FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Grasp planning for handoff between robotic manipulators

David Miguel Ribeiro de Sousa

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor at UMBC: Dr. James Timothy Oates

Supervisor at FEUP: Dr. António Paulo Gomes Mendes Moreira

Co-supervisor at FEUP: Dr. Germano Manuel Correia dos Santos Veiga

July 31, 2017

Resumo

Este projeto de dissertação resulta de um programa de intercâmbio com a University of Maryland, Baltimore County em colaboração com a instituição de origem, Faculdade de Engenharia da Universidade do Porto. O objetivo deste projeto foi melhorar a forma como manipuladores robóticos colaboram, em particular permitindo a estes a troca de objetos de uma forma adaptativa.

A troca de objetos entre manipuladores robóticos é uma tarefa que ainda não se encontra dominada, mas que tem bastante relevância para a utilização de robots nas várias aplicações em que podem ser utilizados, ou poderão o vir a ser.

Esta operação requer diferentes passos intermédios, que incluem a pega simultânea do objeto por duas garras distintas. Este passo, crítico, não é trivial e necessita de um planeamento de cada uma das poses das garras dos manipuladores. Nesta dissertação é apresentada uma solução para o planeamento da pose da garra do manipulador que entrega o objeto.

Uma vez que o espaço das possíveis poses da garra tem uma dimensão elevada, devido ao grande número de variáveis envolvidas, é proposto um algoritmo baseado em heurísticas e métricas com fundamento físico para lidar com esta questão.

Abstract

This Master's thesis research project results from an exchange program with the University of Maryland, Baltimore County in collaboration with the home university, Faculdade de Engenharia da Universidade do Porto. The goal of this project was to improve the way robotic manipulators collaborate, in particular allowing them to exchange objects in a flexible manner.

The handoff of objects between robotic manipulators is a task that has not been mastered but is very relevant for the prevalence of robots in the various applications that they can or will be used.

The handoff operation requires several steps, which include the simultaneous grasp of the object by two different end-effectors. This critical step is not trivial, and a careful planning of each of the grasp is necessary. In this thesis, a solution for the planning of the grasp of the hander robot is proposed.

Since the space of possible grasps is high-dimensional due to the great number of variables involved, an algorithm based on heuristics and physics based metrics is proposed to deal with this issue.

Acknowledgments

First of all, I'd like to thank Dr. Tim Oates, from the University of Maryland, Baltimore County (UMBC) for welcoming me at his lab, the Cognition, Robotics, and Learning (CORAL) lab, and for his orientation and unconditional support.

I'd also like to thank Dr. Cynthia Matuszek, from UMBC, for the help provided and for allowing to be part of the Interactive Robotics and Language (IRAL) lab. I'm also grateful for the help from my colleagues at CORAL and IRAL labs, whose discussions were always fruitful.

I also thank Dr. António Paulo Moreira for allowing me to follow my interests in robotics during my studies in FEUP and for the orientation provided.

Finally, I thank my parents and my sister, whose support made this journey possible.

David de Sousa

"Stay hungry.
Stay foolish"

Stewart Brand, Whole Earth Catalog

Contents

List of Figures

List of Tables

Abbreviations and Symbols

DOF	Degrees Of Freedom
GWS	Grasp Wrench Space
IK	Inverse Kinematics

Chapter 1

Introduction

The new applications of robotic platforms are growing. Robots have become pervasive in today's world, its applications ranging from medical, industrial, security, to even social. With this ever growing range of applications the issue of collaboration between robots has gained particular relevance.

The grasping of objects by a manipulator is a complex and widely addressed problem. One would expect that robots would be able to grasp objects as easily as humans, but this is not the case. The amount of computations to accurately generate a robust grasp can be very large. On top of that, in some cases there are unknown variables, that can increase the uncertainties, like the weight distribution of the object to be grasped, or the exact behaviour of an under-actuated joint.

When an object handover between two manipulators is considered, the problem becomes even more complex, as the object has to be grasped simultaneously by two different end effectors. This two grasps have to be robust by themselves, and cannot collide or interfere with each other.

For an easier handover action, the grasp of the manipulator that hands the object (hander) should be sturdy but at the same time, allow as many different grasps as possible by the receiving end-effector. This is as true for robots, as it is for humans, even though humans are more dexterous than most robotic systems.

1.1 Goals and scope

The cooperation between robotic manipulators has special interest today, given that most deployed robots are of this type. For a handoff operation to be completed, the following steps have to be successfully accomplished:

1. **Hander manipulator has to grasp the object;**
2. Hander manipulator has to present the object to the receiving manipulator;
3. Receiver manipulator has to simultaneously grasp the object;
4. Hander has to release the object;

5. Receiver has to move object to the final position.

The scope of the solution that is purposed in this thesis corresponds to the first step of the handoff: "1. Hander manipulator grasps the object". This step plays a major role on the success of the handoff action. Grasp actions, either by the hander or receiving manipulators are complex, and represent a task that's not yet overcome in such a way that allows for practical applications.

As such, the goal of this research, is to develop a solution for the planning of the hander grasp poses that maximizes the probability of a successful object handoff.

Moreover, it is also intended that the solution fulfills the following requirements:

1. Can be used with independent robotic manipulators;
2. Generate grasps for novel objects;
3. Can be used with any kind of gripper, independently of their number of degrees of freedoms, and physical structure.

The first requirement translates the need for a solution that enables the handoff operation between two distinct robotic manipulators, and not only between two arms of the same robot. The purpose of requirement number two, is to guarantee that the solution works with any object presented to the system and not only predefined objects. Finally, the third requirement assures that the solution is not limited to a single type of gripper, as it is the case of most current solutions, as discussed in Chapter 2.

1.2 Relevance of the problem

The exchange of objects between robotic manipulators, is a task that has several real-world applications ranging from industrial and commercial applications, to consumer applications.

Industrial and consumer applications vary, for example, from the exchange of objects between two manipulators on a production line, to the exchange of a mail package between two autonomous guided vehicles (AGVs) with a robotic arm. As an example of a consumer application, we could imagine a case where an object handoff between two manipulators mounted on wheelchairs, is needed.

In particular, the problem of grasp planning for robot-robot handoff is especially relevant, given that the grasp operations are one of the most difficult steps involved.

1.3 Outline of the document

This thesis document is composed by six chapters, including the present one.

In Chapter 2 the related work and technologies are described, where the knowledge on the topic is reviewed and the tools that are to be used on the development of the project are presented.

Chapter 3 gives insight on the algorithm proposed and details the contributions resulted from this work.

In Chapter 4 an overview of the implementation of the solution is provided. In this chapter the architecture of the final solution is discussed, giving particular attention to what tools were used, and how.

Chapter 5 presents the results obtained from the proposed method for generating hander grasps, and how these results were obtained.

Chapter 6 addresses the conclusions and the future work, presenting a discussion of the developed work and its relevance.

Chapter 2

Related Work and Technologies

This chapter presents the state of art on the topic of grasp planning in the context of robotic manipulators, as well as the technologies used in the development of the research for this thesis.

2.1 Grasp planning

In order for robotic manipulators to be able to pickup novel objects, the grasp to be used must be planned. Grasp planning in the context of robotics comprises the planning of the pose and posture of the gripper in relation to the object to be grasped. The pose of the gripper specifies its position and orientation, while the posture includes the values of the joints' angles of the gripper's actuators.

Leon *et al.* in [3] divided the problem of grasp planning into two different sub-problems: grasp synthesis and grasp analysis. According to the authors the definitions of the two sub-problems are:

> **Grasp synthesis** is the problem of finding a suitable set of contacts given an object and some constraints on the allowable contacts. [3]

> **Grasp analysis** consists on finding whether the grasp is stable using common closure properties, given an object and a set of contacts. Then, quality measures can be evaluated in order to enable the robot to select the best grasp to execute. [3]

2.1.1 Grasp synthesis

Grasp synthesis comprises the search for a valid grasp given a certain object and gripper. In order to find a valid grasp a large search space has to be explored.

The space of possible grasps is high-dimensional. It contains the 6 extrinsic degrees of freedom (DOFs) that include the 3-dimensional translation and rotation of the gripper, as well as the intrinsic DOFs that constitute finger configurations. The finger's configuration corresponds to the values of the angles of the gripper's fingers' joints. This very large search space cannot be explored exhaustively.

Several approaches have been developed in order to lower the dimensionality of the search space. In [4] [5] [6] [7] [8] the authors opt to use a large set of pre-computed grasps as templates to plan grasp poses for both novel and known objects. This approach has the inconvenience of requiring a database with a great number of grasps, which requires significant effort to collect, and doesn't guarantee good results for novel objects presented to the system.

Recent approaches that leverage machine learning methods have also been implemented [9] [10]. However these approaches are limited by the data set available and are usually only applicable to parallel jaw two-finger grippers, taking advantage of their simplicity. For more complex grippers where finding two opposed parallel surfaces isn't enough, this approach has severe limitations.

Based on neuroscience research that demonstrates that humans plan grasping actions in a lower dimensionality than the number of DOFs of the human hand, Ciocarlie *et al.* [11] presented the concept of Eigengrasps which allows the reduction of the search space dimensions. The concept provides a way to reduce the dimensionality of the search space related to the posture of the gripper. This is achieved by considering a set of predefined gripper postures, called eigengrasps, that can be linearly combined to achieve a good representation of the space of possible grasp poses of the gripper. The authors claim that a number as low as two eigengrasps is sufficient to synthesize grasp poses for common everyday objects. This method allows the reduction of the dimensionality of the search space of the intrinsic DOFs from the number of actuated joints to a vector with dimensionality equal to the number of eigengrasps used.

Simulated Annealing

The simulated annealing is a search algorithm that tries to find the global optima of a function in feasible time [12]. It's based on a stochastic model where a temperature value T, that decreases with the number of iterations, influences the probability of jumps between states of a function. A jump from the current state, to a new random one, is performed if:

$$e^{\Delta D/T} > R(0,1) \tag{2.1}$$

where ΔD is the variation of the cost function between the current state and the randomly selected one, T the simulated annealing temperature, and R is a random number in the interval $[0,1]$. As the algorithm "cools down", the global minimum (or maximum) of the cost function D is reached with an accuracy and precision dependent on the number of iterations.

Its use in grasp planning was proposed by Ciocarlie *et al.* [11], specifically the very fast simulated re-annealing technique developed by Ingber [13]. In grasp planning, this algorithm is used to search the extrinsic and intrinsic DOFs of the gripper, and the cost function D represents the quality of each grasp.

Inverse Kinematics

When synthesizing a grasp, the capability of the robot to reach the desired pose has to be checked. The poses that a manipulator is able to reach are intrinsically limited by its structure. The inverse kinematics (IK) feasibility of the goal pose has to be calculated to assure that its position and orientation are possible to be reached by the manipulator. Figure 2.1 presents the kinematic structure of a JACO manipulator.

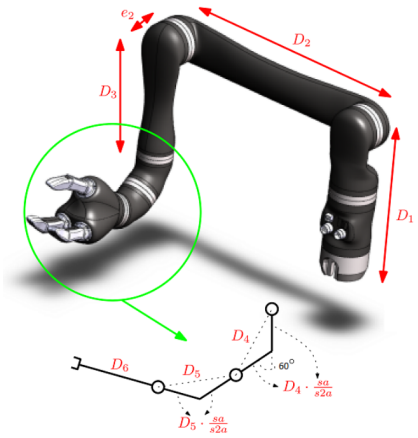


Figure 2.1: Structure of a JACO manipulator.

2.1.1.1 Grasp analysis

In order to compare the robustness of different grasps, it's necessary to be able to assign a quality value to each one of them. This quality value of a grasp can be obtained by two different possible approaches, that are very different in their computation cost: physics based metrics and heuristics.

Physics based metrics

Physics based metrics are based on contact point models that are represented by friction cones of the contact points between the object and the gripper's fingers [3]. These contact point models can be: contact without friction, contact with friction, and soft-finger contact. The first one assumes that the fingers can only transmit forces to the object along the normals of the contact points. The point contact with friction presumes a contact point according to the Coulomb model of friction. This is the type of contact point model that applies to most cases. The last one, the soft-finger contact, is applied when the contact happens with a soft finger's surface, which is able to oppose rotations around the contact's normal.

When a contact between the gripper's fingers and the object exist, it's possible to create a 6-dimensional vector that corresponds to the force and torque components applied by a finger at

the contact point. This vector configuration was proposed by Ferrari *et al.* [14], who called it a wrench, being further explored in [15].

The grasp wrench space (GWS) corresponds to the space of wrenches that can be applied to the object by every contact point [3]. Therefore, the grasp is stable against external forces whose magnitude is lower and whose direction is opposed to those on the GWS. It's then straightforward to conclude that the larger the GWS, the more stable the grasp because it is able to resist a larger number of possible external disturbances.

Consequently, two grasp quality metrics were proposed by Ferrari *et al.* [14]: the GWS volume and epsilon (ε). The GWS volume corresponds to the volume of the convex hull formed by the wrenches. The value of $epsilon$ is the radius of the largest 6D sphere that can be inscribed in by the convex hull. These metrics give direct correspondence to the grasp's quality, however they're computationally costly.

Heuristics

Heuristics are another possible grasp quality measurement, and were proposed by Ciocarlie *et al.* [11]. These simple computations can be used to evaluate the quality of grasps. They are based on the concept of virtual contact points, which are points empirically placed on the surface of the gripper's fingers. In Figure 2.2, it's possible to observe the normals of these virtual contact points represented in red.

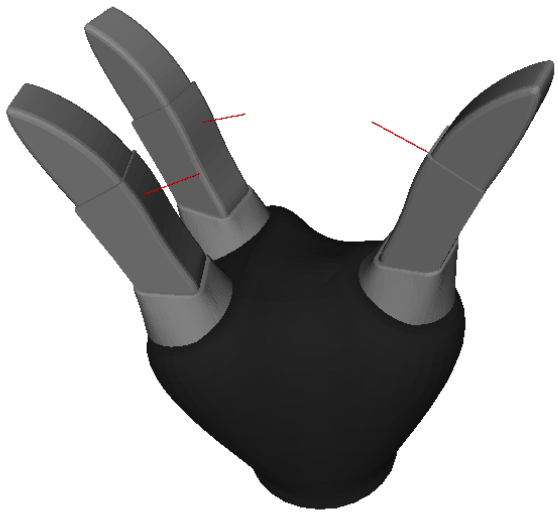


Figure 2.2: Example of the virtual contact points in red.

The first heuristic tries to minimize the distance from the finger's virtual contact points to the nearest object's surface. The second one aims at reducing the angle between the normal of the contact point and the normal of the nearest point on the object's surface. A cost function can be calculated based on the mentioned heuristics, and according to the author, these heuristics will converge to a grasping pose where the fingers reach a stable posture that enclosures the object.

This method represents a computationally efficient manner of assigning a quality value to a grasp, since the underlying calculations are simple to compute.

2.1.2 Grasp planning for handoff

When the goal is to exchange an object between two robotic manipulators, there are several constrains that have to be taken into account. The grasp from the hander robot must allow the receiving robot to grasp the object simultaneously. Furthermore, not only do both robots have to be able to grasp the object at the same time, but collisions on the approaching of the receiving manipulator and on the retracting of the hander robot must be considered.

Most work assumes the handover between two arms of the same robotic platform [16] [2], not taking into account the uncertainties involved when the manipulators are not fixed between them.

The solution proposed by Saut *et al.* [1] analyses a large number of individual grasps, generating a grasp list. Then, from this list, the grasps that collide with obstacles, for instance the table where the object is placed, are removed. After this step, the grasps whose grippers collide are removed. Finally, the grasps are rated according to the quality of the worst of both individual grasps and their IK-feasibility. The robotic platform used was an anthropomorphic robot with two arms whose position are fixed and a gripper with four fingers and total of 13 DOF's. This setup is complex and expensive.

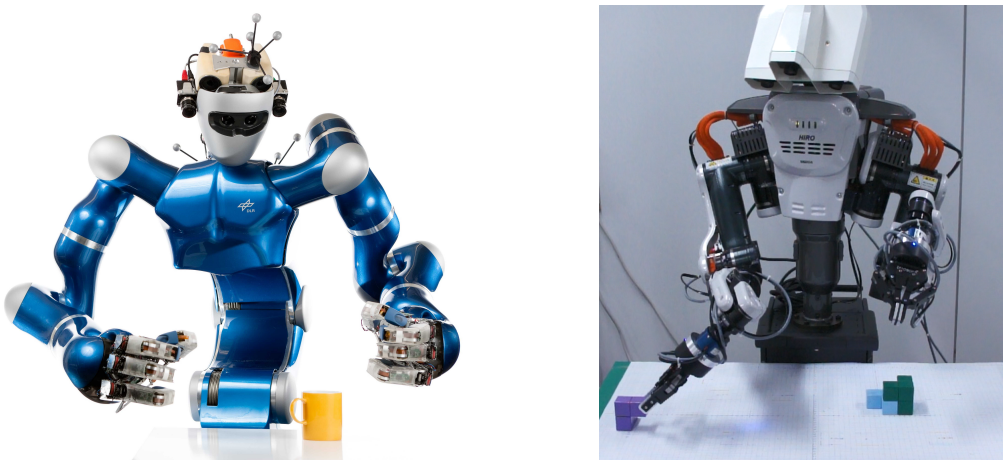
Wan *et al.* [2] uses a technique in all similar, where a large number of grasps for the hander gripper and for the receiving gripper are generated independently, and are then paired according to the absence of collisions, the angle of the grippers and their IK-feasibility. Heuristics were used to filter unwanted grasps, for example, filtering grasps that approach the object from directions that were not the same as their orientation on the robot. This approach is limited to opposite jaw grippers and was developed considering that both arms belong to the same robotic platform.

In Figure 2.3, the robots used in [1] and [2] are illustrated.

This two approaches are not very efficient given that the dual grasps are generated independently, which makes most of them useless when combining them. Moreover, the constrains that are ignored by having the two manipulators fixed between them, limits the applications where these solution can be used.

2.2 Robotic Manipulators

Robotic manipulators can be distinguished by their characteristics, including the number of DOFs, the supported payload, its reach, and several more. Given that one of the requirements of the



(a) DLR's Justin (b) Kawada's Nextage

Figure 2.3: In (a) robot used in [1] and in (b) the one used in [2].

solution is the independence of the manipulators, the ones used should not be linked to the same structure.

2.2.1 Jaco

The robotic manipulator used in this work was the JACO manufactured by Kinova®. This robotic arm has 6 DOFs and a lightweight carbon-fiber structure, weighting 5.2 kg, with a reach of 90 cm. The fact that it has low power consumption, with an average 25W, makes it ideal for mobile applications. It's used in wheelchairs as an aid for people with physical disabilities and in research on robotic platforms, as illustrated in Figure 2.4.

(a) Wheelchair with JACO (b) WPI robot

Figure 2.4: Examples of the JACO arm in mobile applications.

2.3 Grippers

Most current grasp planning approaches assume that the gripper is a planar parallel-jaw gripper, since this design has lower dimensionality of intrinsic DOFs of the gripper. It is also simpler in the sense that the grasp planning algorithm only has to select two opposing surfaces of the object that the gripper is able to grasp. Although this simple gripper design simplifies the task of predicting successful grasps, it lacks the flexibility of grippers with a larger number of fingers and DOFs. Moreover, under-actuated grippers represent a good low-cost solution for manipulation problems.

While in research the complexity of grippers has increased, with grippers that have an ever increasing number of DOFs, in consumer and industrial applications the grippers used tend to be simpler and more robust, as shown in Figure 2.5.

(a) Kinova KG-3 gripper (b) Robotiq gripper (c) Robonaut hand

Figure 2.5: Examples of grippers for different applications: consumer (a), industrial (b) and research (c).

2.3.1 Kinova KG-3 Gripper

The Kinova® KG-3 gripper is an underacted gripper, with 3 flexible fingers that can adapt to the shape and size of objects. It was developed with assistive robotic applications in mind, and is capable of grasping common everyday objects. This gripper was used in this work due to its compatibility with the JACO manipulator.

2.4 Frameworks

2.4.1 Robot Operating System

The robot operating system (ROS) is an open-source framework for the development of robotic solutions. Its architecture is based on nodes that exchange messages, providing abstraction over the hardware and underlying operating system. The concept of packages makes it possible to integrate and repurpose available code in different programming languages. It's widely adopted,

Figure 2.6: Example of the flexibility of the under-actuated finger's joints.

which justifies the existence of a large ecosystem with numerous contributors. Moreover, Kinova® provides the drivers for their robots for ROS.

2.4.2 GraspIt!

"GraspIt!" is a simulator that can simulate and evaluate grasps [17]. Contacts between the grippers, objects to grasp and obstacles can be detected in the simulation environment. It also leverages the use of eigengrasps, which is useful to reduce the search space dimensionality as presented in [11]. For single-hand grasp planning, the "GraspIt!" framework is widely used. It requires a mesh model of the gripper and of the object to be grasped, but does not support the simulation of under-actuated joints.

The "GraspIt!" simulator integrates the ability to compute the quality of a grasp using the GWS, based on the work developed by Miller *et al.* [15].

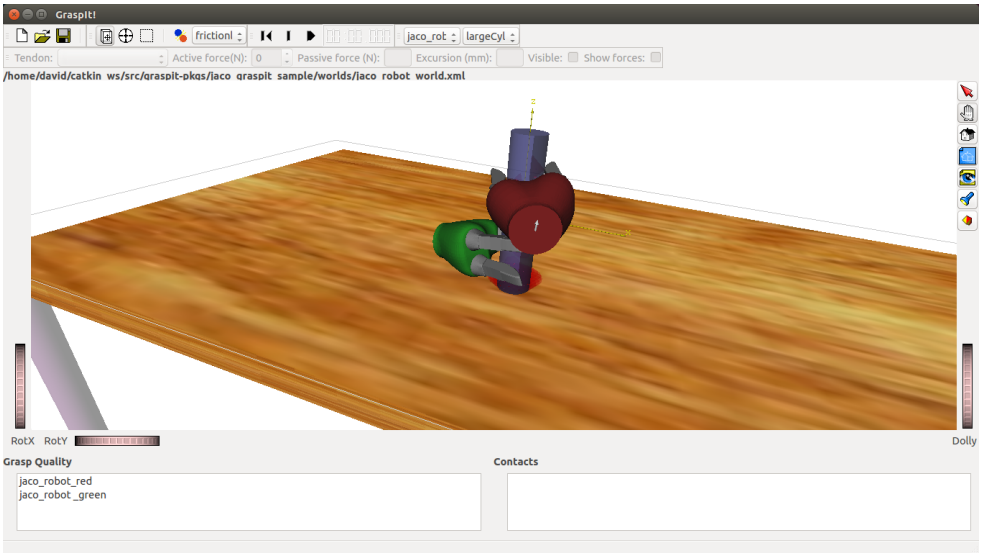


Figure 2.7: Simulation in the GraspIt! simulator.

2.4.3 MoveIt!

"MoveIt!" is a motion planning framework that allows planning trajectories of robotic manipulators, avoiding collisions with obstacles and solving the inverse kinematics of the robot [18].

Its use include, for example, given a goal grasp pose the calculation of the trajectory to the robot's final positions, while avoiding collisions with obstacles and with the robot itself. It's easily integrated with the ROS framework, and is compatible with a large number of robotic platforms.

Chapter 3

Approach

To increase the probability of a successful robot-robot handoff, the object has to be robustly grasped simultaneously by two end effectors. Moreover, for increasing the flexibility and adaptation of the operation, the first grasp should enable as many as possible different grasps by the second gripper. In order to achieve this goal, a large number of first grasps has to be synthesized, and among the robust ones, select the one that gives higher flexibility to the second grasp.

There are constrains that have to be taken into account when planning a grasp. Not only the grasp has to steadily hold the object, but the robotic system has also to be able to reach the goal pose. The kinematics of the structure of a manipulator doesn't allow its end-effector to reach any pose. This limitation makes some grasps' poses IK-unfeasible.

This chapter will focus on the approach used. The algorithm developed will be explained, as well as the heuristics introduced in this work.

Because the execution of the first grasp can fail or result in a grasp different from the one that was originally planned, the planning of the receiving grasp before the execution of the first grasp on the real robot, can become irrelevant. This occurs, especially in the presence of underactuated joints, such is the case tested. Therefore, the grasp for the receiving manipulator won't be considered final, and instead, it is assumed that after the grasp is executed on the real system a new grasp planning, considering the final pose of the hander gripper, is executed.

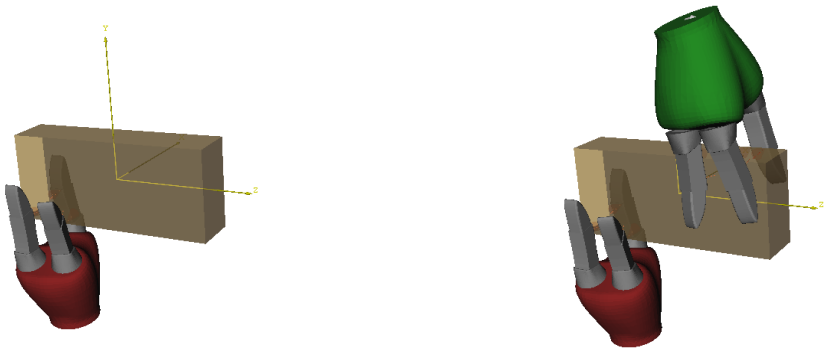
3.1 Solution proposed

The solution proposed is to find the grasp of the hander manipulator that gives the receiving manipulator more options for stable grasps. This follows the logic that the grasp planning by the receiving gripper is a complex task, and by optimizing the first grasp, the probability of a successful handoff is greatly increased.

3.1.1 The Algorithm

The algorithm proposed starts with the synthesis of a large number of grasps, in the magnitude of tens of thousands, for the hander gripper. For each grasp by the hander gripper, the quality is

assessed with several different heuristics, that are presented next. After the large number of grasps is analyzed, the top K grasps with higher quality are retained. This top K grasps are then re-ranked using more precise and computationally expensive physics based metrics, using the GWS described in subsection 2.1.1.1.



(a) Single grasp (b) Dual grasp

Figure 3.1: Example of a single grasp from the hander gripper in (a) and a dual grasp in (b).

For each hander grasp, such the one presented in Figure 3.1a, in the top K, a large number of simultaneously grasps by the receiving gripper, like the one represented in Figure 3.1b, is then synthesized.

During the synthesis of the receiving grasps, each one of these simultaneous grasps n_r is evaluated not only by the heuristics used in the single grasp planning but also by new ones that translate the quality of the grasps as a pair. The top K_r grasps by the second gripper are then ranked according to physics based quality metrics, as well as by the existence of intersections of the convex hulls of the grippers. The final quality of a hander gripper is given by the average of the quality of the top K receiving grasps that it allows. This algorithm has several parameters that can be adjusted, namely K_h and K_r that are the number of hander and receiving grasps that are retained, as well as n_h and n_r that correspond to the number of grasps evaluated for the hander and receiving grippers, respectively. Algorithm 1 helps understanding the steps involved.

Algorithm 1: Generate grasp pairs

Result: Ranked K_h hander grasps

Evaluate n_h grasps for the hander gripper according to heuristics methods;
Rank top K_h grasps according to physics quality metrics;
foreach *Top K_h grasp* **do**
    Evaluate n_r grasps by the receiving gripper;
    Rank top K_r grasp pairs according to physics quality metrics, and convex hull intersections;
end
Rank hander gripper's grasps according to the quality of possible receiving grasps;

3.2 Synthesis of Single Grasps

For obtaining valid grasp pairs, it's necessary that the individual grasps are valid. In a first step, heuristics are used to evaluate the quality of the grasps. Then, with a smaller number of grasps, pre-selected by the use of the heuristics, more complex metrics are applied.

3.2.1 Heuristics

As seen in Chapter 2, heuristics can be used to assign a quality value to a grasp. The first two heuristics presented here were introduced in [11] and are: minimization of the distance from the virtual contact points to the object's surface, and minimization of the angle between the finger's surface and nearest object's surface. The remaining heuristics were introduced in this work.

These heuristics are used to evaluate the quality of pre-grasp poses, which means that the gripper's fingers are not in contact with the object. The reason behind this is that contacts between surfaces require more computations to simulate, and therefore the execution time would increase if they were considered during the iterations of the simulated annealing algorithm.

The cost function, that is minimized by the simulated annealing algorithm, is the sum of different components. Each of these components corresponds to a heuristic. The cost function for the synthesis of the hander grasps is given by:

$$C_h = k_d \cdot c_d + k_a \cdot c_a + k_c \cdot c_v + k_o \cdot c_o \tag{3.1}$$

where c_d is the cost component associated with the distance between the fingers and the objects, while k_d is the weight of this heuristic on the final cost function. As for c_a, it is the cost component associated with the angle between the finger's and objects surface's, c_v the cost associated with the distance of fingers to the object's vertices, and c_o the cost related with the grasping of opposing surfaces of the object. Finally, k_a, k_c and k_o are the weights of each of these components, respectively.

3.2.1.1 Distance between the fingers and the object

This heuristic tries to minimize the virtual contact point's distance to the nearest object's surface. It's intended that the fingers are as close to the object's surface as possible. As depicted in Figure 3.2 by the blue arrow, the distance from the virtual contact point to the nearest point on the object's surface is a line perpendicular to the object's surface.

The expression that translates this heuristic is:

$$c_d = \sum_{i=0}^{N} d_i \tag{3.2}$$

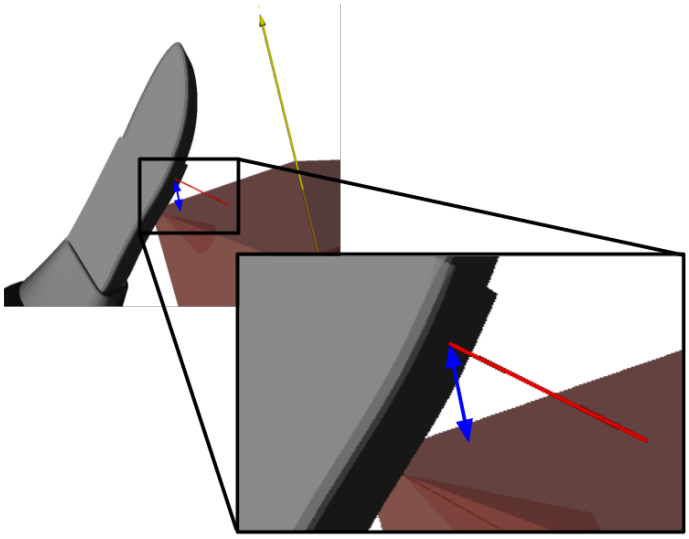


Figure 3.2: Distance from the virtual contact point to the nearest point on the objects surface.

where N is the number of virtual contact points of the gripper, and d_i the distance of the contact point i to its nearest object's surface for that particular pose.

3.2.1.2 Angle between finger's and object's surfaces

This heuristic tries to minimize the angular differences between each of the contact points' normals and the normal of their closest object's surface point. Owing to the fact that the fingers are able to exert force in the direction normal to its surface, and therefore maximizing the friction between the fingers and the object's surface when the two surfaces are parallel, allowing a steadier grasp. Figure 3.3 illustrates the angle that is minimized.

This heuristic can be translated by the following expression:

$$c_a = \sum_{i=0}^{N} \angle(\widehat{n}_i, \widehat{n}_s) \tag{3.3}$$

where c_a is the cost associated with the angle component, $\widehat{n}_i$ the finger's surface normal vector at the i-th virtual contact point and $\widehat{n}_s$ the surface's normal of the closest point to the virtual contact.

3.2.1.3 Object's vertices proximity

On object's mesh models, vertices concentrate on areas where changes in the surface's curvature occur. With the goal of leaving as much surface available for the receiving gripper to grasp, grasping on the middle of the object is to be avoided. This heuristic tries to bring the gripper's fingers close to inflection points of the object's shape, on the premise that the object's model vertices concentrate on these points.

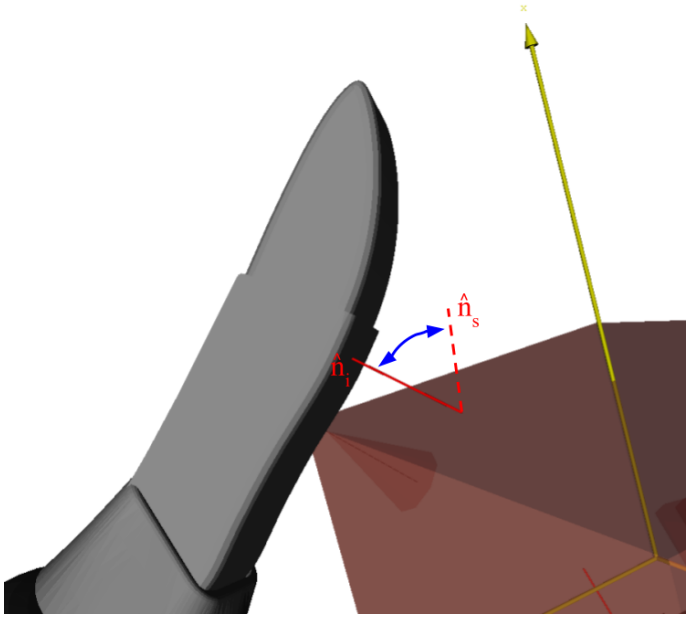


Figure 3.3: Angle between the object's and finger's normals.

Minimizing the distance to all the vertices doesn't always work, as in most cases when a point gets close to a particular object's vertex, it also gets equally distant from the other vertices. That is, if we imagine a two-dimensional line and a point between the two ending vertices of the line, as we get close to one of the vertices, we get equally distant from the other vertex. Therefore, the sum of the distances to these two points would remain the same.

Hence, instead of minimizing to all the vertices, the distance to the nearest vertices of the current pose of the gripper is to be minimized. Consequently, this heuristic minimizes the distance of the virtual contact points to their closest object's vertices, for each gripper's pose. In figure 3.4 it's illustrated an example of the distances of a virtual contact point to some of the object's vertices.

The distances from each virtual contact point to each object's vertex are sorted in ascending order. Then, the distance is divided by its index on the array, so the shortest distances are given a larger weight on the result. The simulated annealing algorithm will then prioritize the minimization of those values with a greater effect on the final cost function. This heuristic can be expressed as:

$$c_v = \sum_{i=0}^{N} \sum_{j=0}^{n} \frac{d_{ij}}{j+1} \tag{3.4}$$

where n is the number of vertices of the object's model, and d_{ij} is the distance of the contact point i to the j vertex. The vertices are sorted by their distance, in a manner that $d_{i,j} \leq d_{i,j+1}$.

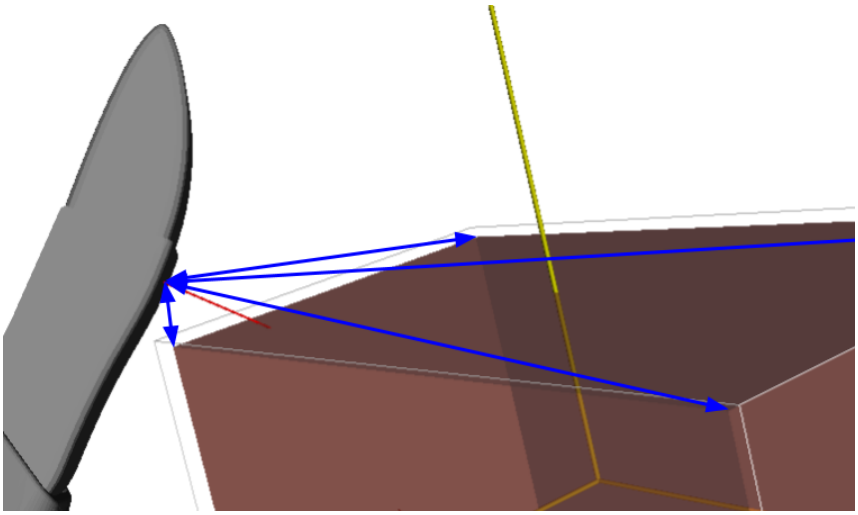


Figure 3.4: Distance of the contact points to some of the vertices.

3.2.1.4 Object's opposing surfaces

The previous heuristics still don't guarantee that the gripper pose will grasp opposing surfaces of the object. Grasping opposing surfaces is desirable because the normal component of the force that can be applied to the object surface is greater, and therefore resulting in a larger friction force, for the same friction coefficient μ, assuming the Coulomb's friction model: $F_f = \mu F_n$.

This heuristic tries to maximize the angle between the normal's of the object's surface contacting with opposing fingers. On the three-finger gripper illustrated in Figure 3.5 the opposing fingers correspond to the thumb and the index, and to the thumb and the pinkie. On the Figure 3.5b the thumb and the index fingers are grasping adjacent surfaces, whose normals make a 90^o angle between them. While, in Figure 3.5a, both the pairs thumb/index and thumb/pinkie are close to object's surfaces whose normals make an 180^o angle between them. The second option is preferred, given that the friction force that can be applied is greater than on the first case.

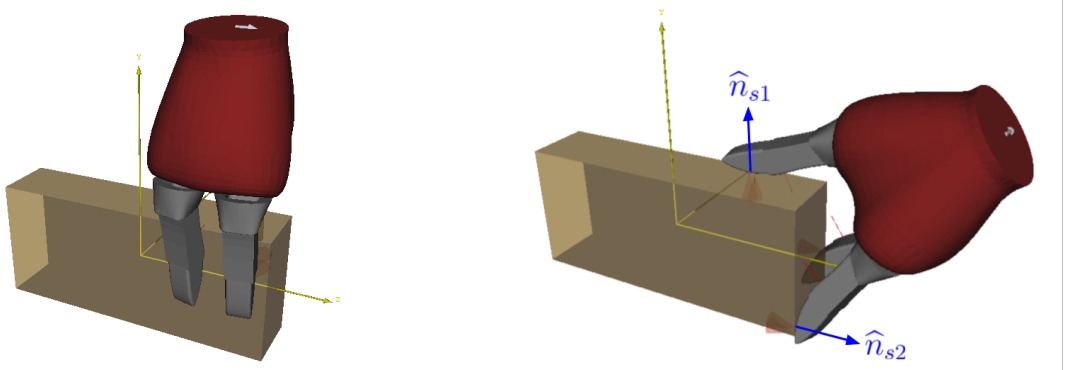
The expression that translates this heuristic is:

$$c_o = -\angle(\widehat{n}_{s1}, \widehat{n}_{s2}) \tag{3.5}$$

where $\widehat{n}_{s1}$ and $\widehat{n}_{s2}$ are the normals of nearest surface points from the corresponding virtual contact points on opposing fingers.

3.2.2 Inverse Kinematics

In order to assure a synthesized grasp is actually reachable by the manipulator, the inverse kinematics (IK) feasibility of the grasps is checked, based on the object's pose in relation to the manipulator's base frame.



(a) (b)

Figure 3.5: Example of a desired opposed surface grasp (a) and an undesired grasp (b).

Since the computation of the IKs (i.e. the angles of the robot's joints, given an end-effector goal pose) is computationally expensive, it's not efficient to compute it to every grasp. So, an alternative approach is used: every 500 iterations of the simulated annealing algorithm the, IK-feasibility of the grasp is computed. If the grasp isn't IK-feasible it is added to an "avoid list". Every time the simulated annealing algorithm comes close a state in this list, it automatically jumps to another random state and the IK-feasibility is re-checked until a valid state is found.

This approach requires fewer calculations, but still results in IK-feasible grasps. It was only used on the synthesis of the hander grasp, because as stated before there are too many uncertainties that affect the receiving grasp.

For IK-feasibility check, the object's position in relation to the base of the robot has to be defined before the algorithm is run.

3.2.3 Physics based metrics

Given that physics based metrics are several orders of magnitude more complex to compute than the previously presented heuristics, it's not viable to exhaustively compute these metrics to every grasp, since it would considerably increase the execution time. Consequently, these metrics are only computed on the final top K grasps.

The poses obtained from the simulated annealing algorithm are pre-grasp poses because the fingers, although close to the object's surface, they are not yet in contact. To obtain a grasp from this pre-grasp pose the fingers are closed until contact with another body is detected, obtaining that way contact points between the object and the gripper. This action is performed for every pre-grasp in the top K results of the simulated annealing algorithm.

The physics based quality metrics used are the volume of the GWS and its epsilon (ε). Where the closer ε is to 1, the more effective is the grasp. Similarly, the larger the volume of the GWS

the more stable it is. As discussed in the subsection 2.1.1.1, these metrics give the capacity of the forces and torques applied by the gripper's fingers on the object to restrain its movements.

These two metrics were summed for each grasp, and subsequently, the grasps were sorted according to their quality.

3.3 Dual grasps

After obtaining a robust hander grasp, a coexistent grasp, the receiving, must be synthesized. Moreover, the receiving grasp has to be robust by itself, but the quality of the grasp pair must also be analyzed.

Some work developed on the topic of dual grasping opt to synthesize individual grasps and then use a set of heuristic filters to select the grasps that can be paired together without collisions [1] [19].

The approach used is different because instead of generating two list of independent grasp, the hander grasp candidates are obtained first, and the receiving grasps are generated for each of those candidates. Therefore, the used process is more efficient given that, when synthesizing the second grasp the quality of the grasps as pair is being taken into account.

The metrics presented next are used in the evaluation of the grasp of the receiving gripper by the simulated annealing algorithm.

3.3.1 Heuristics

The quality of a grasp pair for an object handoff is difficult to evaluate. There are no physics based metrics for the measurement of their quality. But the success of the handoff task is dependent, of not only selecting robust individual grasps but a good grasp pair as well.

In order to select two individual grasps that can be combined on the same object for the handoff operation, it's necessary that no collision between them occur either in the grasping state, the approaching of the receiving manipulator, or on the retracting of the hander manipulator. The approaching of the receiving manipulator includes the pre-grasp pose, a pose where the gripper is oriented according to the final position but it's a pre-defined distance away from the object. Once the hander manipulator reaches the pre-grasp pose, it moves linearly (if its kinematics structure allows it) to the goal position and only then it closes the fingers in order to grasp the object. On this approaching movement or in the closing of the fingers, collisions can occur. The retracting of the hander robot happens after the receiving manipulator successfully grasps the object.

This retracting includes opening the fingers, and move to a post-grasp position which is a position similar to the pre-grasp. To synthesize a good receiving grasp after obtaining hander grasps candidates, a set of new heuristics were introduced. The cost function for the synthesis of the receiving grasp is given by:

$$C_r = C_h + k_{dg} \cdot c_{dg} + k_{ag} \cdot c_{ag} \tag{3.6}$$

where C_h is the cost function given by the heuristics presented in 3.2.1, and c_{dg} and c_{ag} two new heuristics presented next. While, k_{dg} and k_{ag} are the weights of these two heuristics.

3.3.1.1 Distance between the grippers

To increase the probability of a successful handoff, the pose of the hander and receiving grasp should be as distant as possible, in order to avoid collisions. Not only the center of mass of the grippers should be distant but also the fingers, because they're more prone to collisions given that their links are moving after the pose of the gripper is stationary.

This heuristic maximizes the distance between the virtual contact points of the two grippers. In Figure 3.6 illustrates, with blue arrows, the distance of a virtual contact point of the green gripper to the virtual contact points of the red gripper.

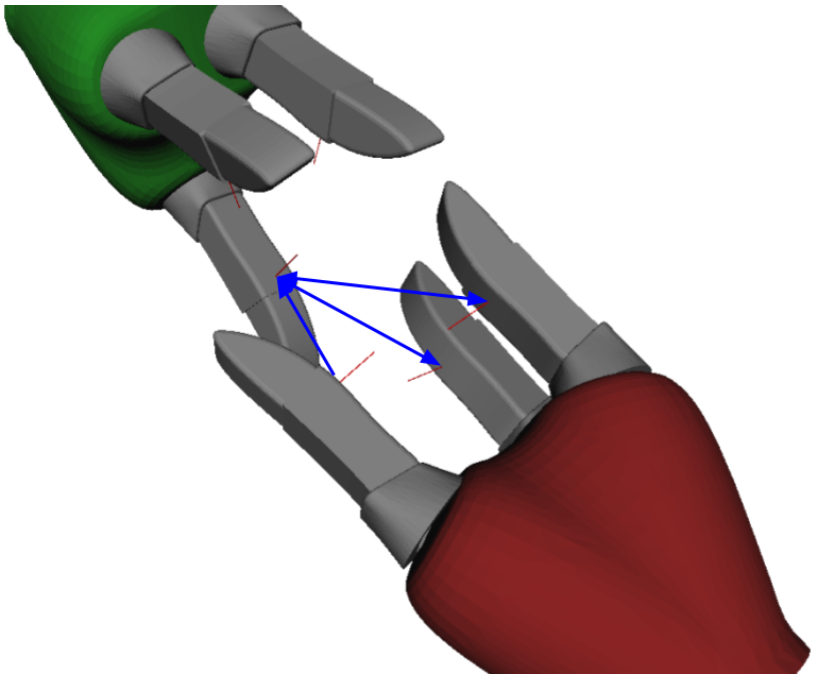


Figure 3.6: Example of the distances of a virtual contact point on the green gripper to the virtual contact points of the red gripper.

The following expression translates the heuristics used to maximize the distance between the fingers of both grippers:

$$c_d = \sum_{i=0}^{N_h} \sum_{j=0}^{N_r} d_{ij} \tag{3.7}$$

where N_h is the number of virtual contact points of the hander gripper, and N_r of the receiving gripper, and d_{ij} is the distance of the i-th virtual contact point of the hander gripper to the j-th virtual contact point of the receiving gripper.

3.3.1.2 Angle between the grippers

In order to increase the probability of successful handoff the value of the angle between the grippers can be set to a known angle between the position of the manipulators, in order to avoid IK-unfeasible poses for the receiving grasp orientation. Figure 3.7 illustrates the angle between the two grippers.

The angle formed by the two grippers can be set to a G value, and the simulated annealing will try to orient the grippers in order to minimize the difference of the angle G and the angle of the grippers.

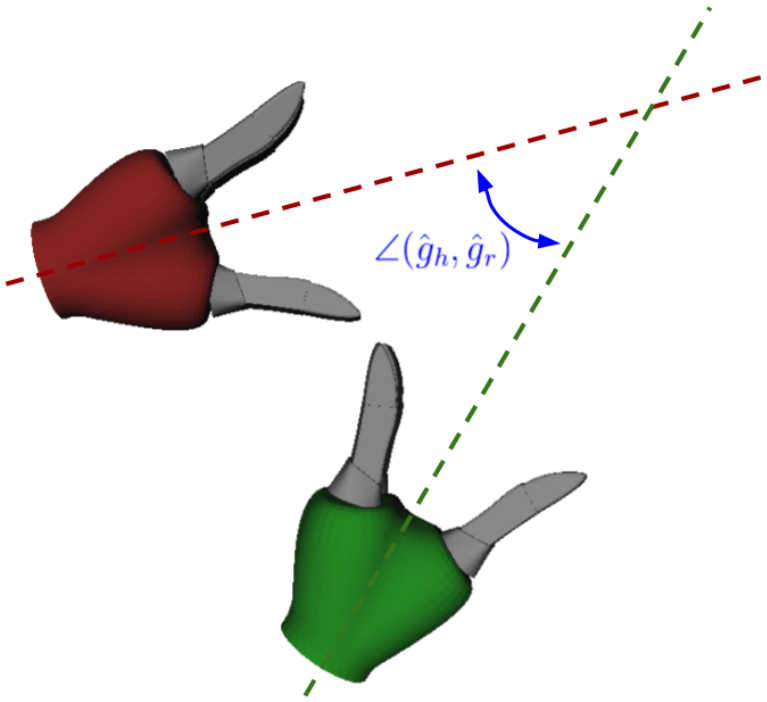


Figure 3.7: Angle between the hander and receiver grippers.

The expression that translates this heuristic is:

$$c_{ag} = \|\angle(\hat{g}_h, \hat{g}_r) - G\| \tag{3.8}$$

where $\hat{g}_h$ and $\hat{g}_r$ are the vectors oriented along the base of the hander and receiver grippers, respectively.

3.3.2 Convex hull

Since, collisions between the grippers have to be avoided, either on the approach of the receiving gripper and/or the retracting of the hander gripper, the intersections of convex hull of the grippers on their final pose are analyzed.

The convex hull of a 3D model can be analogized to an elastic fabric that wraps the object. It encloses the external points of the object's model creating a convex surface. When intersections of the convex hull of the grippers exist, there's a higher probability of collisions between them.

The convex hull computation is too costly and cannot be performed to every grasp, therefore this analysis in only performed on the final top K_r grasp pairs. That is, when the final number of iterations for a receiving grasp is reached, the intersections of the convex hull of the grippers are computed.

If convex hulls intersect, the grasp pair quality is subtracted by a constant. There are cases where the object is too small and all the grasp pairs' convex hulls intersect, in that case all of them are subtracted by the same constant and still sorted by the other parameters, so this heuristic doesn't affect those cases.

$$C_f = \begin{cases} C_r + H, & if \text{ convex hulls intersect} \\ C_r, & otherwise \end{cases}$$

where H is a constant with a value of greater magnitude than the cost function.

3.4 Final ranking

Seeing that, the final goal is to select a hander grasp that allows for the receiving grasps with greater quality, the qualities of the top K_r receiving grasps for each hander grasp in the top K_h are averaged:

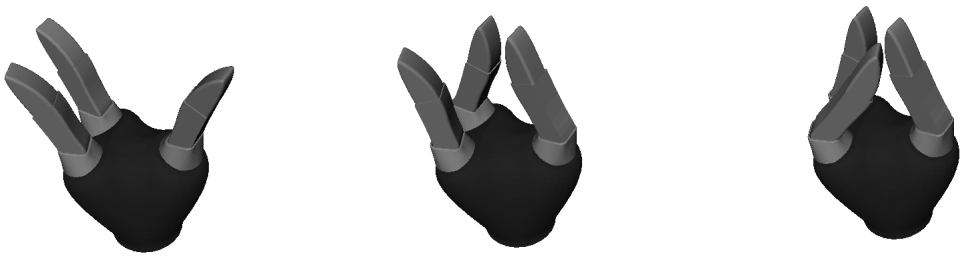
$$Q_r = \frac{1}{K_r} \sum_{i=1}^{K_r} q_{r,i} \tag{3.9}$$

where Q_r is the average of the quality of the top K_r grasps for a grasp in the K_h. While, $q_{r,i}$ is the quality of each i-th receiving grasp in the top K_r grasps. The final quality of the hander grasp, Q_p, is then given by the average of its quality and the average of the qualities of the receiving grasps that it allows:

$$Q_p = \frac{Q_r + q_h}{2} \tag{3.10}$$

3.5 Eigengrasps

As discussed in Chapter 2 the number of variables that compose the search space is significantly large. Eigengrasps help reduce the number of variables to search on the intrinsic DOFs of the gripper. According to the results obtained by Ciocarlie *et al.* [11], as low as two dominant eigengrasp can be used to generate good results. Therefore, an eigengrasp was composed with the two most common postures of this gripper. These postures include the pinch grasp where the index and thumb of the gripper are closed. And a full closed grasp where all the gripper's fingers are closed. Figure 3.8 provides an illustration of the eigengrasps used. The eigengrasps allow searching a vector that corresponds to the linear combination of the postures that compose the eigengrasp.



(a) Open (b) Pinch (c) Closed

Figure 3.8: The hand poses that were used as eigengrasps and the default pose.

3.6 Simulated annealing

Given the high-dimensionality of the search-space, the simulated annealing algorithm was used to synthesize the grasps, in particular, the "Very Fast Simulated Re-Annealing" algorithm proposed by Ingber [13]. In this modified algorithm the scheduling of the value of the simulated annealing temperature, T, is decreased exponentially according to:

$$T = T_0 \cdot e^{-k^{1/D}} \tag{3.11}$$

where D is the dimension of the search space, which for the grasp synthesis is 7: 3 variables give the 3D translation, another 3 the rotation and finally the eigengrasp vector. The variable k is the annealing step, while T_0 is the initial temperature.

For each iteration of the algorithm, a new random neighbor state is generated from the current state by obtaining a new random value to each of the variables involved. Each variable, i, of the new state S_{ni} is given by:

$$S_{ni} = S_{ci} + T \cdot (-1)^{round(R(0,1))} \cdot \left(1 + \frac{1}{T}\right)^{R(-1,1)} \tag{3.12}$$

where S_{ci} is current value of that variable, and $R(A,B)$ is a random number in the interval $[A,B]$. A "jump" to the new state is performed if the following condition is verified:

$$e^{\frac{c_c - c_n}{T}} > R(0,1) \tag{3.13}$$

where c_c is the value of the cost function for the current state and c_n the cost function of the new state.

Chapter 4

Implementation

This chapter provides an overview of the implementation of the solution proposed. It addresses the tools used and how the whole pipeline operates. It's worth mention that all the tools used are open-source and available online.

In a first step, the grasp planner "GraspIt!" is used to run the algorithm described is Chapter 3. The output of the grasp planner is a ranked list of hander grasp poses, which are saved on a file.

On the second step, the motion planner "MoveIt!" reads the grasps poses and plans the trajectory of each one, by their order on the ranking. If the first grasp is not possible to execute due to collisions with obstacles in the environment, the next one is tested. When a valid grasp pose is found, the motion planner sends the trajectory points to the robot driver, which executes them on the real manipulator. Both, the robot drivers and the "MoveIt!" motion planner are ROS nodes.

In Figure 4.1 is illustrated the architecture of the solution implemented. In the link `https://goo.gl/VoMNF4` the reader can find a video presenting an overview of the implementation.

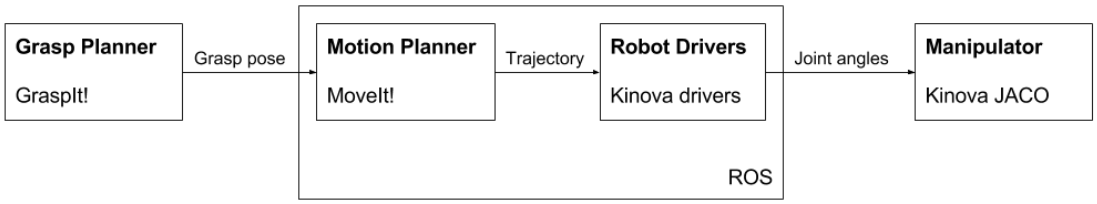


Figure 4.1: Architecture of the implementation of the solution.

4.1 Grasp generation

To implement the algorithm proposed, the "GraspIt!"[1] framework was used. This open-source simulator is implemented in C++, and its source code had to be altered in order to allow the simulation of multiple grippers, using the previous presented metrics for evaluating the quality of both the individual grasps, and the grasps pairs.

In this framework there are three kinds of entities that interact on the simulation world: robots, objects and obstacles. The grippers used correspond to robots entities, while the item to be grasped is an entity of the type object. Other bodies can be added as obstacles. Because in the real world the objects are supported by some other body, and this obstacle has to be taken into account for collisions detection when generating the grasps, it should also to be present on the simulator. Therefore, on the synthesis of the grasp by the hander gripper, a table was added below the object in order to avoid grasps that collide with this obstacle, since that on the real world environment it was also present.

The implementation of the simulated annealing algorithm in this tool is single-thread. The GUI of the simulator, illustrated in Figure 4.2 allows to visualize the grasps.

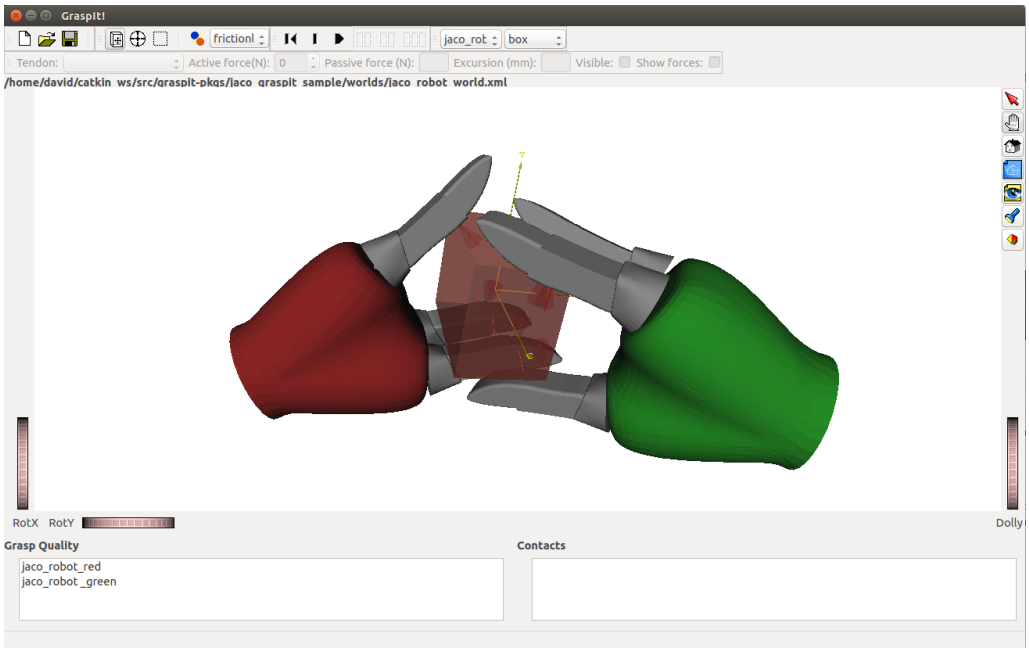


Figure 4.2: The GUI of the GraspIt! framework.

The grasp generation is started manually, and once it finishes the grasps poses are exported to a file that can be imported by "MoveIt!". The grasps exported, have the coordinates in the object's local coordinate system. This includes the translation, as well as rotation represented by a quaternion.

[1] https://github.com/graspit-simulator/graspit

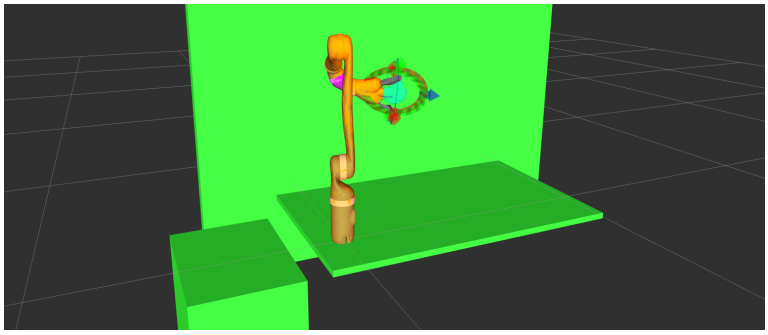
4.1.1 Convex Hull

During the execution of the algorithm, the convex hull of the 3D models of the grippers is computed as was explained in Subsection 3.3.2. The used framework, doesn't include a way to compute the convex hull of the bodies in the simulation enviroment. For that reason, the library CGAL[2] was used to compute the convex hulls of the grippers and the collisons between them.

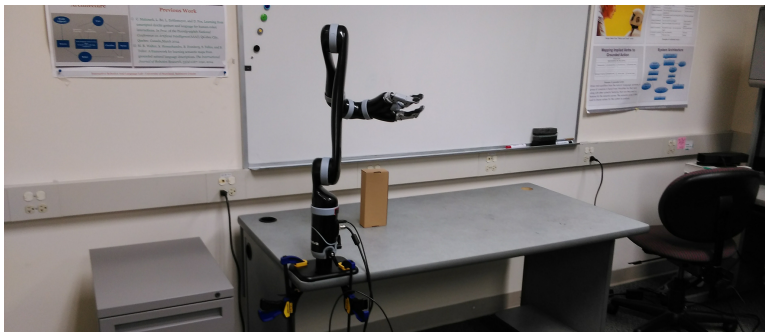
The 3D model of the KG-3 gripper that was used had more than 100.000 vertices. They were "down-sampled" by a factor of 3. Meaning that the convex hull was computed on approximately 33.000 points, which still is good enough to validate intersection, but reduced the execution time significantly, from an average of 348 ms to 139 ms per convex hull intersection computation.

4.2 Motion planning

The tool used to plan the motion of the manipulator, given the final gripper pose and the 3D environment was "MoveIt!"[3]. The grasps generated are imported to this framework, and the trajectory for the goal grasp is planned considering the obstacles of the environment that were previously defined in the simulation world. Figure 4.3 illustrates the environment simulated in "MoveIt!" and the laboratory where the tests took place.



(a) MoveIt! environment



(b) Real lab environment

Figure 4.3: Example of the environment of the lab modeled in MoveIt!.

[2] https://github.com/CGAL/cgal/
[3] http://wiki.ros.org/moveit

4.3 Real world execution

To control the robotic manipulator used, the Kinova JACO, the drivers provided by the manufacturer[4] were used. The trajectory generated by the MoveIt! framework, that corresponds to a group of points of the positions of the manipulator, is published to a *rostopic* that is subscribed by the robot drivers. The drivers communicate with the robot's hardware via an USB connection.

[4] https://github.com/Kinovarobotics/kinova-ros

Chapter 5

Analysis of Results

This chapter starts by presenting the results of the simulations, followed by the results obtained with the real manipulator.

5.1 Number of Simulated Annealing's Iterations

To obtain a robust grasp, enough grasps need to be sampled. For each iteration, the simulated annealing algorithm generates a random grasp obtained from the current grasp changing its parameters, which include translation, orientation and posture. The random distribution of the "jumps" is correlated with the temperature of the simulated annealing algorithm, which decreases with the number of iterations. The lower the temperature, the lower the probability of a larger change in the pose of the gripper occur.

The Figure 5.1 illustrates the lowest value of the cost function found with the number of iterations. The data was obtained from the average of three runs of the simulated annealing for different objects. As it's possible to analyze, most of the improvement of the cost function value happen before the 40.000 iterations. By iteration 40.000 the cost function has dropped 71.8%, while in iteration 70.000 the decrease was of 95.7%. The lowest value, for these runs, was achieved after the iteration 138.000.

5.2 Execution time

The execution time is a concern for the grasp planning. Due to the great number of grasps analyzed the execution time can greatly exceed the reasonable execution time expected for a robot to plan a grasp of an object.

No current algorithm allows the synthesis of a complex grasp in real time, that is, in a time similar to the interval taken by humans to plan a grasp (that can be in the magnitude of a few hundreds milliseconds). Instead, current solutions take several dozens of seconds for known objects [2] or up to several hundreds of seconds for novel objects [11].

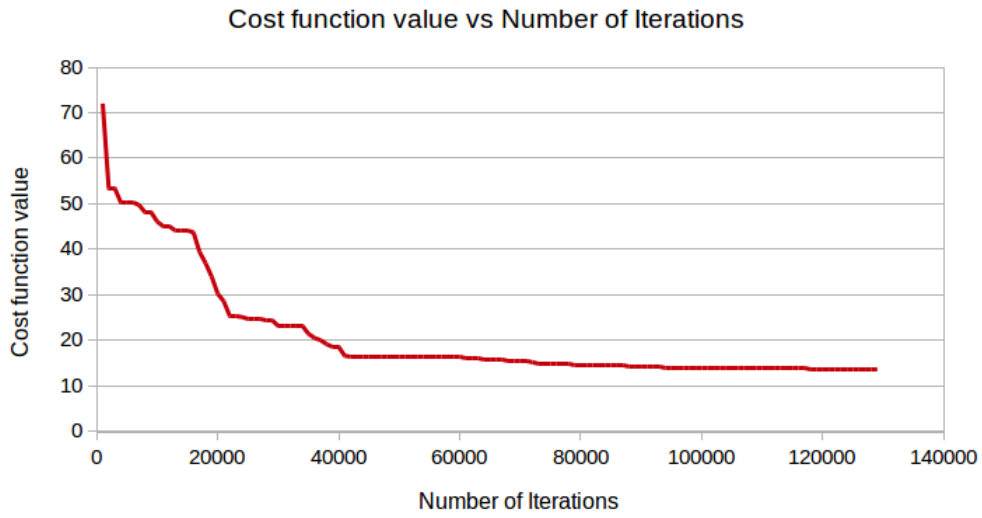


Figure 5.1: Decrease of the cost function with the number of iterations of the Sim. Ann.

The execution time is always linked to the platform used to execute the algorithm, and if the implementation is able to take full advantage of that platform. The results presented bellow were obtained from a consumer laptop system whose CPU was an Intel® Core™ i7-4700HQ CPU @ 2.40 GHz, and with a single-thread implementation. It's expected that the execution time would drop significantly with a faster processor and a multi-threaded implementation.

As expected, the execution time of the algorithm has a linear correlation with the number of iterations, as well as with the number of top K grasp pairs analyzed. That means, that each iteration (i.e. the generation of a random grasp and its analysis) takes about the same time.

5.2.1 Execution time results

Figure 5.1 depicts the execution time as a function of the number of iterations of the simulated annealing algorithm, for a value of K equal to 3. The execution time increases linearly with the number of iterations.

Figure 5.3 illustrates the execution time with the variation of K, for 20.000 iterations. The execution time also increases linearly with the value of K.

Execution time calculation

A rude approximation of the execution time can then be calculated with the expression:

$$T_e = (n_h + K \cdot n_r) \cdot t_i \tag{5.1}$$

where n_h is the number of iterations for the hander grasp, n_r the number of iterations of the receiving grasp, and t_i the average execution time for each iteration on that specific system. For the platform where the test were performed, $t_i \approx 0.00083$s.

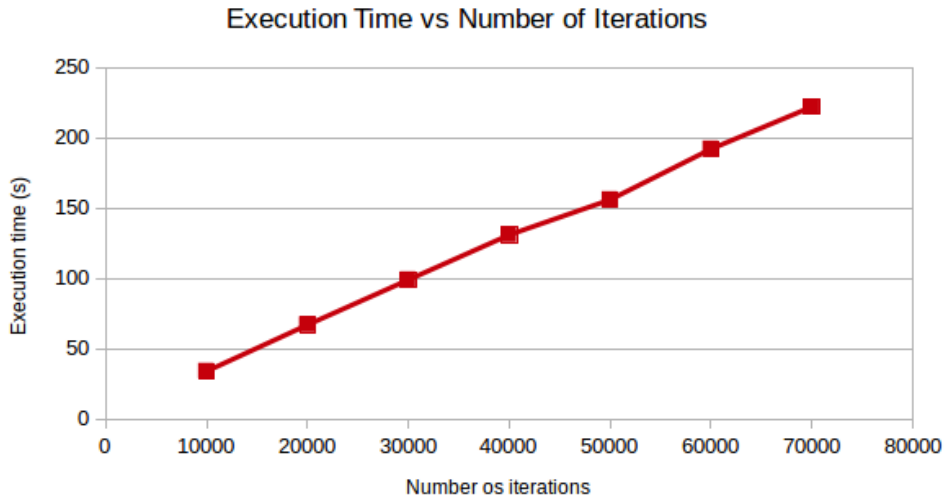


Figure 5.2: The execution time as a function of the number of iterations per run for K = 3.

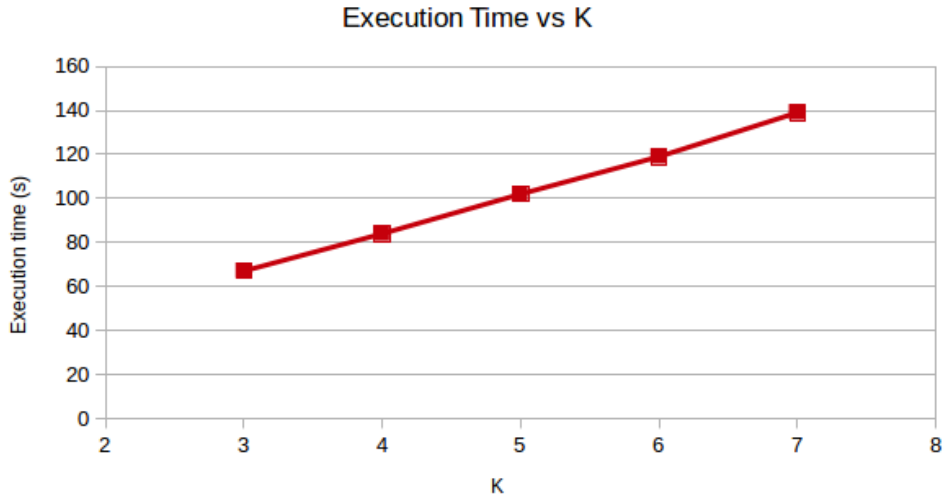


Figure 5.3: The execution time as a function of K for a fixed 20.000 iterations.

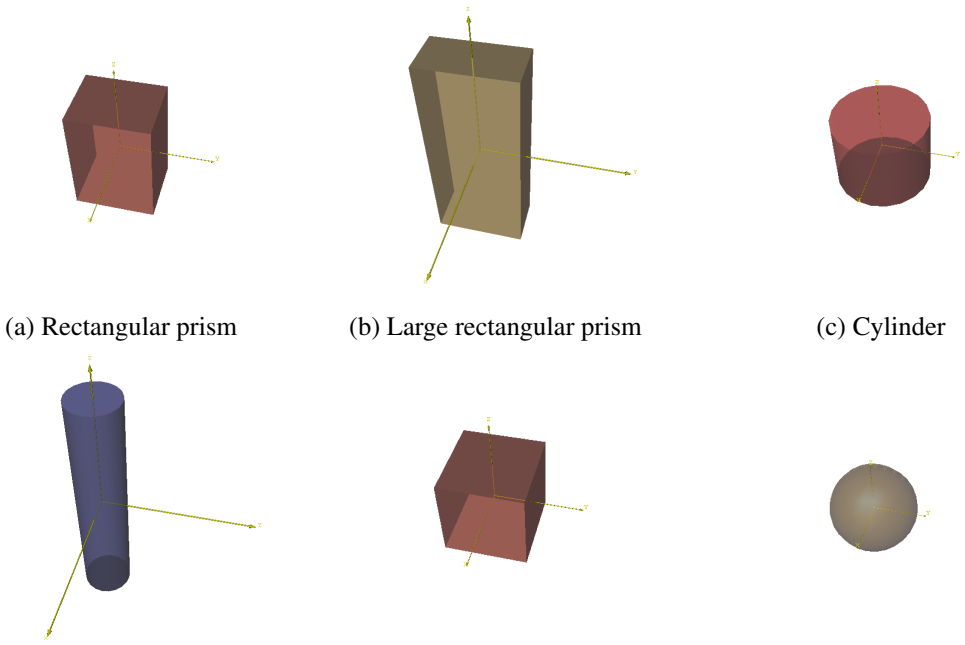
5.3 Simulation results

5.3.1 Objects used

For simplification purposes, the solution was tested with simple shaped objects. The objects used and their dimensions, in millimeters, are presented in Table 5.1. These objects used are illustrated in Figure 5.4.

Table 5.1: Objects used and their dimensions in millimeters.

	object	height	width	lenght	diameter
(a)	rectangular prism	57	71	99	-
(b)	rectangular prism	50	97	225	-
(c)	cylinder	65	-	-	86
(d)	cylinder	250	-	-	45
(e)	cube	73	73	73	-
(f)	sphere	-	-	-	76



(a) Rectangular prism (b) Large rectangular prism (c) Cylinder

(d) Large cylinder (e) Cube (f) Sphere

Figure 5.4: Objects used in the simulation.

5.3.2 Grasp pairs ranking validation

In order to validate the ranking method for the grasps pairs, three human subjects ranked 5 different grasps pairs, for 3 different objects. Each human subject was asked to sort five grasps pairs of the same object, from the one with most probability of success, to the one with least. Then these ranks were compared with the ranking given by the algorithm, using ρ, the Spearman's rank correlation

coefficient [20]. This measure gives the statistical dependence between the ranking of variables. It's value varies between $[-1, 1]$, depending on the degree of correlation of the two ranks.

The result obtained are presented in Table 5.2.

Table 5.2: Correlation between the human's and the algorithm's grasp pairs ranking.

object	ρ average
rectangular prism	0.93
cylinder	0.90
large cylinder	0.87

The values close to one, allow to conclude that the method used by the algorithm to rank grasps according to their quality is inline with the human ability to differentiate which one of two grasp pairs is better.

5.3.3 Algorithm results

For the verification of the algorithm, a total of 108 grasp pairs were analyzed by three different human subjects. The grasp pairs resulted from the execution of the proposed algorithm, and the 3D model of the scene was saved for each one them. In the scene was present the object, and the two grippers, identified with different colors. The subject could zoom and rotate the 3D scene in order to verify all the details of the pose of the grippers, allowing to verify collisions and invalid poses.

The subjects were asked to rate the underlying handoff operation correspondent to each grasp pair, as a success or fail. For the failed grasps pairs, the main reason of failure was also annotated. The reason for the failure of the handoff given the grasp pair's pose can be: invalid hander grasp, invalid receiver grasp, collision on the approach of the receiver gripper, or collision on the retracting of the hander gripper.

For these tests, the number of iterations used was 70.000, which represents a trade-off between the final energy function value (and consequently the quality of the grasp) and the execution time. As discussed in section 5.1, this number of iterations results in a value that, in average, corresponds to 95.7% of the final value of the cost function. The value of K was 6, which means that 6 grasp pairs are obtained at the end of the execution of the algorithm. The success rates for different objects can be found in Table 5.3.

The overall success rate was 78.4%.

Success rate and position on the rank

The success rate of the grasp pairs according to their position on the ranking is presented in Figure 5.5 and in Table 5.4. These results, represent the percentage of grasp pairs for the different objects that were classified as valid, according to their position on the final rank obtained from the developed algorithm.

Table 5.3: Success rate of the grasp planning for different objects.

object	success rate
large rectangular prism	83.3%
sphere	72.2%
small rectangular prism	59.3%
small cylinder	77.8%
cube	79.6%
large cylinder	98.1%
total	**78.4%**

The success rate of the grasps pairs decreases with their position up the ranking, which allows to confirm the results from the Subsection 5.3.2. It's then possible to conclude that the heuristics introduced to sort the grasps pairs according to their quality, are valid.

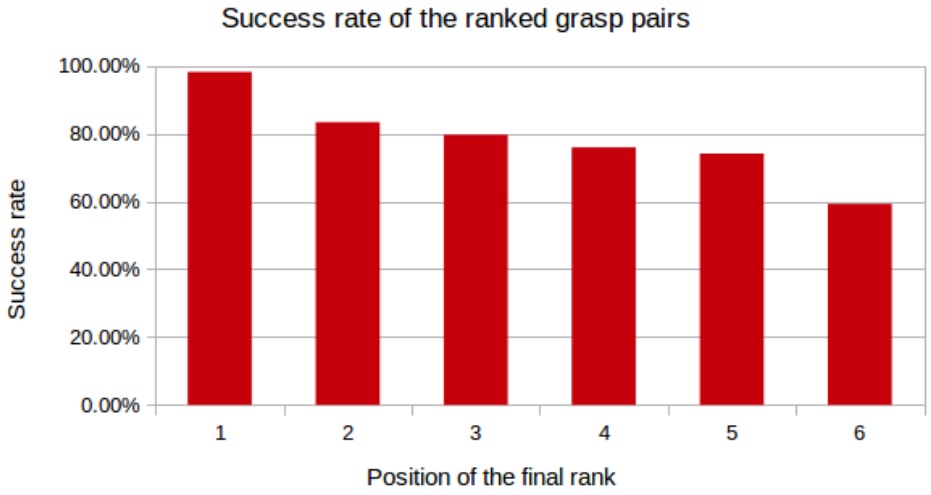


Figure 5.5: Success rate of the grasp pairs according to their position of the rank.

Table 5.4: Success rate of the grasp pairs according to their position on the rank.

position on rank	success rate
1st	98.15%
2nd	83.33%
3rd	79.63%
4th	75.93%
5th	74.07%
6th	59.26%

To verify the agreement between the raters, the Fleiss's κ was calculated as an inter-annotator agreement measurement [21]. It varies in the range $[-1,1]$ according to the agreement between

the raters, being the value 1 total agreement and a value lower than zero an agreement worse than the expected agreement if the ratings were random.

In Table 5.5 is presented the average κ considering only the validity of the grasp pairs, and the value of κ taking into account the reason given for the cause of failure.

Table 5.5: Fleiss's kappa of the final ratings

	Fleiss's κ
Valid/Fail only	0.705
Considering failure reason	0.648

Although, having a high value, the agreement is not total because is some cases it can be hard to predict the success of the handoff from the pose of the grippers in the simulator. When considering the reason of failure the agreement between the raters decreases slightly, because on some of the failed grasp pairs, there are multiple reasons for the handoff to fail and can be difficult to select the main cause of failure.

5.3.4 Baseline

As a baseline to comparison of the obtained results, the grasps generated by the method described by Ciocarlie *et al.* [11] were rated the same way the grasps generated by the developed method were. A total of 36 grasp pairs, for 6 different objects were rated by 3 different human subjects. The overall success rate for the baseline was 44.4%.

Table 5.6: Comparison between the developed method and the baseline

	success rate
developed method	**78.4%**
baseline	44.4%

For the cases tested, the algorithm developed in this work has a much higher probability of a successful handoff than the algorithm described in [11], that served has a base for the development of the work presented in this document. Even though the work developed by Ciocarlie *et al.* was not designed with the handoff of objects in mind, it was used as a comparison because it offers a solution for the grasp planning of novel objects for any kind of gripper, features that the previously developed works in the handoff of object between manipulators don't offer.

The Fleiss's κ was also calculated as an inter-annotator agreement measure, for the baseline results. These values are presented in Table 5.7. The decrease in the inter-annotator agreement, may indicate that there's more failure reasons for the grasp pairs, and when failures happen it can be hard to nominate the main cause of failure.

Table 5.7: Fleiss's kappa for baseline ratings

	Fleiss's κ
Validity agreement	0.600
Considering failure reason	0.429

5.4 Real Robot Results

The final tests were performed on a real robotic manipulator. A total of 36 first grasps, rated as valid by all the raters, were performed with the JACO robotic manipulator in 4 different objects. For each object, three different grasps poses were tested, and for each grasp pose three tries were performed. Due to time and infrastructure constrains, it wasn't possible to test the handoff of objects between two manipulators. Instead, only the execution of the hander grasps was tested, given that it is the scope of this work.

In Table 5.8 the success rate of the execution of the hander grasps for different objects is presented. The overall success rate obtained was 83.4%.

Table 5.8: Experimental results of the performed grasps with the real robot

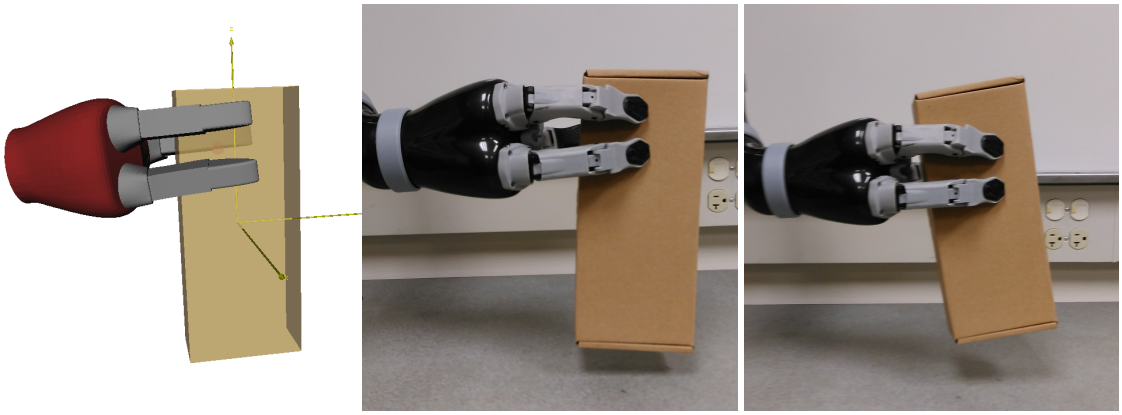
object	success rate
cube	77.8%
cylinder	100%
sphere	66.7%
parallelepiped	88.9%
total	**83.4%**

The cases where the robot failed to grasp the object, were either due to slipping or the imprecision of the positioning of the arm. Some of this situations were justified by the use of an under-actuated gripper.

5.4.1 Under-actuated gripper's grasps

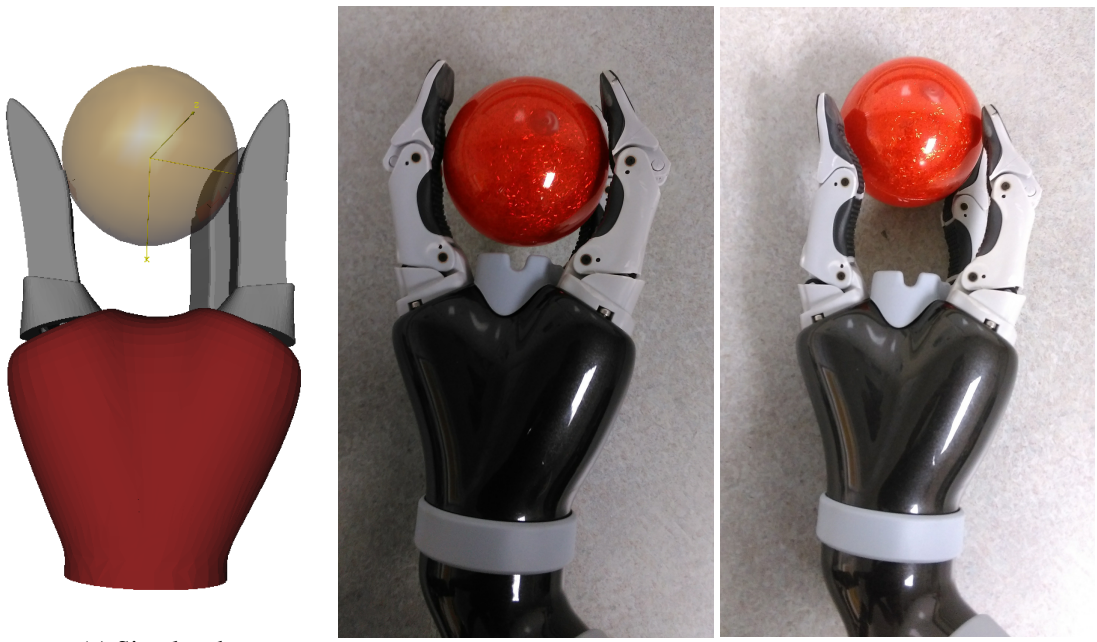
In some cases, the final stable grasp on the real manipulator differed significantly from the planned grasp, that resulted from the simulation. This happens because there's no control over the under-actuated joints, and these adapt to the shape of the object creating sometimes situations where the object slides to a stable position, dissimilar from the planned one. Nevertheless, the tests on the real robot show that, even not taking into account the flexibility of the under-actuated fingers of the gripper during the simulation, the generated grasps still successfully grasp the objects.

Figures 5.6, 5.7 and 5.8 present examples of the simulation of the planned grasp and different execution tries of that same grasp.



(a) Simulated (b) Execution 1 (c) Execution 2

Figure 5.6: Example of the simulated grasp and the final grasps for a rectangular prism on an under-actuated gripper.



(a) Simulated (b) Execution 1 (c) Execution 2

Figure 5.7: Example of the simulated grasp and the final grasps for a sphere on an under-actuated gripper.

(a) Simulated (b) Execution 1 (c) Execution 2

Figure 5.8: Example of the simulated grasp and the final grasps for a cylinder on an under-actuated gripper.

Chapter 6

Conclusion and Future Work

6.1 Summary

The work elaborated in this thesis project focused on the development of a new method for planning grasp poses for the hander manipulator in object handoff operations.

6.2 Contributions

With the goal of developing a solution for the generation of grasps that maximize the probability of a successful handover between two robotic manipulators, it was proposed an algorithm that leverages the use of heuristics and physics based metrics, to synthesize grasp poses.

The use of heuristics is justified by the high-dimensionality of the search space. Because an exhaustive analysis of each state, would lead to an unbounded execution time, simple heuristics that are based on virtual contact points are used. The concept of eigengrasps, first introduced by [11], is also employed.

The use of physics based metrics to evaluate the quality of the grasp is limited to the grasp candidates resulting from the search with the heuristics. These more reliable metrics are based on the grasp wrench space, and give direct evaluation of the robustness of the grasps.

The algorithm was implemented using the framework *GraspIt!*, that allows the simulation of grasps. This implementation allowed obtaining simulation results, which were first analyzed by human subjects, and then tested on a real manipulator.

The results acquired show that the proposed method can successfully be used to generate grasps poses for the exchange of objects between robotic manipulators.

6.3 Future work

Some modifications can be introduced to the developed solution, such as, the integration with the whole pipeline that allows two manipulators to exchange an object. A portion of the pipeline aiming at the exchange of objects between manipulators that was not implemented in this project,

because it wasn't part of the defined scope, includes the optimization of the placement of the object for the receiver robot to grasp it. One example of a solution is the work being developed by Bell *et al.* [22], that will possibly integrate the work developed in this project.

The re-planning of the receiving grasp pose, after the execution of the hander grasp and knowing the position of the object presented by the hander manipulator can be based on the developed solution. The heuristics, in conjunction with the physics based metrics, presented in this work can be exploited to this end.

Given that the 3D mesh model of the object has to be available for the simulating the grasp, one of the logical next steps would be to come up with a solution to generate this models. This task could be approached using different techniques, for instance, obtaining a complete model of the object using RGB-D images from different angles [23] or using NN for predicting the 3D model from a single point cloud, as the work presented in [9].

References

[1] J. P. Saut, M. Gharbi, J. Cortés, D. Sidobre, and T. Siméon. Planning pick-and-place tasks with two-hand regrasping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4528–4533, October 2010. doi:10.1109/IROS.2010.5649021.

[2] Weiwei Wan and Kensuke Harada. Achieving High Success Rate in Dual-arm Handover Using Large Number of Candidate Grasps. *Advanced Robotics*, pages 1111–1125, 2016. URL: https://drive.google.com/file/d/0Bwgh40s_zDJ1TTVDd1pNNy05ZDA/view.

[3] Beatriz León, Antonio Morales, and Joaquin Sancho-Bru. Robot Grasping Foundations. In *From Robot to Human Grasping Simulation*, volume 19, pages 15–31. Springer International Publishing, Cham, 2014. DOI: 10.1007/978-3-319-01833-1_2. URL: http://link.springer.com/10.1007/978-3-319-01833-1_2.

[4] O. J. Rubio, K. Huebner, and D. Kragic. Representations for object grasping and learning from experience. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1566–1571, October 2010. doi:10.1109/IROS.2010.5648993.

[5] Andrew T. Miller, Steffen Knoop, Henrik I. Christensen, and Peter K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003. URL: http://ieeexplore.ieee.org/abstract/document/1241860/.

[6] Alexander Herzog, Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, Jeannette Bohg, Tamim Asfour, and Stefan Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2):51–65, January 2014. URL: http://link.springer.com/10.1007/s10514-013-9366-8, doi:10.1007/s10514-013-9366-8.

[7] Nikolaus Vahrenkamp, Dmitry Berenson, Tamim Asfour, James Kuffner, and Rüdiger Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2464–2470. IEEE, 2009. URL: http://ieeexplore.ieee.org/abstract/document/5354625/.

[8] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, May 2015. doi:10.1109/ICRA.2015.7139793.

[9] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *arXiv:1703.09312 [cs]*, March 2017. arXiv: 1703.09312. URL: http://arxiv.org/abs/1703.09312.

[10] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. URL: http://journals.sagepub.com/doi/abs/10.1177/0278364914549607.

[11] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and Systems Manipulation Workshop-Sensing and Adapting to the Real World*. Citeseer, 2007. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.1332&rep=rep1&type=pdf.

[12] Olivier de Weck. Simulated Annealing. 2010. URL: http://mit.espe.edu.ec/courses/engineering-systems-division/esd-77-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec10.pdf.

[13] Lester Ingber. Very Fast Simulated Re-Annealing. 1989. URL: https://www.researchgate.net/profile/Lester_Ingber/publication/2836339_Very_Fast_Simulated_Re-Annealing/links/09e41505a243da2e54000000.pdf.

[14] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, May 1992. doi:10.1109/ROBOT.1992.219918.

[15] A. T. Miller and P. K. Allen. Examples of 3d grasp quality computations. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1240–1246 vol.2, 1999. doi:10.1109/ROBOT.1999.772531.

[16] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V. Dimarogonas, and Danica Kragic. Dual arm manipulation - A survey. *Robotics and Autonomous Systems*, 60(10):1340–1353, October 2012. URL: http://linkinghub.elsevier.com/retrieve/pii/S092188901200108X, doi:10.1016/j.robot.2012.07.005.

[17] A. T. Miller and P. K. Allen. Graspit! A versatile simulator for robotic grasping. *IEEE Robotics Automation Magazine*, 11(4):110–122, December 2004. doi:10.1109/MRA.2004.1371616.

[18] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. *arXiv:1404.3785 [cs]*, April 2014. arXiv: 1404.3785. URL: http://arxiv.org/abs/1404.3785.

[19] Weiwei Wan and Kensuke Harada. Regrasp Planning using 10,000s of Grasps. *arXiv:1705.09400 [cs]*, May 2017. arXiv: 1705.09400. URL: http://arxiv.org/abs/1705.09400.

[20] Wayne W. Daniel. *Applied nonparametric statistics*. PWS-Kent Publications, 2nd edition edition, 1990. Google-Books-ID: 0hPvAAAAMAAJ.

[21] Joseph L. Fleiss. Measuring Nominal Scale Agreement Amonf Many Raters. 1971. URL: http://www.wpic.pitt.edu/research/biometrics/Publications/Biometrics%20Archives%20PDF/395-1971%20Fleiss0001.pdf.

[22] Neil R. Bell, Tim Oates, and Cynthia Matuszek. Discovering Morphology from Action Observation. In *Workshop on Heterogeneity and Diversity for Resilience in Multi-Robot Systems*, June 2017. Department of Computer Science and Electrical Engineering University of Maryland, Baltimore County. URL: http://iral.cs.umbc.edu/Pubs/BellRSS2017_heterogeneity-workshop.pdf.

[23] Johann Prankl, Aitor Aldoma, Alexander Svejda, and Markus Vincze. RGB-D object modelling for object recognition and tracking. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 96–103. IEEE, 2015. URL: http://ieeexplore.ieee.org/abstract/document/7353360/.