prehension: Three dextrous hands in one. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 396–401, 1987.

[40] R. A Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973.

[41] H. Jeong and J. Cheong. Evaluation of 3d grasps with physical interpretations using object wrench space. *Robotica*, 30(3):405–417, 2012.

[42] S. El Khouri, L. Miao, and A. Billard. Bridging the gap: One shot grasp synthesis approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2027– 2034, 2012.

[43] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM Journal of Computation*, 15:287–299, 1986.

[44] D.G. Kirkpatrick, B. Mishra, and C.K. Yap. Quantitative Steintz's theorems with applications to multifingered grasping. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 341–351, 1990.

[45] C. A Klein and B. E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

[46] R. Köker, T. Çakar, and Y. Sari. A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. *Engineering with Computers*, 30(4):641–649, 2014.

[47] G. A Kragten and J. L Herder. The ability of underactuated hands to grasp and hold objects. *Mechanism and Machine Theory*, 45(3):408–425, 2010.

[48] E. Kreyszig. *Differential Geometry*. Dover, 1991.

[49] R. Krug, Y. Bekirogluz, and M. A. Roa. Grasp quality evaluation done right: How assumed contact force bounds affect wrench-based quality metrics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1595–1600, 2017.

[50] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev. On the efficient computation of independent contact regions for force closure grasps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 586–591, 2010.

[51] M. Laskey, J. Mahler, Z. McCarthy, F.T. Pokorny, S. Patil, J. van den Berg, D. Kragic, P. Abbeel, and K. Goldberg. Multi-armed bandit models for 2d grasp planning with uncertainty. In *Proceedings of the IEEE International Conference on Automation Science and Engineering*, pages 572–579, 2015.

[52] C. Laugier. A program for automatic grasping of objects with a robot arm. In *Proceedings of the 11th International Symposium on Industrial Robots*, pages 287–294, 1981.

[53] C. Laugier. Planning fine motion strategies by reasoning in the contact space. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 653–659, 1989.

[54] J. Li, H. Liu, and H. Cai. On computing three-finger force-closure grasps of 2-d and 3-d objects. *IEEE Transactions on Robotics and Automation*, 19(1):155–161, 2003.

[55] Z. Li and S.S. Sastry. Task oriented optimal grasping by multifingered robot hands. *IEEE Journal of Robotics and Automation*, 4(1):32–44, 1988.

[56] Y. Lin and Y. Sun. Grasp planning to maximize task coverage. *The International Journal of Robotics Research*, 34(9):1195–1210, 2015.

[57] G. Liu, X. Xu, J.and Wang, and Z. Li. On quality functions for grasp synthesis, fixture planning, and coordinated manipulation. *IEEE Transactions on Automation Science and Engineering*, 1(2):146–162, 2004.

[58] S. Liu and S. Carpin. Global grasp planning using triangular meshes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4904–4910, 2015.

[59] S. Liu and S. Carpin. Kinematic noise propagation and grasp quality evaluation. In *Proceedings of the IEEE Conference on Automation Science and Engineering*, pages 1177–1183, 2016.

[60] S. Liu and S. Carpin. Partial convex hull algorithms for efficient grasp quality evaluation. *Robotics and Autonomous Systems*, 86:57–69, 2016.

[61] S. Liu, Z. Hu, H. Zhang, M. Kwon, Z. Wang, Y. Xu, and S. Carpin. Grasp quality evaluation and planning for objects with negative curvature. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2223–2229, 2017.

[62] M. Luo, T. Mei, X. Wang, and Y. Yu. Grasp characteristics of an underactuated robot hand. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2236–2241, 2004.

[63] J. Mahler, F. T Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1957–1964, 2016.

[64] X. Markenscoff and C. H Papadimitriou. Optimum grip of a polygon. *The International Journal of Robotics Research*, 8(2):17–29, 1989.

[65] M. T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[66] A. T Miller and P. K Allen. Graspit!: A versatile simulator for grasp analysis. In *Proceedings of the ASME Dynamic Systems and Control Division*, pages 1251–1258, 2000.

[67] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *Robotics & Automation Magazine, IEEE*, 11(4):110–122, 2004.

[68] S. Momani, Z. S Abo-Hammour, and O. MK Alsmadi. Solution of inverse kinematics problem using genetic algorithms. *Applied Mathematics & Information Sciences*, 10(1):225, 2016.

[69] H. Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

[70] R.M. Murray, Z. Li, and S.S. Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[71] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[72] D.S. Necsulescu, A. Fahim, and C. Lu. Stochastic error propagation in robot arms. *Advanced Robotics*, 8(5):459–476, 1994.

[73] V. Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988.

[74] Y. C Park and G. P Starr. Grasp synthesis of polygonal objects using a three-fingered robot hand. *The International Journal of Robotics Research*, 11(3):163–184, 1992.

[75] F.T. Pokorny and D. Kragic. Classical grasp quality evaluation: New theory and algorithms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3493– 3500, 2013.

[76] N. Pollard. *Parallel methods for synthesizing whole-hand grasps from generalized prototypes*. PhD thesis, MIT, 1994.

[77] J. Ponce, S. Sullivan, A. Sudsang, J. Boissonnat, and J. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *The International Journal of Robotics Research*, 16(1):11–35, 1997.

[78] M. Prats, P. J Sanz, and A. P Del Pobil. Task-oriented grasping using hand preshapes and task frames. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1794–1799, 2007.

[79] D. Pratticchizzo and J.C. Trinkle. Grasping. In B. Siciliano and O. Khatib, editors, *Handbook of robotics*, chapter 28, pages 671–700. Springer, 2008.

[80] F. P. Preparata and S. J Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977.

[81] M. A Roa and R. Suárez. Computation of independent contact regions for grasping 3-d objects. *IEEE Transactions on Robotics*, 25(4):839–850, 2009.

[82] M. A Roa and R. Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, 2015.

[83] E. Rohmer, S. PN Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013.

[84] R. B Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, 2010.

[85] R. B Rusu and S. Cousins. 3d is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[86] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[87] J. K Salisbury and J. J Craig. Articulated hands: Force control and kinematic issues. *The International Journal of Robotics research*, 1(1):4–17, 1982.

[88] A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[89] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 404–413. ACM, 1986.

[90] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(1):423–434, 1991.

[91] R. Seidel. Convex hull computations. In J. E. Goodman and J. O' Rourke, editors, *Handbook of discrete and computational geometry*, chapter 22. Chapman & Hall/CRC, 2004.

[92] K. B Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996.

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

[94] A. Stein, E. Geva, and J. El-Sana. Cudahull: Fast parallel 3d convex hull on the gpu. *Computers & Graphics*, 36(4):265–271, 2012.

[95] M. Strandberg. A grasp evaluation procedure based on disturbance forces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1699–1704, 2002.

[96] M. Strandberg and B. Wahlberg. A method for grasp evaluation based on disturbance force rejection. *IEEE Transactions on Robotics*, 22(3):461–469, 2006.

[97] J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, pages 477–526, 2011.

[98] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. `http://ompl.kavrakilab.org`.

[99] M. Taylor, A. Blake, and A. Cox. Visually guided grasping in 3d. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 761–766, 1994.

[100] H. Toshani and M. Farrokhi. Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a lyapunov-based approach. *Robotics and Autonomous Systems*, 62(6):766–781, 2014.

[101] J. Trinkle, J. Abel, and R. Paul. Enveloping, frictionless, planar grasping. In *Proceedings of the IEEE International Conferene on Robotics and Automation*, pages 246–251, 1987.

[102] M. Trobina and A. Leonardis. Grasping arbitrarily shaped 3-d objects from a pile. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 241–246, 1995.

[103] C. Tung and A. C Kak. Fast construction of force-closure grasps. *IEEE Transactions on Robotics and Automation*, 12(4):615–626, 1996.

[104] J. Varley, C. DeChant, A. Richardson, A. Nair, J. Ruales, and P. K Allen. Shape completion enabled robotic grasping. *arXiv preprint arXiv:1609.08546*, 2016.

[105] C. W Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, 1986.

[106] J. Weisz and P. K Allen. Pose error robust grasping from contact wrench space metrics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 557–562, 2012.

[107] D. E Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2):47–53, 1969.

[108] W. Wolovich and H. Elliot. A computational technique for inverse kinematics. In *Proceedings of The 23rd IEEE Conference on Decision and Control*, pages 1359–1363, 1984.

[109] J. D Wolter, R. A Volz, and A. C Woo. Automatic generation of gripping positions. *IEEE transactions on systems, man, and cybernetics*, (2):204–213, 1985.

[110] Z. Xue, J M. Zoellner, and R. Dillmann. Automatic optimal grasp planning based on found contact points. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1053–1058, 2008.

[111] A. C. Yao. A lower bound to finding convex hulls. *Journal of the ACM*, 28(780-787), 1981.

[112] T. Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

[113] J. Zhang, G. Mei, N. Xu, and K Zhao. A novel implementation of quickhull algorithm on the gpu. *arXiv preprint arXiv:1501.04706*, 2015.

[114] T. Zheng. An efficient algorithm for a grasp quality measure. *IEEE Transactions on Robotics*, 29(2):579–585, 2013.

[115] Y. Zheng and W. Qian. Improving grasp quality evaluation. *Robotics and Autonomous Systems*, 57(6):665–673, 2009.

[116] X. Zhu and J. Wang. Synthesis of force-closure grasps on 3-d objects based on the Q distance. *IEEE Transactions on Robotics and Automation*, 19(4):669–679, 2003.