Adaptive Grasping Planning:
A Novel Unified and Modular Grasping Pipeline Architecture

Por

João Pedro Carvalho de Souza

Orientador: Doutor José Boaventura Ribeiro da Cunha

Co-orientador: Doutor António Paulo Moreira

Co-orientador: Doutor Luís Freitas Rocha

Tese submetida à

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

para obtenção do grau de

DOUTOR

em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no

DR – I série–A, Decreto-Lei n.º 74/2006 de 24 de Março e no

Regulamento de Estudos Pós-Graduados da UTAD

DR, 2.ª série – Deliberação n.º 2391/2007

Adaptive Grasping Planning: A Novel Unified and Modular Grasping Pipeline Architecture

Por

João Pedro Carvalho de Souza

Orientador: Doutor José Boaventura Ribeiro da Cunha

Co-orientador: Doutor António Paulo Moreira

Co-orientador: Doutor Luís Freitas Rocha

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
DOUTOR
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série–A, Decreto-Lei n.º 74/2006 de 24 de Março e no
Regulamento de Estudos Pós-Graduados da UTAD
DR, 2.ª série – Deliberação n.º 2391/2007

Orientação Científica :

Doutor José Boaventura Ribeiro da Cunha

Professor Associado com Agregação do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Doutor António Paulo Moreira

Professor Associado do
Departamento de Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia da Universidade do Porto

Doutor Luís Freitas Rocha

Investigador Sénior do
Centro de Robotica Industrial e Sistemas Inteligentes
Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

"In the middle of difficulty lies opportunity" | "No meio da dificuldade encontra-se a oportunidade."

Einstein (1879 – 1955)

"Success is going from failure to failure without losing enthusiasm | Sucesso é ir de fracasso em fracasso sem perder o entusiasmo"

Winston Churchill(1874 – 1965)

-

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
Mestrado em Engenharia Electrotécnica e de Computadores

Os membros do Júri recomendam à Universidade de Trás-os-Montes e Alto Douro a aceitação da dissertação intitulada "**Adaptive Grasping Planning: A Novel Unified and Modular Grasping Pipeline Architecture** " realizada por **João Pedro Carvalho de Souza** para satisfação parcial dos requisitos do grau de **Doutor**.

maio 2022

Presidente: **Nome Presidente do Júri**,
Direção do Mestrado em Engenharia Eletrotecnica e de Computadores do Departamento de Engenharias da Universidade de Trás-os-Montes e Alto Douro

Vogais do Júri: **Doutor XXX** ,
Professor XXX

Doutor José Boaventura Ribeiro da Cunha,
Professor Associado com Agregação do Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro

Doutor António Paulo Moreira,
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Doutor Luís Freitas Rocha,
Investigador Sénior do Centro de Robotica Industrial e Sistemas Inteligentes da Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

Adaptive Grasping Planning:
A Novel Unified and Modular Grasping Pipeline Architecture

João Pedro Carvalho de Souza

Submetido na Universidade de Trás-os-Montes e Alto Douro
para o preenchimento dos requisitos parciais para obtenção do grau de
Doutor em Engenharia Electrotécnica e de Computadores

Resumo — A preensão robótica persiste como um problema na indústria moderna que busca técnicas autônomas, rápidas de implementação e eficientes em cenários complexos. A implantação de uma estrutura de preensão modular e reconfigurável para robôs atendendo demandas reais é modesta, mesmo com várias metodologias propostas. Cientificamente, a comunidade de robôs carece de uma arquitetura de pipeline de apreensão bem estruturada e formalizada que organize abordagens que permitam a avaliação e a base para novos avanços. Ao oferecer este novo pipeline de compreensão, dotado de interface de usuário adequada e metodologias incorporadas, a comunidade científica terá à sua disposição uma estrutura baseada em software para fácil integração e teste de novas contribuições científicas. A indústria terá uma ferramenta intuitiva e poderosa capaz de resolver o problema de apreensão em diversos cenários.

Palavras Chave: Arquitetura de Software Modular, Planejamento de Preensão; Manipulação Robótica.

Adaptive Grasping Planning:

A Novel Unified and Modular Grasping Pipeline Architecture

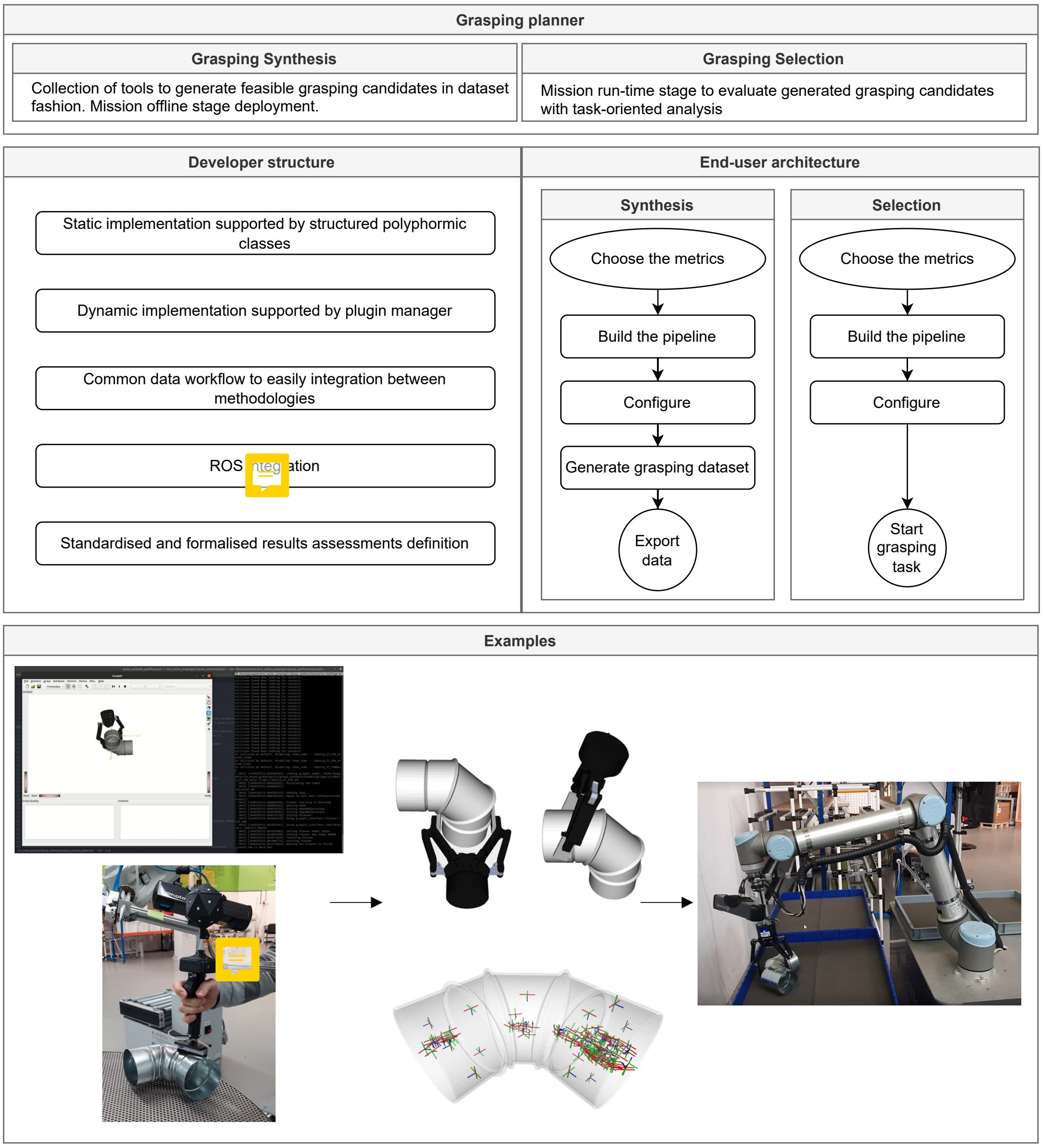
João Pedro Carvalho de Souza

Submitted to the University of Trás-os-Montes and Alto Douro
in partial fulfillment of the requirements for the degree of
Philosophiae Doctor in Electrical Engineering and Computers

Abstract — The robotic grasping persists as a problem in the modern industry that seeks autonomous, fast implementation, and efficient techniques in complex scenarios. The deployment of a modular and reconfigurable grasping framework for robots attending real demands is modest, even with several methodologies proposals. Scientifically, the robot community lacks in a well structured and formalized grasping pipeline architecture that organizes approaches allowing the evaluation and the base to new advancements. By offering this novel grasping pipeline, endowed with proper user interface and embedded methodologies, the scientific community will have at their disposal a base software structure for easy integration and testing of new scientific contributions. The industry will have an intuitive and powerful tool capable of solving the grasping problem in several scenarios.

Key Words: Modular Software Architecture, Grasping Planning; Robotic Manipulation.

Graphical Abstract



Agradecimentos

TODO Meis summo ex eam, facer animal voluptaria te quo. Ad vix ancillae reprehendunt, mei integre erroribus te, eum aliquam platonem corrumpit ex. Pri tractatos definiebas ex, per alii zril diceret ea. Te per purto nulla scripta, dolorum facilis eam ea, autem dicant putent ex vis. Ad sed prima ipsum, ex cum alia simul. In mel equidem verterem, dicta gloriatur definitionem eos ad. Mel id nulla inermis. Sed placerat reformidans at. Nibh eius eu vix, cum dolore eripuit evertitur eu, at velit phaedrum sed. At eum erant aperiri mediocrem, ius inani disputando ex. Usu aliquid moderatius disputationi te, dico ornatus moderatius an mei.

Per affert conclusionemque no. Quo et dico mazim vivendum, id senserit repudiare efficiantur vix, pri at consequat adversarium. Te quem mandamus qui, an enim laudem sed. An has omnis summo eleifend, ei sed eripuit similique.

UTAD, Nome

Vila Real, xx de xxxxx de 20XX

Publications

This Ph.D. thesis proposal results in following publications:

1. João Pedro Carvalho de Souza, Luís F. Rocha, et al. (2021). "Robotic grasping: from wrench space heuristics to deep learning policies". In: *Robotics and Computer-Integrated Manufacturing* 71, p. 102176. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102176. URL: https://www.sciencedirect.com/science/article/pii/S0736584521000594
2. João Pedro C. de Souza, Carlos M. Costa, et al. (2021). "Reconfigurable Grasp Planning Pipeline with Grasp Synthesis and Selection Applied to Picking Operations in Aerospace Factories". In: *Robotics and Computer-Integrated Manufacturing* 67, p. 102032. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.102032. URL: http://www.sciencedirect.com/science/article/pii/S073658452030243X
3. João Pedro Carvalho de Souza, António Amorim, et al. (2021). "Industrial Robot Programming by Demonstration using Stereoscopic Vision and Inertial Sensing". In: *Industrial Robot: the international journal of robotics research and application*. ISSN: 0143-991x. DOI: 10.1108/IR-02-2021-0043

Projects Deployment

The following R&D projects deployed this Ph.D. thesis proposal:

- **Fasten.** European Union's Horizon 2020 research and innovation programme under grant agreement 777096. http://www.fastenmanufacturing.eu/.
- **Mari4 Yard.** European Union's Horizon 2020 research and innovation programme under Grant Agreement 101006798. https://www.mari4yard.eu/.
- **Produtech 4S&C.** European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement with reference POCI-01-0247-FEDER-046102. http://mobilizadores.produtech.org/en/produtech-4-s-c
- **Produtech SIF.** European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement with reference POCI-01-0247-FEDER-024541. http://mobilizadores.produtech.org/en/produtech-sif

Índice geral

Índice de figuras

Glossário, acrónimos e abreviaturas

Glossary

Acronym list

PbD Programming-by-Demonstration

LbD Learning-by-Demonstration

DL Deep Learning

CNN Convolutional Neural Network

FCN Fully Convolutional Network

SL Supervised Learning

RL Reinforcement Learning

DRL Deep Reinforcement Learning

MDP Markov Decision Process

POMDP Partially Observable Markov Decision Process

SANN Simulated Annealing

SVM Support Vector Machine

LbD Learning by Demonstration

DLSR Dictionary Learning and Sparse Representations

CGD Cornell Grasp Dataset

RPN Region Proposal Network

GPSR Grasping Prediction Success Rate

GPT Grasping Prediction Time

CPU Central Process Unit

GPU Graphics Processing Unit

Run-GPT Run-Time Grasping Prediction Time

Offline-GPT Offline Grasping Prediction Time

GPU-GPT GPU Grasping Prediction Time

CPU-GPT CPU Grasping Prediction Time

GRSR Grasping Reaching Success Rate

GHSR Grasping-Holding Success Rate

HGSR Handling Grasping Success Rate

GQ-CNN Grasp Quality Convolution Neural Network

IoT Internet of Things

DOF Degree of Freedom

ICP Interest Contact Point

ICR Interest Contact Region

COG Center of Gravity

COBB Center of Bounding Box

ROS Robot Operation System

TCP Transmission Control Protocol

DRL Dynamic Robot Localisation

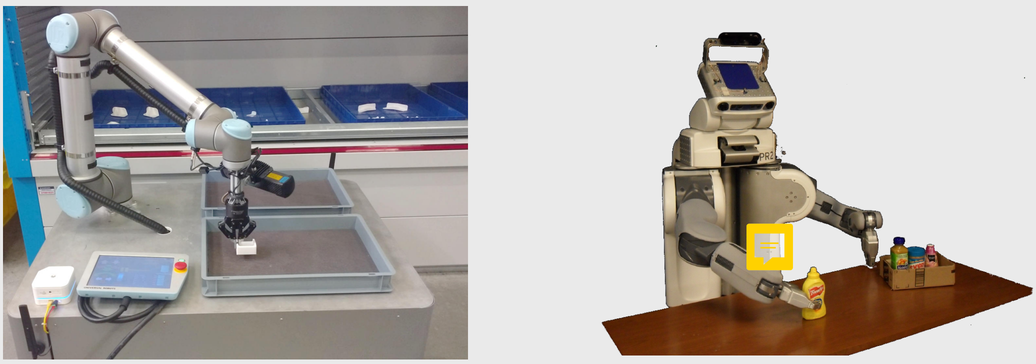
OR Object Recognition Package

GUI Graphical User interface

1 Introduction

1.1 Context

The robotic grasp is an important and challenging task that is still today the focus of several works. Although this function is intuitive and mastered by humans, for robots, it is a contemporary and imperative issue. The range of applications is wide, e.g., from bin-picking in the industry's logistics to a delicate and accurate human-machine interaction in domestic and collaborative robots applications.



(a) Industrial application.

(b) Domestic employment.

Figure 1.1 – Robotic grasping in different contexts.

In the early days, robotic manipulation was resumed as a context of human teleoperation (Bejczy 1980). More recently, it evolved to Programming-by-Demonstration (PbD) (Ferreira et al. 2016) and to the beginnings of Learning by Demonstration (LbD) (Suleman et al. 2011) even though the offline programming had been consolidated in the industrial robot programming (Souza, Castro, et al. 2020; Castro et al. 2020). The first studies focused on the analytical grasping modelling concerning the stability and equilibrium (Dizioğlu et al. 1984; Nguyen 1987b; Nguyen 1987a; Ponce et al. 1995a; Jia-Wei Li et al. 2003), besides metrics to evaluate them (Ferrari et al. 1992; Bicchi and Kumar 2000; Máximo A. Roa et al. 2015). A large number of papers regarding this theme was established, achieving success grasping in specific cases. Nonetheless, the complexity rise when more general solutions are pursued: gripper's design and number of fingers (J. Chen et al. 2020); previous knowledge of workpieces shape and properties (Babin et al. 2019; Babin et al. 2019; Björnsson et al. 2018), e.g., object-agnostic grasping or not; and cluttered or occluded scenes (D'Avella et al. 2020).

Figure 1.2 – Amazon robotic picking challenge winner of 2017 (Zeng, S. Song, Yu, et al. 2019). The Amazon robotic picking challenge is an example effort to promote development the warehouse robotic grasping, such as the IROS challenge (IROS 2020)

Subsequently, guided by computer processing and learning algorithms advancements, works like (Saxena et al. 2008) proved that it is possible to create learning policies to grasping objects, particularly for object-agnostic grasping scenarios. Nowadays, the advent of Deep Learning (DL) and the interesting results achieved in computer vision tasks motivate researchers to explore its capacity to grasp detection (Lenz et al. 2015; Redmon et al. 2015; Kumra et al. 2017; Watson et al. 2017; Chu et al. 2018; Asif et al. 2018; L. Chen, Huang, Y. Li, et al. 2020; Guo et al. 2017; Gariepy et al. 2019; Mousavian et al. 2019; Ghazaei et al. 2019; Pas, Gualtieri, et al. 2017; L. Chen, Huang, and Meng 2019; Mahler, Pokorny, et al. 2016; Mahler, Liang, et al. 2017; Mahler and Goldberg 2017; Mahler, Matl, X. Liu, et al. 2017; Mahler, Matl, Satish, et al. 2019; Y. Song et al. 2020).

Even with exciting discoveries and results successfully deployed in specific use cases, the robotic grasping still does not have a feasible generalisation solution that comprises the modern industry demands of fast design and easy deployment. It is important to note that, the results assessments standardisation is also a problem. Actually is challenging to study and choose a grasping methodology since several proposals use different results analyses that typically are not in accordance with the application necessities.

1.2 Motivation

Since a complete and generic solution is unreachable until now, there is exists a lack in the deployment of a modular and flexible grasping framework for robots attending real industry demands and a well structured and formalised architecture in which organises approaches allowing the evaluation and the base to new advancements in science.. Therefore, the main Ph.D. question relies in:

> "Several approaches achieved interesting results in specific and/or controlled scenarios. However, how to choose and deploy them according to industry demands?"

Afterwards, when applying the grasping solution, engineers and researchers still have difficulty comparing the achieved results since the grasping parametrisation and evaluation demand big efforts. Current state-of-art deploy different metrics that, in some cases do not reflex the grasping complexity, which can involve from graspable object detection to stability estimation. Thus, the second PhD question is:

> "How evaluate a grasping methodology in a standard fashion allowing easy comparison between different techniques?"

1.3 Goals

Aiming to answer the questions described in Section 1.2, the present thesis is summarised into two main objectives:

- Design of an innovative modular software architecture able to be hierarchy modified and reconfigured, structured in a pipeline flow which to organize the approaches' studies and evaluation. Therefore, developers can easily integrate new methodologies and end-users can set a pipeline of heuristics according to the task exigences and, choose between the best solution options between methodologies, i.e., the user will just need to pick the method and set its parameters without implementing the methods by self;
- Formalisation and standardisation of the grasping evaluation idea since a grasping problem involves perception, planning, and control. Therefore, a clear standard could improve the methodologies' comparability since each step of the procedure affects the grasping performance. These criteria are applied in state-of-art literature and in the current work.

Other specific developments and contributions are highlighted as following:

- Discussion and review of state-of-the-art proposals regarding grasping solutions and how they evolved over the years;
- Definition of a grasping dataset standard;
- Development of grasping hardware and firmware to support grasping by demonstration applications.

1.4 Thesis Organisation

The remainder of the thesis is organised as follows: Chapter 2 shows and discusses the related work, from different grasping representations (since the characterisation is the most important step in any grasping planning approach) to a review of analytical, learning and deep Learning methods. Chapter 3 presents a grasping evaluation discussion followed by the proposal of formalisation and standardisation of the grasping evaluation idea. Latter, the proposed standard is applied to state-of-art literature. The modular grasping pipeline proposal is described in Chapter 4 with its sub-systems and hardware structures. Chapter 5 shows the proposal evaluation and test assessments. In the end, Chapter 6 presents the conclusion and the future work discussion and suggestions.

2 Related Work

Given the vast range of works in the robotic grasping, this chapter discusses and reviews state-of-the-art proposals of grasping solutions and how they evolved over the years. This chapter build the basis to the thesis proposals.

The remainder of the chapter is organised as follows: Section 2.1 shows and discusses the different grasping representations since the characterisation is the most important step to any grasping planning approach. A review of Analytical, Learning and Deep Learning methods is given in Sections 2.2.1, 2.2.2, and 2.2.3, respectively.

2.1 Grasping Representation

There are several approaches regarding how to represent a grasping, i.e., how to describe and characterise it. This subject is a valuable step to the development of grasping planning or grasping detection algorithms and will be discussed in this section.

Some approaches model the physical interaction between the active pairs and evaluate its equilibrium and stability performance with the wrench space analyses and Coulomb's law in multi-fingered grasping (Figure A.5). It is a prevalent approach in analytical methodologies (Y.-H. Liu et al. 2004; El-Khoury et al. 2009; Andrew T Miller et al. 2004; Andrew T et al. 2001) and present in Learning and DL policies (Mahler, Pokorny, et al. 2016; Mahler, Liang, et al. 2017; Mahler and Goldberg 2017; Mahler, Matl, X. Liu, et al. 2017; Mahler, Matl, Satish, et al. 2019). Typically, it depends on the active pair's 3D shape (Point-Cloud, Voxel Grids, or 3D models) since the contacts need to be evaluated in an iteration algorithm or simulation. Well-recognised quality metrics are the Convex Hull's volume and the Epsilon radius of the grasping wrench space configuration proposed by Ferrari et al. in (Ferrari et al. 1992) (Figure 2.2). Since a complete wrench space analyses demand effort, a simplified modelling strategy is the antipodal restraints (Nguyen 1987b) represented by Figure 2.3, which is commonly used by proposals that employ two-finger grasping.

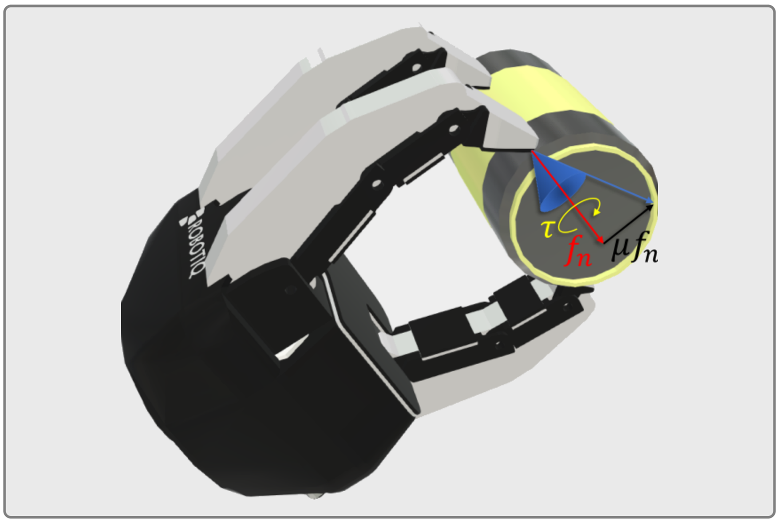


Figure 2.1 – Soft finger friction contact model.

In many applications, the model shapes unfamiliarity is a significant drawback, and the ability to exceed the grasping to novel objects is necessary. This capability is also referred to as object-agnostic grasping. Decomposition heuristics and learning algorithms try to solve this issue. Even though, in most cases, the gripper topology generalisation is not considered. The point (Saxena et al. 2008) and the oriented rectangle grasping representations (Yun Jiang et al. 2011) (Figure 2.4) were the options of several works to find a grasping pose over an object. These metrics, used in Supervised Learning (SL) methodologies, commonly require a labelled ground-truth, as Cornell Grasp Dataset (CGD) (Lab n.d.) and Jacquard (*Jacquard dataset* 2018) datasets. The point representation, normally indicated by a 3D pose, has a lack of physical limitation descriptors, like gripper's width and orientation approach angle, which are included in the rectangle methodology, see Figure 2.4. The point representation metric only evaluates a distance between the detected grasping pose and the ground-truth, while the rectangle is also evaluated by the Jacquard threshold, i.e., the $Area(G \cap G^*)/Area(G \cup G^*)$, where G and G^* are the predicted and the ground-truth grasping rectangle.

Wrench Space Analyses

Each contact can be modelled by a wrench vector $\mathbf{w}$ composed of forces and torques. All contact wrenches associated mould the convex-hull configuration, which can evaluate a grasping equilibrium. In a case of planar multi-fingered grasp, a force-closure grasp has the wrench space origin included by its convex-hull geometry (left convex-hull of Figure 2.2), unlike a non-force closure (right convex-hull of Figure 2.2). The ϵ is an example of the quality value to define the best force-closure configuration. It represents the wrench vector's distance to the origin ($\{O\}$), which is the shortest, i.e., the worst wrench vector to support an external perturbation.

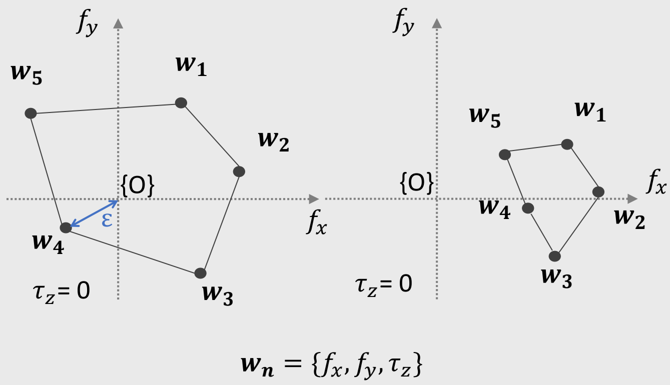


Figure 2.2 – Wrench space analyses representation

Following (Yun Jiang et al. 2011), Lenz et al. (Lenz et al. 2015) propose that the

6DOF-Rectangle could be simplified and confirm that this representation can be projected back to a 3DOF space. However, it was not explained this mapping. Later works (Redmon et al. 2015; Watson et al. 2017; Gariepy et al. 2019; Asif et al. 2018) used this approach and achieved an interesting grasping detection success rate.

Antipodal Grasping

Two-finger contact with friction can be defined as force-closure if, and only if, the line that connects each contact lays inside both friction cones. A non-force closure and a force closure antipodal grasping are represented by the left and right grasps of Figure 2.3, respectively.

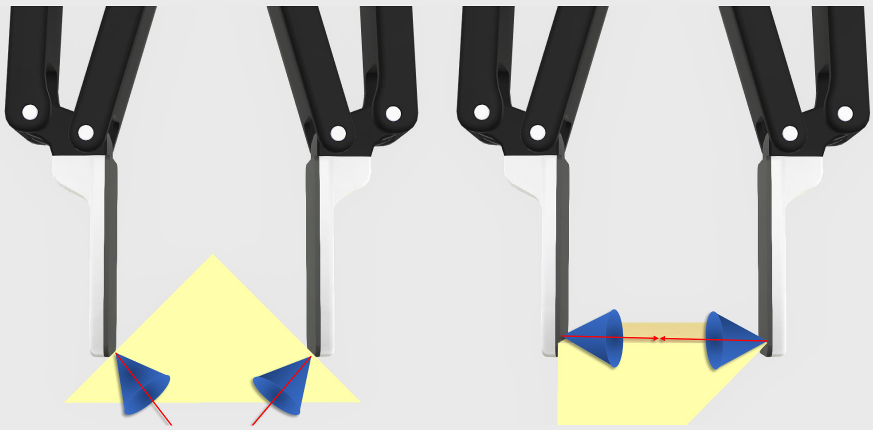


Figure 2.3 – Antipodal grasping restriction

Even if several state-of-the-art works use these image-based metrics to assess the efficiency of grasping detection, this approach is questionable (Guo et al. 2017; Gariepy et al. 2019; Mousavian et al. 2019; Ghazaei et al. 2019; Pas, Gualtieri, et al. 2017; Choi et al. 2018; L. Chen, Huang, and Meng 2019). Some authors (Gariepy et al. 2019; Ghazaei et al. 2019) affirm that the grasping is not completely defined like an object's classification in an image, i.e., there still exist many grasping possibilities which are not mapped in a ground-truth database. Despite the gripper's physical limitations of rectangle representation modelling, it is possible to note that the grasping performance does not reflect several works' high success detection rates. This approach does not consider the real physical interactions between active pairs,

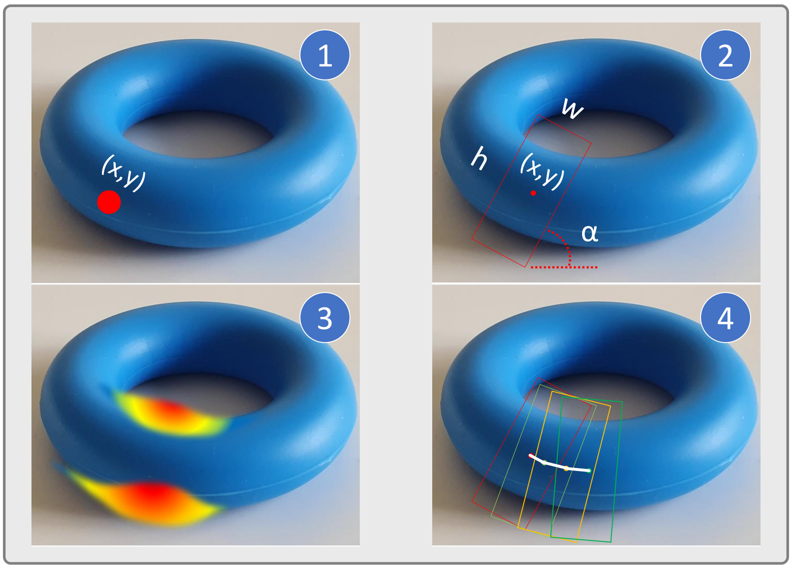


Figure 2.4 – Grasping representation in 2D images. Grasping point (1) describes a grasping position (x, y) without any physical considerations, while the rectangle representation (2) also represents the gripper's width (w), height (h), and orientation (α). The grasping belief map (3) models a spatial uncertainty in a grasp distribution fashion. The grasping path (4), described by a white line, leads the prediction of several possible rectangle grasp candidates.

e.g., force-closure properties and Coulomb's law. It also does not consider sensing noises and robot action's inconsistencies. Thus, Guo et al. (Guo et al. 2017) propose a hybrid deep learning architecture combining the visual rectangle representation and tactile sensing for robotic grasping detection. Their experiments indicate that tactile data improve the grasping detection task. Ghazaei et al. (Ghazaei et al. 2019) present the grasping belief maps to generate a set of grasping over an image without classifying the object whilst considering uncertainties (Figure 2.4). A similar approach of pixel-wise representation is also verified by (Zeng, S. Song, Welker, et al. 2018). Another continuous approach is the Grasp Path proposed by (L. Chen, Huang, and Meng 2019; L. Chen, Huang, Y. Li, et al. 2020) (see Figure 2.4). Asserting that the predicted comparison with the ground-truth database could eliminate other graspable candidates that are not included in the overlap threshold, the authors of (L. Chen, Huang, and Meng 2019; L. Chen, Huang, Y. Li, et al. 2020) formulated a path that leads to the set configuration of rectangular grasping poses.

In their tests, Grasp Path show to be less dependent on the Jacquard threshold.

2.1.1 Discussion

As it is possible to see in this section, grasping representation is an essential phase in grasping planning. The mathematical formulation and description, such as wrench space analyses, antipodal restriction, ϵ, and volume metrics are the basis for understanding physical phenomena between the active-pair interaction, e.g., basis widely used in 3D model grasping representations (3D-sensing and CAD simulations). However, these mathematical description techniques showed to be limited by several factors, e.g., high complexity implementation according to application demands and the necessity to know the gripper and graspable object characteristics. Therefore, these approaches typically work well in specific cases.

The 2D image grasping representation, as already stated, has a lack of fully physical grasping representation. This factor is prominent in point representation since the grasping is only described by its spatial coordinate. Differently, the rectangle representation also includes gripper properties, e.g. finger's width and height, but does not consider friction or any spacial uncertainty, such as the belief map's. The Grasp Path could also be an alternative since it guides several grasp hypotheses to contour the limited space search problem.

It is important to note that, in numerous proposals, the discussed cases are used, and the gripper structure generalisation is not considered (typically, only two fingers are investigated), showing the complexity of this task.

Heretofore, to the best of the authors' knowledge, the most promising representation is formulated by DexNet dataset proposed in [32-36]. The DexNet is constituted by 3D CAD models labelled with robust grasps for two-finger and suction grippers, i.e., the database was build considering analytical methodologies, iterative algorithm, and simulation interactions. They also included bin-picking and ambidextrous policies. This leads to the idea that the human grasping intuiting is more complicated than only visual representation.

2.2 Robotic Grasping Approaches

Since the advent of robotic operations, numerous proposals explore the grasping solution idea, from analytical to deep learning approaches. In summary, analytical methods showed to be the first solution to several proposals that achieved exciting results in specific cases. They also formulate the basis used in recent days. However, the main problem is the object grasping generalisation and complexity rise according to the application demand. Currently, the computer science field advances led to machine learning usage that shows to be a potential fashion to deploy effective grasping solutions, until now not reached but with promising results.

Therefore, the present section is an effort to categorise these strategies over the years and build a structured basis for new studies. This section is organised as follows: first, the analytical methods are explored in Section 2.2.1 following by the data-driven grasping policies, Section 2.2.2. As the deep learning methodology has been the study-case of several state-of-art proposals, this data-driven policy subcategory is presented apart in Section 2.2.3.

2.2.1 Analytical Methods

In the beginning, grasping planning was focused on analytical methods. This grasping class is based on a problem's mathematical model considering the kinematics and dynamics formulations. Well-accepted and suitable definitions and propositions regarding form- and force-closure grasping were first explored by (Dizioğlu et al. 1984; Nguyen 1987b; Nguyen 1987a). Subsequently, several proposals extend its grasping stability analyses like (Bicchi 1995; Ponce et al. 1995b; J.-W. Li et al. 2003) and the firsts algorithms were proposed, as (Y.-H. Liu 1999; Y.-H. Liu 2000; Ding et al. 2000). The readers are encouraged to review (Murray et al. 1994; Prattichizzo et al. 2016) for an embracing grasping analysis.

The analytical methods are applied in specifics cases, with the complexity rising according to the applications' demand. The disparity in mathematical modelling and the real operation is also an issue. Complete and generic analytical grasping solutions are presented by (Y.-H. Liu et al. 2004; El-Khoury et al. 2009) and elucidate these problems. Y.-H. Liu et al. (Y.-H. Liu et al. 2004) propose a complete analytical solution to find a 3D force-closure grasp, frictional or friction-less, for any type of object, including the one with a curved surface. The algorithm combines a local process with a recursive strategy of problem decomposition. First, it inserts n-contacts on the object and recursively tries to find a convex hull that includes the origin. When the search finds a local minimum, the algorithm sets the recursive decomposition method decomposing the problem in sub-problems. Results are analysed over numerical examples without a completely guarantee to find a feasible grasp; an issue is a computational complexity that needs to be evaluated according to the number of contacts used.

El-Khoury et al.(El-Khoury et al. 2009) addressed robust force-closure grasps for generic objects with n-finger hands. To achieve the goal, authors randomly generate $n-1$ fingers position and define the last finger position using the strictly negative linear combination of one of the first generated $n-1$ fingers wrench basis. In this way, the authors reached a faster algorithm, to the detriment of the success rate.

It is important to note that some analytical approaches only aim at estimating the stability of suitable grasps, sometimes called grasping synthesis. However, it is also necessary to determine the best grasp to perform the task. Some analytical methods treat this step as an optimisation problem ((Ciocarlie et al. 2009; Rakesh et al. 2018)) of some quality measurements. As listed by Máximo A. Roa et al. (Máximo A. Roa et al. 2015), a quality measurement can be categorised based on the contact points' position and the hand configuration. An extensive review of these quality methods is presented in (Máximo A. Roa et al. 2015). These measurements define the grasp analyses optimisation that, in several cases, do not guarantee the grasp determination because of the local minima problem.

Ciocarlie et al. proposed in (Ciocarlie et al. 2009) to embed the "Very Fast Simulated Re-Annealing" (Ingber 1989) (Kirkpatrick et al. 1983) optimisation algorithm, into *Graspit!* simulator (Andrew T et al. 2001), a widely used grasping planning environment (Morales et al. 2006; Souza, Carlos M. Costa, et al. 2021). With a 3D pair-wise model, this algorithm tries to reduce the distance of the gripper's contact points and the object's surface evaluating wrist pose and gripper/hand posture. As discussed by Ciocarlie et al. (Ciocarlie et al. 2009), the hand posture is defined by *eigengrasps*, a subspace of movement based on how human-generated hand postures. The *eigengrasps* reduces the hand's DOFs based on how humans select appropriate grasps and hand postures. Studies (Ciocarlie et al. 2009; Santello et al. 2002) show that humans simplify, unconsciously, the problem with a pattern in the movement.

Besides the interesting results, the discussed methods demand some unfeasible application efforts, e.g., the optimisation's convergence time could limit the run-time grasping decision process. Thus, the strategy to divide the grasping planning into offline (grasping synthesis) and run-time processes (grasping selection) are used (Souza, Carlos M. Costa, et al. 2021). Typically, the offline procedure generates a list of grasp candidates. The run-time selects the best grasp (under task-oriented capabilities) after matching the sensing data to a dataset. The major problem of these techniques is to know the objects' shape and the gripper's structure. Therefore exceed grasping of novel objects based on already known ones is not possible. This evaluation incorporates the sensing step into grasping planning rather than only perform object localisation and recognition.

Once knowing how to grasp spheres, cylinders, cones, and boxes with analytical methods, A. Miller et al. (A. Miller et al. 2003) investigate the idea to approximate an object to these primitive shapes. Jain et al. did the same (Jain et al. 2016) by using the point cloud directly. In this context, some approaches use object model decomposition in a set of primitives and execute the "grasping by parts". Therefore, the grasping synthesis is simplified, and heuristics methods, associated with analytical approaches as quality control, are proposed. Not only the shape definition is used, but the decomposition tree with superquadrics shapes (Goldfeder et al. 2007), Reeb graph (Aleotti et al. 2012), and Medial Axis Transformation (Przybylski et al. 2011) methods are examples trying to solve the related issue.

As discussed in the presented section, the analytical approach has a list of issues that can be summarised as high modelling complexity; high computational complexity; practical inconsistencies; and restricted assumptions. Therefore, the use of learning methods has gained attention, and its development is growing. This fact is verified by the crescent number of papers regarding this theme.

2.2.2 Data-Driven Grasping and Learning Policies

As presented in Section 2.1, in robotic grasping there is a lack of full representation of the input data, i.e., a partial data representation structure. For instance, a 2D image cannot map the truly grasping interaction, while the simulation could reduce this problem in the loss of applicability. Modelling all physical interaction and noise is unfeasible. However, this issue is not restricted to robots but also presented in humans. Humans can plan to grasp using a stereo-based image with eyes and based on previous experience (Castiello 2005). This simple motivation lead studies about learning methodologies to supply these demands, and some pioneer studies can be reviewed in (Oztop et al. 2001; Wheeler et al. 2002; Rezzoug et al. 2002).

Indirectly, the optimisation grasping techniques use data or previously known interaction to find a possible solution or refine it (Máximo A Roa et al. 2009). Therefore this kind of methodology can also be included in the data-driven category. However, looking for structured categorisation, this section will only discuss methods regarding classification and machine learning algorithms.

Pelossof et al. (Pelossof et al. 2004) explore the application of Support Vector Machine (SVM), trying to find a regression function to build a map between object shape, grasp parameters, and grasp quality. The authors face the problem of grasp representation in SL methodology. Therefore, it was proposed the use of the superquadric objects since these are easily characterised. This approach limits the grasping generalisation only to novel superquadric objects. The "GraspIt!" simulator was deployed to generate the database and propose as future work the superquadric combination to characterise more complex objects, later studied by (Goldfeder et al. 2007). The complexity of feature extraction, while considering different grippers, was also a pioneer study by Dini et al. (Dini et al. 2000), where a classic Neural Network was proposed to classify an object-agnostic grasping qualitatively. A more recent work (Pas and Platt 2018) use SVM to classify antipodal grasping directly from a point cloud and without recognising it. This paper also evaluates the performance in a densely cluttered environment. The same methodology was used in (Mahler, Liang, et al. 2017) but with a grasping proposed database. Mahler, Liang, et al. (Mahler, Liang, et al. 2017) also evaluate the use of the Random Forest SL algorithm motivated by the work of (Seita et al. 2016).

Saxena et al. (Saxena et al. 2008) was one of the first to explore the supervised grasping learning technique and target object 2D image-based features. They were also motivated by the fact that human object recognition is not related to grasping (Goodale et al. 1991). For this, the authors propose using logistic regression to find grasping point representations and were able to grasp a variety of novel household objects with a success rate of 87.8%. Based on this, Yun Jiang et al. (Yun Jiang et al. 2011) proposes the use of a two-stage SVM classification to detect rectangular grasping contesting the point representation (see Section 2.1). The first stage is faster and less accurate, while the second stage is more accurate with complex feature detection. Their studies motivate the grasping by an image that later evolves to deep learning policies (Section 2.2.3).

Trottier et al. (Trottier et al. 2017) use the rectangle strategy, RGB-D images, and Dictionary Learning and Sparse Representations (DLSR), showing a different strategy to SL procedure. The authors propose several architectures trying to avoid the big dataset needed for deep learning policies (discussion in Section 2.2.3). They achieved a state-of-art grasping detection rate, but the processing time was not qualified to perform the grasping.

The Reinforcement Learning (RL) applied to robotic grasping is the focus of studies (Rossler et al. n.d.; Baier-Lowenstein et al. 2007; Boularias et al. 2015; Platt 2007) that try to avoid the SL approaches shortcomings, e.g., the time-spend to build a labelled database and the limitation of grasping performance according to the supervisor/teacher ability to interpret the physical problem.

Typically this class of algorithms is based on trial-and-error approaches focusing on maximising a cumulative reward function. This concept allows an object-agnostic grasping procedure without environment restrictions modelling. However, the major drawback is the time-consuming learning experiments. Boularias et al. (Boularias et al. 2015) use the strategy to push the objects before grasping them in clutter scenes, which was modelled as a Markov Decision Process (MDP). A kernel-based RL methodologies were implemented, showing that pushing movements can improve grasping performance. Instead of using visual information, Platt (Platt 2007) investigates how to grasp using the contact relative motion model, i.e., using a force sensor as feedback, the algorithm tries to increment small displacement of the finger until reach grasp stability. Modelling it as a Partially Observable Markov Decision Process (POMDP), (Platt 2007) tries to solve the optimal control problem using RL methodologies. First, a simulated RL was used, which was later transferred to the real robot. This approach is more related to a refined grasping procedure than a grasp synthesis since an initial random grasping point needs to be a priori estimated.

Nowadays, the SL and RL methods were guided to DL methodologies based on the promising results of deep architectures applied in robotic task generalisation. Therefore, it is possible to define a distinctive category that will be presented in Section 2.2.3, the deep learning grasping.

2.2.3 Deep Learning Grasping Policies

The new deep learning policies and deep network architectures have leveraged computer vision detection tasks, as the object recognition problem proposed by ImageNet Large Scale Visual Recognition Challenge (*Large Scale Visual Recognition Challenge (ILSVRC)* n.d.). In this field, the advent of deep Convolutional Neural Network (CNN) (Krizhevsky et al. 2012), and the constant improvement of its architecture shown promising results. These results have been drawn the attention of robotic researchers that seek to apply the generalisation capability of these networks in the grasping issue, as can be seen by the competitor's proposals of Amazon Picking Challenge (Lawrence n.d.).

Lenz et al. (Lenz et al. 2015) was one of the first to investigate the DL in grasp planning. In their proposal, a grasp rectangle (Yun Jiang et al. 2011) is elected after a two-stage cascade detection learning network using RGB-D images. The first layer, less accurate, is responsible for learning a large number of direct features from the view, and the second layer selects the best grasp position using these features. The input image is gathered using a sliding window technique. However, this strategy compromises the real-time application, as reported by (Redmon et al. 2015; Guo et al. 2017; Chu et al. 2018). Lenz et al. (Lenz et al. 2015) verified that the DL improve the learning process since a hand-engineering feature modelling was not needed. Another precursor DL work was proposed by Redmon et al. in (Redmon et al. 2015) which applied the AlexNet (Krizhevsky et al. 2012) CNN architecture to grasping prediction. Redmon et al. (Redmon et al. 2015) explored the Transfer Learning concept between DL applications, latterly used by works (Kumra et al. 2017; Pas, Gualtieri, et al. 2017; L. Chen, Huang, and Meng 2019; Mahler and Goldberg 2017; Zeng, S. Song, Yu, et al. 2019; L. Chen, Huang, Y. Li, et al. 2020), and verified that this approach supports the training process preventing over-fitting due to the limited labelled database size. Their proposal uses a complete RGD image to direct regression to rectangle grasp. The strategy of replacing the blue channel with depth is also verified in subsequent works (Kumra et al. 2017; L. Chen, Huang, and Meng 2019; Y. Song et al. 2020) that adapt the image classification CNN architectures to the grasping problem. The authors affirm that adding an extra channel in these architectures avoids the pre-training phase with the image classification dataset. Therefore, the grasping architecture was pre-trained with object classification of (*Large Scale Visual Recognition Challenge (ILSVRC)* n.d.) and also evaluated by the hypothesis to classify an object before grasping it. The authors achieved an almost 85% of detection success rate (see Table 3.1), however when the proposal is exceeded to real grasping a reduction in the efficiency is related, as can be verified by the results achieved in (Watson et al. 2017) and (Chu et al. 2018).

After the advent of ResNet CNN architecture (He et al. 2016), Kumra et al. (Kumra et al. 2017) propose its use in grasping in two different modalities: uni-modal with direct RGB or RGD data, and multi-modal with RGB and three-dimensional depth data, therefore two ResNet networks. In both cases, the ResNet layer was responsible for extracting the features that were classified as good grasping by a fully connected layer. This multi-modal strategy is also verified in papers as (Guo et al. 2017). However, Guo et al. (Guo et al. 2017) use tactile data motivated by the fact that the grasping is not only a classification image-like problem. It was expected to model the grasp stability during the training process indirectly. With a deep visual network and a deep tactile network, the hybrid architecture achieved an 89% detection success rate. The practical grasping performance was tested by Chu et al. in (Chu et al. 2018) achieving a success rate of 81% (see Table 3.1). Chu et al. (Chu et al. 2018) also proposed the use of ResNet and VGG-16 CNN architecture with candidate regions to a focused feature search (motivated by Region Proposal Network (RPN)). With this approach, the authors achieved interesting detection rates with practical evaluation, see Table 3.1.

The ResNet is also used by Ghazaei et al. (Ghazaei et al. 2019) in a Fully Convolutional Network (FCN) architecture which, instead of finding a regression from RGB images to rectangle grasp, direct output a Grasp Belief map (see Section 2.1). However, to evaluate and train their network, it was used the CGD which caused (Ghazaei et al. 2019) to "translates" the belief map in rectangle grasping, compromising the truly potential evaluation. Other examples of deep CNN applied to the grasping problem is the LeNet of (Pas, Gualtieri, et al. 2017) that use direct Point-Clouds of the scene to estimate feasible antipodal grasps, and the DarkNet53 (from YoloV3) (L. Chen, Huang, Y. Li, et al. 2020) used to estimated the grasp path discussed in Section 2.1.

It is possible to notice that a better grasp success rate is more dependent on a complete grasping modelling than only the Deep Network architecture strategy. This fact can be seen by the DexNet project's design which confirms that the grasping planning is a more complex task than only a classification image-like problem. Mahler, Pokorny, et al. start the DexNet project (Mahler, Pokorny, et al. 2016) designing an algorithm to create a more reliable database called DexNet 1.0. This database, interactively generated, is composed of a set of parallel-jaw grasps associated with the object's 3D model. Each grasp is labelled with a probability of force closure under uncertainty in object pose, gripper pose, and friction coefficient. In the following works, the database evolved, including: scenario constraints, as planar base surface; different scene points of view; bin-picking and dense clutter scenarios; and ambidextrous policies to select the appropriate gripper (suction or two-finger). With the DexNet, a Grasp Quality Convolution Neural Network (GQ-CNN) was proposed to select and define a robust grasp configuration in single object grasping and bin-picking. Nevertheless, the authors encounter challenges in grasp flexible, porous objects and with loose packing.

As shown in this section, building a sufficient size labelled database is the main drawback of DL policies. Even though a large database could be available, assess if the labelled database modelling includes all necessary practical assumptions is difficult. Therefore, some proposals try to overcome this problem by generating data through practical experiments or using DRL methodologies. That is the case of Pinto et al. (Pinto et al. 2016) that generates a proprietary database with 50k tries of random robotic grasping and mapping them to the rectangle database. For the authors, the image ground-truth and the simulated interactions database are questionable. However, their reduced success rate probably is due to their grasping representation simplification. Although their proposal is a SL, the authors face a similar problem of RL methodologies: the time spent by the database physical try-and-error.

The authors of (Zeng, S. Song, Yu, et al. 2019) propose the direct use of RGB-D pixel-wise to infer, with FCN, affordances grasping instead of classical mapping to grasping parameters. In a grasp-first-then-recognise workflow, the authors mapped affordance maps of a discrete set of grasping primitives for suction and two-finger grippers. With this direct approach, as related by (Zeng, S. Song, Yu, et al. 2019), a faster, reliable, and able to learn complex grasping rules was verified. Zeng, S. Song, Welker, et al. also investigate in (Zeng, S. Song, Welker, et al. 2018) a Deep Reinforcement Learning methodology to evaluate the synergy between push and grasp, achieving interesting results to adversarial clutter scenarios without previous knowledge of the object's shape. It was proposed the use of two FCN, trained by simulated trial-and-error experiments and Q-learning approach. Studies regarding complex object formats are needed.

Using a monocular camera, Levine et al. (Levine et al. 2018) evaluated the use of a CNN and a servomechanism to end-to-end grasping planning, in a visuomotor control fashion. The visuomotor approach allows a direct map from visual sensing, gets continuous environment cues, and reacts to adversarial conditions and perturbations, i.e., more appropriate to dynamic environments (Morrison et al. 2020). However, a drawback in end-to-end is regarding the portability since the algorithm needs to be re-adjusted according to the robot. In these methods, the control loop entirely depends on the system in usage. Other strategies of this methodology can also be check in (James et al. 2017) and (Viereck et al. 2017). Levine et al. (Levine et al. 2018) confronted the high effort to build a reliable self-supervised database, similar to (Pinto et al. 2016) using several robots during two months to teach their grasping prediction CNN. The authors of (Pinto et al. 2016; Levine et al. 2018) noticed that their approach matched deep RL methodologies requisites.

2.3 Conclusion

Several works were reported in the reviewed literature showing the scientific community's significant effort to solve the robotic grasping issue. This chapter elucidated how complex and deep is the grasping planning problem. It also presented some of the several works on the area, the main contributions, and how robotic grasping evolved and still evolving over the past decades: from wrench space heuristics to DL policies, i.e., to analytical to deep machine learning methodologies.

3 Grasping Evaluation Proposal

After the grasping solution development, researchers encounter a common problem: how to evaluate the proposed methodology? The robotic grasping is typically composed of a complex system that includes, in most cases but not essential, the following issues: object sensing and identification; pose estimation; grasping detection; grasping selection; trajectory planning; force and stability estimation; collision avoidance, among others. All these components have their errors related and directly affect grasping performance. For instance, robotic manipulators and 3D sensors have intrinsic's action and measurement errors, respectively. Besides, estimation and planning algorithms also have considerable accuracy errors and variations.

Regarding this evaluation shortcoming, some authors proposed well-accepted assessments in the literature. These include point and rectangle grasping representation comparing with distance threshold and Jacquard index in DL policies' database ground-truth. The Epsilon and Volume wrench space metrics are also other examples applied to analytical grasping methodologies (Section 2.1). Although these metrics are focused on a specific grasping step (rectangle and point comparison metric in the detection, and Epsilon and Volume in physical active-pair interaction), they do not reflect the practical robot grasping performance. Some papers specify their own evaluation metric to a robotic grasping in a real scenario, but it is still difficult for readers to have a comparative review of the results. Therefore, these motivations allow us to formulate some fair and transparent definitions of the results' assessment. Thus, readers can have a clear idea of what is in the comparison between the methods. With the grasping problem steps presented in Figure 3.1, it is defined four grasping evaluation success rate criteria: Grasp Prediction, Grasping Reaching, Grasping-Hold, and Handling Grasping.

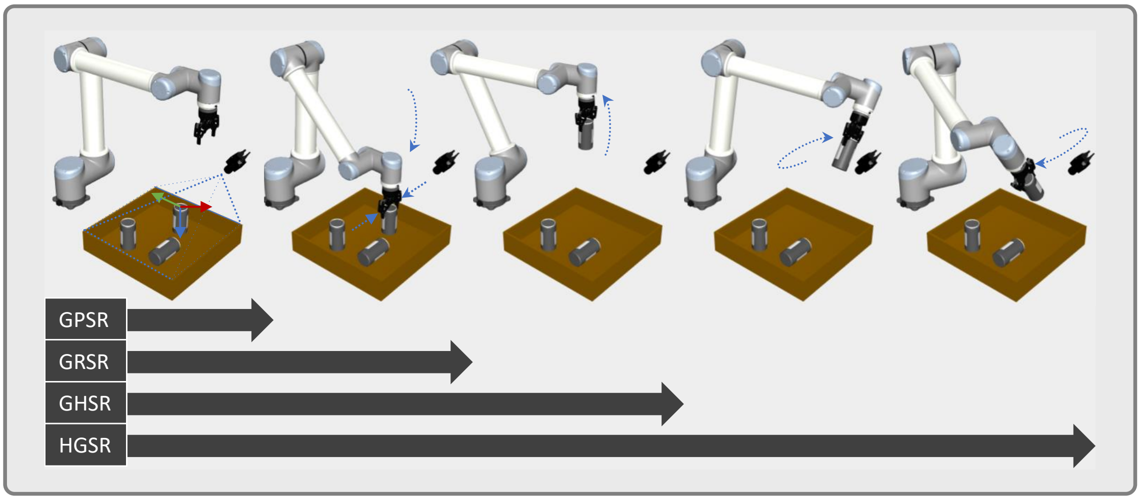


Figure 3.1 – Proposed grasping assessments and their progression timeline. The categories evaluate the grasping performance in different levels based on the complexity of the actions, indicated by the arrows timeline. Each class englobes the prior class resulting in an ascending complexity order from Grasp Prediction (GPSR), Grasping Reaching (GRSR), Grasping-Hold (GHSR) to Handling Grasping success rates (HGSR). The last two successive images describe the 6DOF movements of HGSR.

3.1 Grasp Prediction Success Rate

The Grasping Prediction Success Rate (GPSR) allows the grasp prediction (or detection) evaluation that consists of the grasp pose estimation generated by methodologies such as the DL with CNN (Mahler, Matl, Satish, et al. 2019) and the

Simulated Annealing (SANN) optimization technique (Andrew T et al. 2001; Souza, Carlos M. Costa, et al. 2021).

In these methods, single or multiple grasping poses are compared with ground-truth and an evaluation of how the graspable estimation is likely to perform, with success, given sensing type: RGB image, Depth map, or Point-Cloud. It is necessary to define if the object is previously known or, in case of SL methodologies, if the metric considers sensing-wise (an object sensing data was already presented in the learning phase but with different perspectives from the test case) or object-wise (the object sensing data was never shown in the learning phase). The ground-truth used could be based on a labelled database, such as the CGD database, or human supervision considering the user task expertise. Comparison metrics examples are the Euclidean distance between grasping points, the Jacquard threshold to rectangle grasping, the wrench space volume, or the ϵ-value into a simulation procedure. These methods are discussed in Section 2.1.

This class of success rate is prevalent in deep CNN policy proposals, like (Redmon et al. 2015; Kumra et al. 2017; Asif et al. 2018; Y. Song et al. 2020).Usually, when applied in experimental cases, it has its success rate impaired since it does not evaluate the grasping interaction either the grasping related movements as can be seen in (Chu et al. 2018).

The processing time related is Grasping Prediction Time (GPT). Normally in this step, the convergence time could be higher in analytical/optimization methods (Section 2.2.1) demanding a split in grasping planning: offline (Offline Grasping Prediction Time (Offline-GPT)) and run-time (Run-Time Grasping Prediction Time (Run-GPT)) approaches (Souza, Carlos M. Costa, et al. 2021). In DL policies it could require swap the processing unit from Central Process Unit (CPU) to Graphics Processing Unit (GPU) generating the CPU Grasping Prediction Time (CPU-GPT) and GPU Grasping Prediction Time (GPU-GPT), respectively.

The CPU-GPT and GPU-GPT could be directly compared; however, the unit processing explicitly shows the hardware requirement. In real applications, an ideally need is the CPU-GPT since CPU is cheaper and simpler to be implemented and used, even though this achievement looks like far to state-of-art literature. It is important to note that, with the development of cheaper GPUs and accessible and straightforward libraries, like CUDA (*CUDA Toolkit Documentation v11.0.3* n.d.), this could be an irrelevant issue.

3.2 Grasping Reaching Success Rate

Suppose a grasping task where sensing is performed, followed by the grasp detection, grasp movement, and gripper's close over the grasp point. In this case, the Grasping Reaching Success Rate (GRSR) evaluates the methodology performance considering the deviation from the generate grasping pose and the active grasp point, i.e., the gripper's closing over the object. Therefore, it is a complete way to check the validity of error propagation in a practical scenario compared to GPSR, which only evaluates mathematical modelling parameters where practical inconsistencies can appear. This metric includes the accuracy of object sensing, estimation, grasp prediction, grasping acting and can be specified by these error variations. Works like (Moreira et al. 2016) consider this intermediate error assessment, classified as valid or not by a kNN algorithm, critical to a reliable grasping since the GRSR defines a starter error that can affect the next task steps.

However, not all physical interactions between the active-pair are evaluated, and only the grasping position precision and friction are appraised. It is important to note that small deviations could generate stable grasps, and a vast number of sensors, robotics manipulators, and grippers have a reduced intrinsic's error. Therefore in a controlled environment, the evaluated error could be small. Also, in referred conditions, the estimation, sensing, and planning algorithms must be designed with caution not to generate high error rates, i.e. a complete error propagation from GPSR into GRSR. This metric could be useful in precision grasping applications where the contact grasping region is necessary or the grasp slippage avoiding is critical. In addition, the GRSR could be considered a filter to eliminate a significant pose grasping error and evaluate its propagation in the overall system.

3.3 Grasping-Hold Success Rate

The Grasping-Holding Success Rate (GHSR) allows a complete grasping evaluation criteria. It is considered a robotic grasping application where sensing is performed followed by grasp detection, grasp movement, gripper close over the grasp point, lift the object and hold it for a period. Besides evaluating the error propagation in a practical scenario, as GRSR does, it is possible to check the equilibrium of the grasping performance over perturbations and physical interactions, e.g., the slip, wrench space configuration, and the gravity force actuation. This metric could also be evaluated according to object and gripper material friction changes and holding time variation.

3.4 Handling Grasping Success Rate

Although GHSR could be enough to define a good grasping approach, a more realistic grasping evaluation criteria is the Handling Grasping Success Rate (HGSR). Besides including all considerations of GHSR, the HGSR evaluates the grasping holding of an object and moving it in all robot's DOFs, checking its stability. This metric is relevant for practical scenarios since, in the industry, the work cycle is an important parameter to evaluate. The GHSR could be assessed according to the robot movement's speed and acceleration variation, besides the work cycle. It is also possible to include the object's placing movement. However, any analysis of this procedure could confront a new task apart from grasping.

3.5 Discussion

Aiming to deploy the proposed grasping evaluation in currently state-of-art and, stabilising a fair comparison about the literature solutions, Table 3.1 presents some grasping approaches and their performance. This table is a guiding tool, and the readers are encouraged to check the source since, besides the specific restriction used in each paper's database that can lead to an unfair comparison, the results presented here are an elucidation give the described conditions:

1. All assessments are restricted to agnostic-grasping (the current challenge in grasping) even though some paper also present better results evaluation with previous knowledge of object shape;
2. Only object-wise metric are presented since the object-agnostic is considered;
3. For tests where the author categorise the objects, the most "typical" class is selected in the present table;
4. The column "Model Interaction" indicates if any model interaction is needed to synthesised or selected a grasp (it does not include the database build step of results evaluation) since this affects the grasping performance convergence time;
5. It is only considered static scenario;
6. Each methodology was classified according to proposed Grasp Evaluation, Section 3. Therefore, if realised, it is presented the most complex category in descending order of: HGSR, GHSR, GRSR and GPSR.

Based on Table 3.1 and the grasping evaluation discussed in the presented chapter, it is possible to infer that, even with interesting proposals and results successfully deployed in specific use cases, the robotic grasping still does not have a feasible generalisation solution that comprises the modern industry demands of fast design and easy deployment.

Since a complete and generic solution is unreachable until now, there exists a lack in the deployment of a modular and flexible grasping framework for robots attending to real industry demands and a well structured and formalised architecture which organizes approaches allowing the evaluation and the base to new advancements in science. The evaluation metrics discussed in this chapter could be the first step to achieving this proposal and formalising the research process.

Year	Methodology Description	Clutter Scenario	Gripper Approach	Grasping Representation	Model Interaction	Sensing data type	Processing Time	Result Class	Performance (Success Grasps)
2004	SVM with superquadric shape objects description (Pelossof et al. 2004)	No	Generic*	Eigengrasp	Yes	N/R	N/R	N/R	N/R
2008	Proprietary learning algorithm with logistic regression and image (Saxena et al. 2008)	No	Two-Finger	Point	Yes	RGB	N/R	GHSR	87.80%
2008	Proprietary learning algorithm with logistic regression and image (Saxena et al. 2008)	Light	Two-Finger	Point	Yes	Gray Scale and Depth	N/R	HGSR	80.00%
2011	Two-Stage SVM-rank classification (Yun Jiang et al. 2011)	No	Two-Finger	Rectangle	No	RGB-D	50.000s	GHSR	87.90%
2015	Two-stage deep network (Lenz et al. 2015)	No	Two-Finger	Rectangle	No	RGB-D	13.500s	GHSR	84.00%
2015	Single grasp based on AlexNet direct regression (Redmon et al. 2015)	No	Two-Finger	Rectangle	No	RGD	0.076s	GPSR	84.90%
2015	Multi grasp detection based on AlexNet (Redmon et al. 2015)	No	Two-Finger	Rectangle	No	RGD	0.076s	GPSR	87.10%
2016	K-NN classification of proprietary dataset and histogram of oriented gradient description (Pinto et al. 2016)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	69.40%
2016	Linear SVM of proprietary dataset and histogram of oriented gradient description (Pinto et al. 2016)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	73.30%
2016	CNN with proprietary database (Pinto et al. 2016)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GHSR	66.00%
2016	"Common-Sense" analytical heuristic (Pinto et al. 2016)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	62.11%
2017	Unimodal grasp predictor (DCNN + shallow NN) (Kumra et al. 2017)	No	Two-Finger	Rectangle	No	RGB or RGD	0.062s	GPSR	88.53%
2017	Multimodal grasp predictor (DCNN + shallow NN) (Kumra et al. 2017)	No	Two-Finger	Rectangle	No	RGB and 3Channel Depth	0.100s	GPSR	89.21%
2017	Image and Tactile DCNN (Guo et al. 2017)	No	Two-Finger	Rectangle	No	RGB and Tactile	N/R	GPSR	89.10%
2017	DLSR with NKM-LARS (Trottier et al. 2017)	No	Two-Finger	Rectangle	No	RGB-D	High [a]	GPSR	88.07%
2017	DLSR with GSVQ-ST (Trottier et al. 2017)	No	Two-Finger	Rectangle	No	RGB-D	High [a]	GPSR	88.79%
2017	DLSR with OMP-N (Trottier et al. 2017)	No	Two-Finger	Rectangle	No	RGB-D	High [a]	GPSR	88.56%
2017	DLSR with NKM-N (Trottier et al. 2017)	No	Two-Finger	Rectangle	No	RGB-D	High [a]	GPSR	88.17%
2017	DLSR with RP-N (Trottier et al. 2017)	No	Two-Finger	Rectangle	No	RGB-D	High [a]	GPSR	86.61%
2017	CNN classification on analytical designed dataset (Pas, Gualtieri, et al. 2017)	Dense	Two-Finger	Two-Finger Analytical model	Yes	Point Cloud	1.000s to 8.000s	GHSR	89.00%
2017	Dexnet2.0 and GQ-CNN (Mahler, Liang, et al. 2017)	No	Two-Finger	Antipodal Restrictions	Yes	2.5D	0.8s	HGSR	80.00%
2017	Dexnet2.0 and GQ-CNN (Mahler, Liang, et al. 2017)	No	Two-Finger	Antipodal Restrictions	Yes	2.5D	0.800s	HGSR	93.00%
2017	REG (Mahler, Liang, et al. 2017)	No	Two-Finger	Antipodal Restrictions	Yes	2.5D	2.600s	HGSR	52.00%
2017	IGQ (Mahler, Liang, et al. 2017)	No	Two-Finger	Antipodal Restrictions	Yes	2.5D	1.900s	HGSR	60.00%
2017	IGQ (Mahler, Liang, et al. 2017)	No	Two-Finger	Antipodal Restrictions	Yes	2.5D	1.900s	HGSR	70.00%
2017	CG-CNN with DexNet 2.1 (Mahler and Goldberg 2017)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	9.400s	HGSR	85.00%
2017	CG-CNN with DexNet 2.0 (Mahler and Goldberg 2017)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	10.000s	HGSR	81.00%
2017	SVM with geometric descriptions (Mahler and Goldberg 2017)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	12.857s	HGSR	64,00%
2018	SVM Classifier based on analytical designed dataset (Pas and Platt 2018)	No	Two-Finger	Antipodal Restrictions	Yes	PointCloud	N/R	HGSR	87.80%
2018	SVM Classifier based on analytical designed dataset (Pas and Platt 2018)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	N/R	HGSR	73.00%
2018	VGG-16 (Single Object - Single Grasp) (Chu et al. 2018)	No	Two-Finger	Rectangle	No	RGB-D	0.058s	GPSR	91.70%
2018	RESNET50 (Single Object - Single Grasp) (Chu et al. 2018)	No	Two-Finger	Rectangle	No	RGB	0.120s	GPSR	95.50%
2018	RESNET50 (Single Object - Single Grasp) (Chu et al. 2018)	No	Two-Finger	Rectangle	No	RGBD	0.120s	GHSR	89.00%
2018	EnsembleNet (MobileNet (Reg.) + VGG16 (Reg.) + ResNet50 (Reg.)) (Asif et al. 2018)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	91.20%
2018	EnsembleNet (MobileNet (Joint) + VGG16 (Joint) + ResNet50 (Joint)) (Asif et al. 2018)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	93.70%
2018	EnsembleNet (MobileNet (Joint.) + VGG16 (Reg.) + ResNet50 (Joint.)) (Asif et al. 2018)	Light	Two-Finger	Rectangle	No	RGB-D	N/R	GPSR	92.60%
2018	Planarity Optimization (Mahler, Matl, X. Liu, et al. 2017)	No	Suction	Simple Cup Analytical Model	Yes	PointCloud	N/R	HGSR	67.00%
2018	Centroid Optimization (Mahler, Matl, X. Liu, et al. 2017)	No	Suction	Simple Cup Analytical Model	Yes	PointCloud	N/R	HGSR	78.00%
2018	Planarity and Centroid Optimization (Mahler, Matl, X. Liu, et al. 2017)	No	Suction	Simple Cup Analytical Model	Yes	PointCloud	N/R	HGSR	67.00%
2018	GQ-CNN with Adversarial dataset (Mahler, Matl, X. Liu, et al. 2017)	No	Suction	Simple Cup Analytical Model	Yes	PointCloud	N/R	HGSR	67.00%
2018	GQ-CNN with DexNet 3.0 and Adversarial datasets (Mahler, Matl, X. Liu, et al. 2017)	No	Suction	Simple Cup Analytical Model	Yes	PointCloud	3.000s	HGSR	82.00%
2018	QG-CNN (new architecture) and DexNet2.0 (Jaśkowski et al. 2018)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	N/R	GPSR	86.70%
2018	FCN and Q-Learning (Zeng, S. Song, Welker, et al. 2018)	Dense	Two-Finger	Antipodal Restrictions	Yes	Heightmaps	N/R	GHSR	83.30%
2018	CNN and servoing heuristic (Levine et al. 2018)	Dense	Two-Finger	Analytical	No	RGB	N/R	HGSR	82.50%
2019	Single grasp based on AlexNet direct regression with Grasp Path description (L. Chen, Huang, and Meng 2019)	No	Two-Finger	Rectangle	No	RGD	N/R	GPSR	81.90%
2019	Multi grasp based on AlexNet direct regression Grasp Path description (L. Chen, Huang, and Meng 2019)	No	Two-Finger	Rectangle	No	RGD	N/R	GPSR	84.70%
2019	Single grasp based on ResFCNN and 2D Belief Map description (Ghazaei et al. 2019)	No	Two-Finger	2D Belief Map	No	RGB	N/R	GPSR	81.00%
2019	Multi grasp based on ResFCNN and 2D Belief Map description (Ghazaei et al. 2019)	No	Two-Finger	2D Belief Map	No	RGB	N/R	GPSR	90.60%
2019	6-DOF GraspNet (Mousavian et al. 2019)	Light	Two-Finger	Point Cloud Shape Representation	Yes	PointCloud	3.040s	GHSR	88.00%
2019	Ambidextrous QG-CNN and DexNet4.0 (Mahler, Matl, Satish, et al. 2019)	Dense	Two-Fingers and Suction Cup	Two-Finger Antipodal and Analytical Suction	Yes	PointCloud	11.538s	HGSR	95.00%
2019	Suction geometry heuristic (Mahler, Matl, Satish, et al. 2019)	Dense	Two-Fingers and Suction Cup	Simple Cup Analytical Model	Yes	PointCloud	11.842s	HGSR	80.00%
2019	Composite (Two-Finger and suction) geometric heuristic (Mahler, Matl, Satish, et al. 2019)	Dense	Two-Fingers and Suction Cup	Two-Finger Antipodal and Analytical Suction	Yes	PointCloud	15.126s	HGSR	76.00%
2019	Ambidextrous QG-CNN with DexNet2.0 and DexNet 3.0 (Mahler, Matl, Satish, et al. 2019)	Dense	Two-Fingers and Suction Cup	Two-Finger Antipodal and Analytical Suction	Yes	PointCloud	14.117s	HGSR	76.00%
2019	FCN and ConvNet (Zeng, S. Song, Yu, et al. 2019)	Dense	Two-Fingers and Suction Cup	Discrete Set of Primitives Grasps	Yes	RGB-D	N/R	HGSR	92.40% (Suction) 96.70% (Gripper)
2019	Suction heuristic based on surface normal variance over a cloud (Zeng, S. Song, Yu, et al. 2019)	Dense	Suction Cup	Simple Cup Analytical model	Yes	PointCloud	N/R	HGSR	35.20%
2019	Antipodal heuristic in hill format shapes cloud (Zeng, S. Song, Yu, et al. 2019)	Dense	Two-Finger	Antipodal Restrictions	Yes	PointCloud	N/R	HGSR	92.50%
2020	Backboned in Blacknet53 (YOLOv3) with Grasp Path description (L. Chen, Huang, Y. Li, et al. 2020)	No	Two-Finger	Rectangle	No	RGB	0.110s	GPSR	94.60%
2020	One stage region convolutional network (Y. Song et al. 2020)	No	Two-Finger	Rectangle	No	RGD	N/R	GPSR	95.60%
2020	GG-CNN (Morrison et al. 2020)	No	Two-Finger	Rectangle Like [b]	No	2.5D	0.019s [c]	GHSR	92.00%
2020	GG-CNN with visuomotor feedback (Morrison et al. 2020)	No	Two-Finger	Rectangle Like [b]	No	2.5D	0.019s [c]	GHSR	91.00%
2020	GG-CNN with visuomotor feedback (Morrison et al. 2020)	Dense	Two-Finger	Rectangle Like [b]	No	2.5D	0.019s [c]	GHSR	87.00%

N/R: not reported;
All acronyms are presented in related papers;
[a] As stated by the authors. The algorithm do not achieved competitive performance time;
[b] Minor differences from rectangle representation;
[c] It was presented prediction time and not performance time.

Table 3.1 – Object-agnostic grasping methodology proposals and their performance

4 Modular Grasping Pipeline Architecture

Nowadays, since a completely generic solution, independently from previous knowledge of object shape, gripper design, and sensing type, is an unfeasible solution, there is a gap between real application, research development and scientific evaluation. Thus, creating a re-configurable grasping framework integrating the best of all methodologies could transfer this to a practical robotic grasping task which is main contribution of the presented thesis.

The core of the developed grasping planning architecture pipeline is divided into two parts: the grasping synthesis and the grasping selection (Figure 4.1). In summary, the grasping synthesis is a system architecture responsible for generating all the grasping poses. It creates a set of hypothetical grasping candidates in an offline step, i.e., it runs outside the robot system in a setup phase. The generated data is then uploaded to the robot system to be used during the grasping selection step. This architecture is responsible for choosing the best grasping candidate following a set of heuristics and priorities. It is a task-oriented procedure that analyses the environment and the run-time constraints of the task. The following sections provide a detailed description of the procedure.

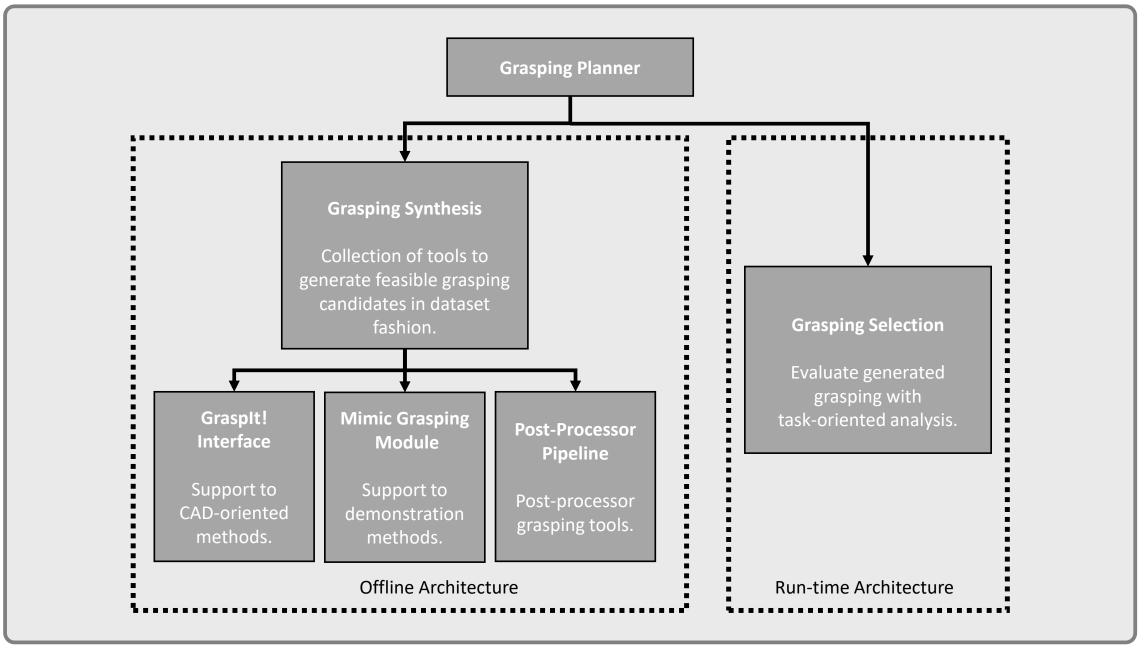


Figure 4.1 – Proposed modular grasping architecture.

4.1 Grasping Synthesis

The grasping synthesis is a set of tools responsible for generating and handling the grasping poses dataset. The grasping synthesis is an offline step, i.e., it runs outside the robot system in a setup phase. The dataset is created as a set of hypothetical grasping pose candidates which are hierarchically structured in a YAML file format. The proposed dataset standard is detailed discussed in Section 4.1.1.

Currently three main components constitutes the grasping synthesis: the "GraspIt!" Interface (Section 4.1.2), the Mimic Grasping Module (Section 4.1.3) and the Post-Processor Pipeline (Section 4.1.4).

4.1.1 Grasping Dataset Standard

A grasping candidate, also referred as candidate, is defined by its pose over an object geometry. Besides this fundamental characteristic, others should be defined according to gripper in usage. Therefore, a structured grasping candidate dataset is proposed.

The candidates are specified by an YAML descriptor (Snippet 4.1) and they are sequentially located in a YAML configuration file. This dataset file can be loaded in the ROS parameter server afterwards.

```yaml
candidate_0:
  method:
    type: 1
  gripper:
    type: 1
    parameters: [125, 0.085, 0.14, 0.11, 0.14, 3, 70.65, 0.028, 0.0016]
    DOFs: [0.16049]
  parent_frame_id: ""
  position:
    x: -0.00563618
    y: -0.0201342
    z: 0.0118393
  orientation:
    x: -0.683578
    y: 0.685041
    z: -0.180015
    w: 0.176166
```

Snippet 4.1: The candidate dataset descriptor example.

The candidates are unique for gripper-object, therefore there exists one datatset per active pair, and they are named as "candidate_id", where id is a integer that defines the order into datatset. The others descriptor parameters are defined below:

- **method/type:** an integer that defines which synthesis method build the candidate.
- **gripper/type:** an integer that defines the gripper type.
- **gripper/parameters:** a dynamic size array with specific gripper's parameters. The "gripper/type" parameter define how to read this array,

e.g for gripper type 0 the sequence is: force [N], velocity[m/s], pre-grasp-width [m], grasp-width [m], grasp-release-width [m], grasping model;

- **gripper/DOFs:** a dynamic size array with fingers' joints Degree of Freedoms (DOFs) value. Since the grasping candidate could be defined by the eingengrasp (Appendix A.0.3) or the individual fingers joints state, the "method/type" define how to extract this information from the array. Another important parameter to define this array is the "gripper/type";
- **parent_frame_id:** reference frame in which the candidate is defined. If it is not declared, it is considered the object reference;
- **position:** position w.r.t. parent frame id, in meters.
- **orientation:** orientation w.r.t. parent frame id, in quaternion.

4.1.2 "GraspIt" Interface

The "GraspIt!" interface is responsible to generate grasping candidates based on CAD modeling by using the "GraspIt!" API and simulator, in association with Robot Operation System (ROS) framework. This simulator was first proposed by (Andrew T Miller et al. 2004) and is widely used in academic community to multi-fingered grasping analysis (Appendix A.0.2) using CAD interaction in virtual environment. The grippers are structured in XML format (besides the 3D model, the Interest Contact Region (ICR) and the eigengrasps, Appendix 1, are also defined). The objects are included by using a Polygon File Format (extension ".ply").

The Figure 4.2 presents the "GraspIt" interface pipeline workflow.

The implemented methods are discussed into Sections 4.1.2.1 and 4.1.2.2. The interface is based on C++ polymorphic classes which allows new heuristics design and also incorporate new future "GraspIt!" functionalities.

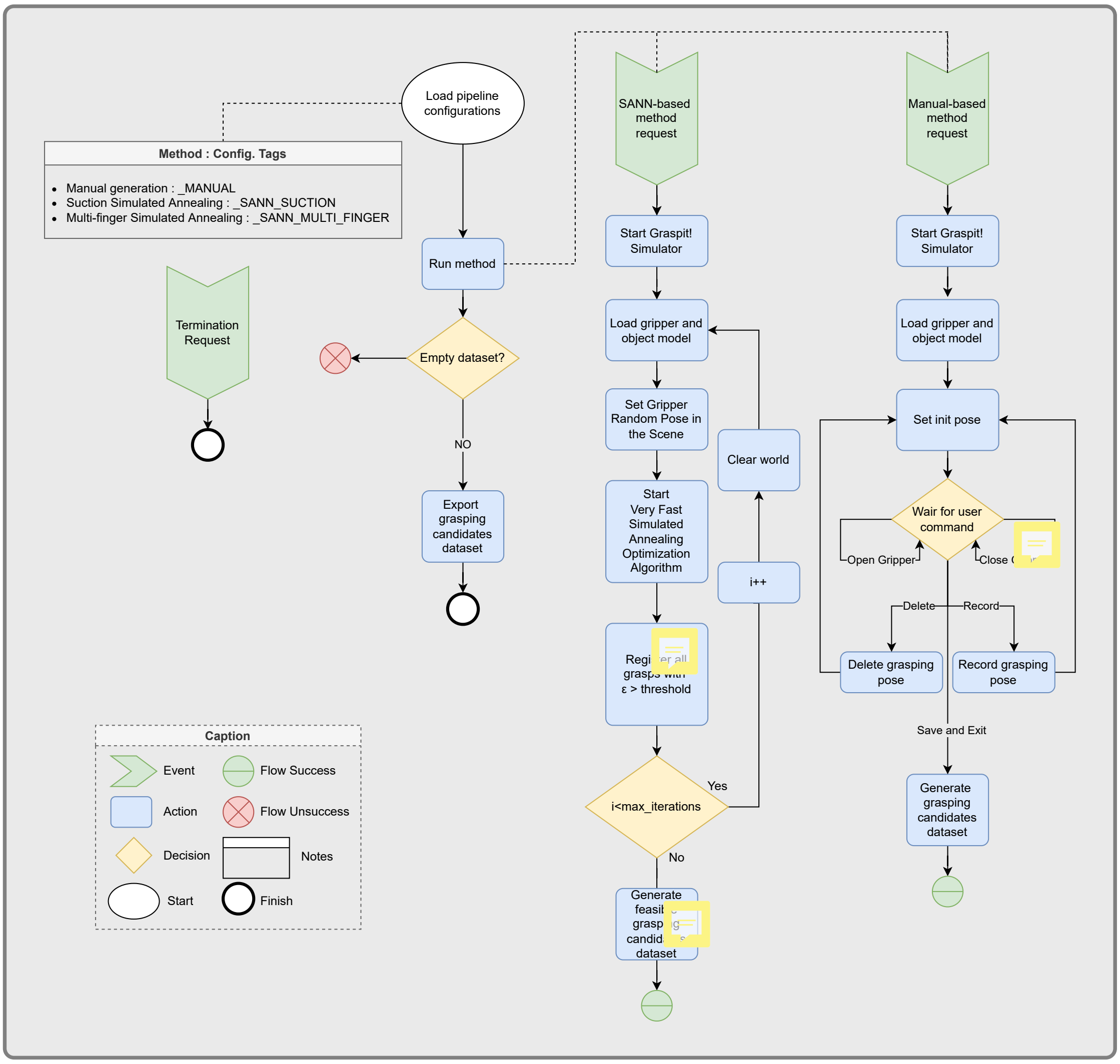


Figure 4.2 – "GraspIt" interface pipeline workflow

4.1.2.1 Manual Feeder

The manual feeder allows the user to define grasping postures using the "GraspIt!" 3D environment. This feeder automatic call and configure the "GraspIt!" virtual interface. It also call a menu, which allows the operator to store, delete, manipulate the gripper and generate the grasping dataset (Section 4.1.1).

The YAML descriptor is defined in Snippet 4.2 and its parameters is presented below:

```
X_MANUAL:
    path:
        gripper_models_file_path: ''
        object_models_file_path: ''
    gripper_file_name: 'robotiq_85_gripper/robotiq_85_gripper.xml'
    object_file_name: 'object.ply'
    config_grasps_parameters:
        gripper_id: 0
        approach_width_multiplier: 2
        release_width_multiplier: 2
        min_width_threshold: 0.015
```

Snippet 4.2: The "GraspIt!" manual descriptor example.

- **path/gripper_models_file_path:** path to locate the gripper XML model;
- **path/object_models_file_path:** path to locate the object ".ply" model;
- **gripper_file_name:** gripper XML file name;
- **object_file_name:** object polygon file format name;
- **config_grasps_parameters/gripper_id:** gripper ID code;
- **config_grasps_parameters/approach_width_multiplier:** the multiplier to define the grasping width in approach procedure. This value is applied over the grasping candidate width;
- **config_grasps_parameters/release_width_multiplier:** the multiplier to define the grasping width in release procedure. This value is applied over the grasping candidate width;
- **config_grasps_parameters/min_width_threshold:** minimum width to consider in grasping procedure. Some grippers have flexible finger that need to be considered.

4.1.2.2 Simulated Annealing based Feeder

The "Graspit!" has support to grasping automatic generation by using the Very Fast Simulated Annealing optimization algorithm. For a detailed explanation see Appendix A.0.3. This functionality is also incorporate to the core pipeline by the YAML descriptor presented in Snippet **??**. Besides the multi-fingered approach (Appendix A.0.2) , a descriptor to suction grippers is designed considering that a good suction grasping is direct related to well define contact point Snippet 4.4.

- **path/gripper_models_file_path:** path to locate the gripper XML model;
- **path/object_models_file_path:** path to locate the object ".ply" model;
- **gripper_file_name:** gripper XML file name;
- **object_file_name:** object polygon file format name;
- **graspable_body_id:** defines which object in scene is the grasping focus;
- **graspable_body_id:** defines which object in scene is the grasping focus;
- **action_graspit_interface_server_name:** since the "GraspIt!" server is deployed as a ROS action server (ROS n.d.[a]), its names should be specified;
- **action_server_timeout:** timeout to detect that the "GraspIt!" server is not running;
- **iterations:** how many iterations the SANN will be executed (Figure **??**);
- **energy_threshold:** convergence optimisation threshold;
- **sim_annealing/max_steps:** maximum steps of one SANN iteration;
- **sim_annealing/feedback_num_steps:** allow visual update of optimisation process;
- **sim_annealing/set_custom_params:** set custom params;
- **sim_annealing/YC:** annealing constant for neighbor generation schedule;

```yaml
X_SANN_SUCTION:
    path:
        gripper_models_file_path: ''
        object_models_file_path: ''
    gripper_file_name: 'generic_simple_suction_cup/
       generic_simple_suction_cup.xml'
    object_file_name: 'object.ply'
    graspable_body_id: 0
    action_graspit_interface_server_name: '/graspit/planGrasps'
    action_server_timeout: 360
    iterations: 10
    energy_threshold: .4
    sim_annealing:
        max_steps: 70000
        feedback_num_steps: 0
        set_custom_params: false
        YC: 7.0
        HC: 7.0
        YDIMS: 8.0
        HDIMS: 8.0
        NBR_ADJ: 1.0
        ERR_ADJ: 1e-6
        DEF_K0: 30000
        DEF_T0: 1e6
    config_grasps_parameters:
        gripper_id: 3
```

Snippet 4.4: "GraspIt!" suction SANN YAML descriptor example.

- **sim_annealing/HC:** annealing constant for error acceptance schedule;
- **sim_annealing/YDIMS:** number of dimensions for neighbor generation schedule;
- **sim_annealing/HDIMS:** number of dimensions for error acceptance schedule;
- **sim_annealing/NBR_ADJ:** adjust factor for neighbor generation schedule

- **sim_annealing/ERR_ADJ:** adjust raw errors reported by states to be in the relevant range of the annealing schedule;
- **sim_annealing/DEF_K0:** starting step;
- **sim_annealing/DEF_T0:** starting temperature;
- **config_grasps_parameters/gripper_id:** gripper ID code;
- **config_grasps_parameters/approach_width_multiplier:** the multiplier to define the grasping width in approach procedure. This value is applied over the grasping candidate width;
- **config_grasps_parameters/release_width_multiplier:** the multiplier to define the grasping width in release procedure. This value is applied over the grasping candidate width;
- **config_grasps_parameters/min_width_threshold:** minimum width to consider in grasping procedure. Some grippers have flexible finger that need to be considered.

4.1.3 Mimic Grasping Module

The "GraspIt!" interface (Section 4.1.2) generates grasping candidates based on CAD modeling interaction. This category could demand unnecessary effort if the gripper CAD modelling is not in disposition or if the 3D design does not compensate for a simple grasping application with a reduced candidate number. Therefore, human demonstration approaches could facilitate this deployment that allows less knowledgeable users to create a grasping representation.

A grasping demonstration task could be defined as two localisation problem: the human manipulation and the graspable object pose estimation. Several techniques are proposed in current literature to deal with both issues TODO, in association or individually. Therefore a structured demonstration grasping module architecture could improve the capacity to deploy, test and evaluate different techniques (e.g. computer vision heuristics and machine learning-based) and hardware setups (e.g. sensor technologies and grippers structures), correctly adapting to different applications necessities.

In this regard, the present thesis proposes a mimic grasping architecture in the format of C++ API based on plugin management. The related plugins system is also developed as an C++ API. The plugin strategy is important since different techniques could be implemented as a dynamic library that is loaded in run-time into the mimic grasping architecture without the need to recompile the core pipeline. The plugin manager API provides metadata interface allowing the designing of plugins and the support to load this type of dynamic library into custom core applications, Figure 4.3 elucidates the proposal.

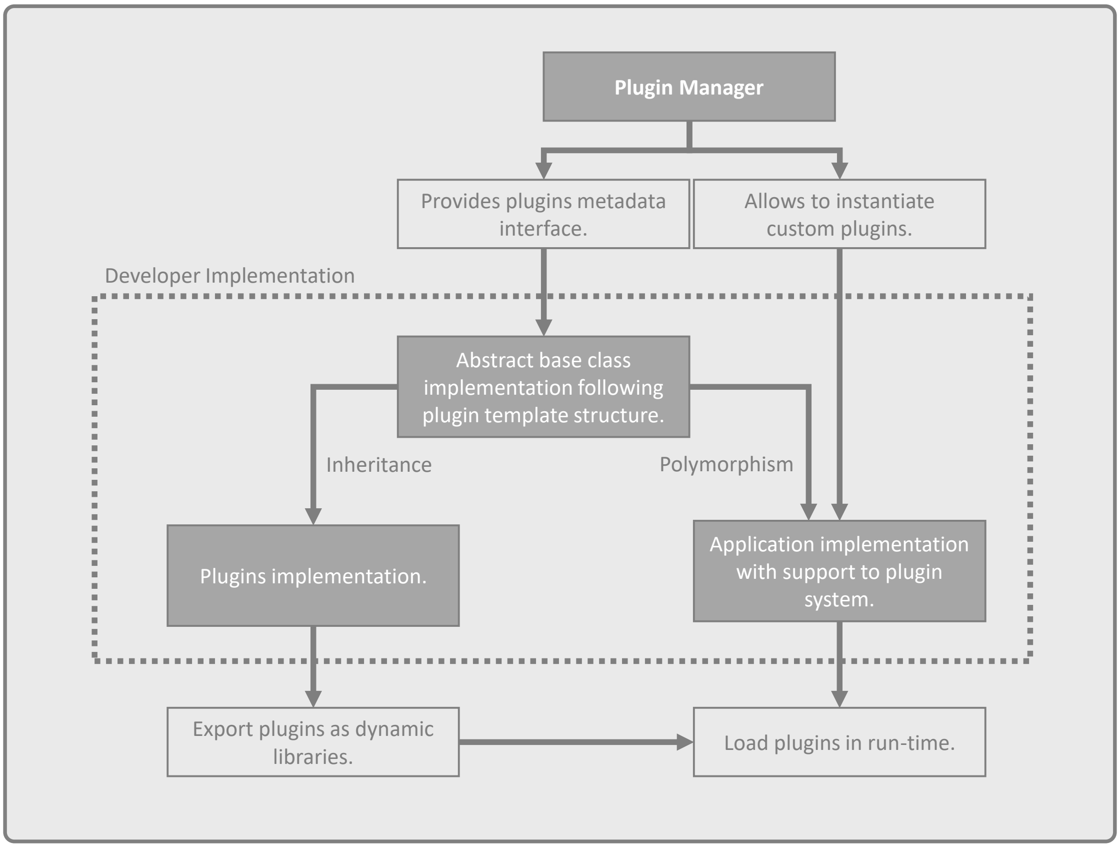


Figure 4.3 – Proposed plugin system management.

The mimic grasping pipeline structured workflow is presented in Figure 4.4. The mimic workflow concept considers that tools are necessary to define the human grasping, such as gloves or handler mechanisms, or at least a remote control technique that allows demonstrators requests while performing the demonstration. In addition, it is considered that the two localisation methodologies are processed in sequence after the operator record request.

Namely, this workflow starts by loading API general configurations, such as tool communication parameters and localisation methods definitions. The localisation method definition is composed by:

- **plugin_name:** the plugin dynamic library implementation file name;
- **plugin_config:_file:** the plugin dynamic library configuration file name. This is necessary since the parameters are custom defined according to plugin implementation;
- **executor_cmd and terminator_cmd:** typically, but not mandatory, plugins are client implementations since the localisation server could be loaded a part. Therefore the pipeline need to bring up these outer systems by running a user defined script (or a simple command), caller executor. At end, a shutdown scripts (or command), called terminator, is expected to turn off these systems.

When the system is started the implemented plugins are loaded into API memory with its respective parameters, the terminator is executed and the communication with demonstration tool verified. If all process is correctly executed, the demonstration process is started, i.e. the human operator positions himself in grasping configuration, execute the grasping and request the record. This procedure can be performed several times and, at end, the operator can export the grasping datatset (which respect the proposed standard 4.1.1).

This Ph.D thesis also develops the handler hardware (Section 4.1.3.1) and its firmware (Section 4.1.3.2) for deployment in a use-case. The proposed use-case is important to verify the mimic grasping pipeline functionality. This use-case consists of two plugins based on: 6D mimic pascal application and DRL C++ ROS package. The first identifies a robot gripper replica operated by a human using stereoscopic vision (Section 4.1.3.3) while the second estimates the graspable object pose using a structured light camera, Photoneo Phoxi 3D Camera (Schmalz 2022b)

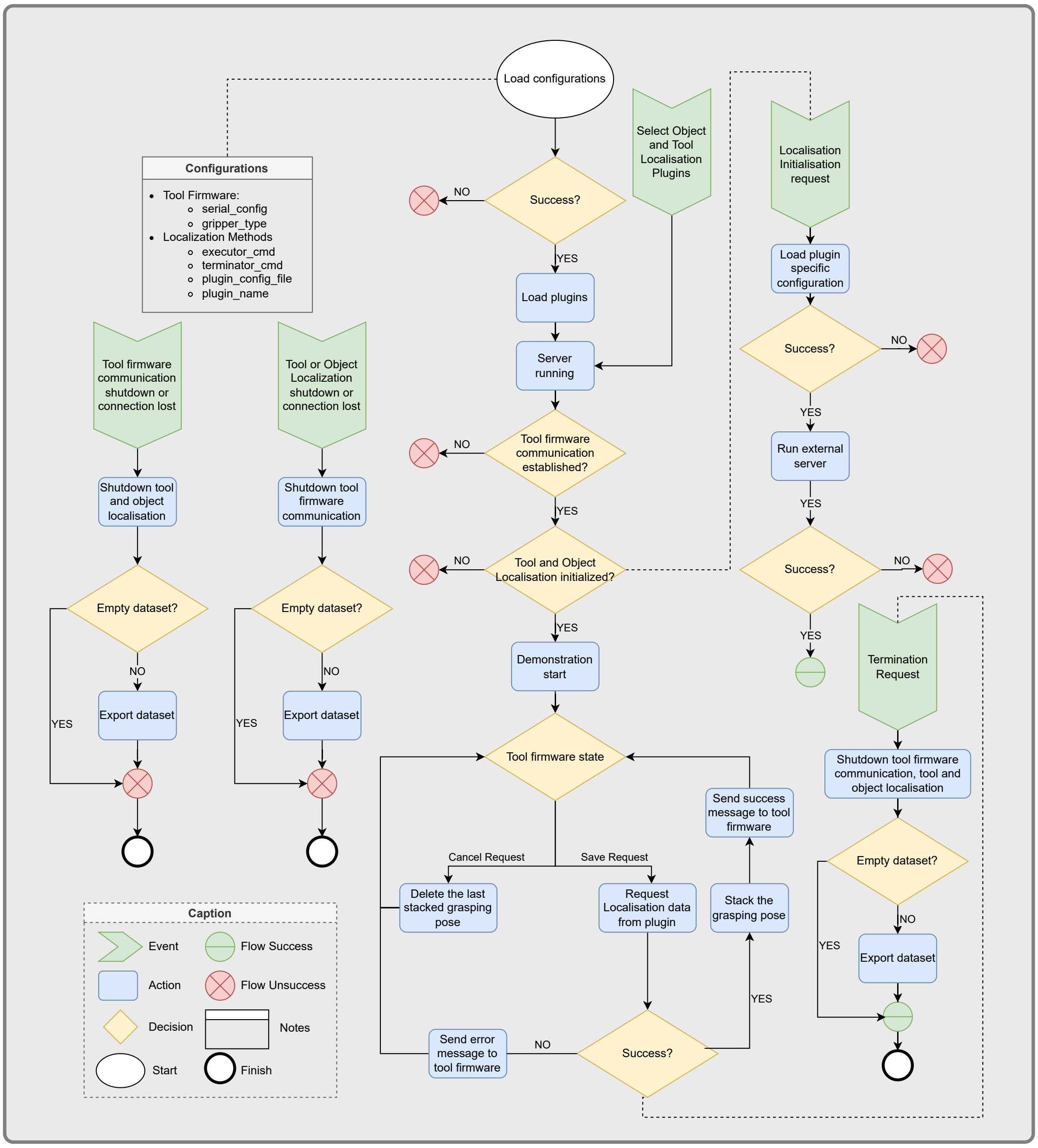


Figure 4.4 – Proposed modular mimic grasping architecture.

(Section 4.1.3.4). A GUI is implemented which loads the mimic grasping API and assesses the use-case (Section 4.1.3.5).

4.1.3.1 Tool Hardware

A handle tool is designed to a human perform a grasping demonstration, Figure 4.5. The handler is created following the modular concept since, it need to support different robots' grippers and detection hardware, if needed. Therefore, the human can teach how to grasp with the same gripper used by the robot.
In addition, the handler supports an Arduino Nano which is embedded into an electronic case (Figure 4.5), two buttons to open/close the gripper and record/delete grasping, and an RGB LED to visual firmware states feedback. Eletronic schematics is presented in Appendix B.0.1. The gripper support is a modular part, allowing to quickly change tools Figure 4.5. The cover is also modular since tool localisation techniques could have different identification devices. The developed tool firmware is discussed in Section 4.1.3.2

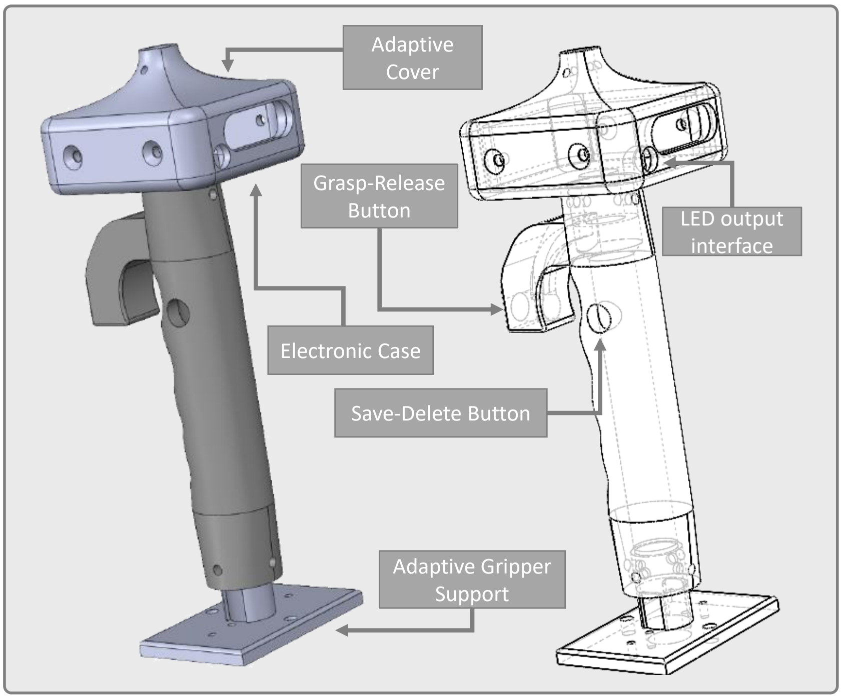


Figure 4.5 – Proposed handler tool.

For the present thesis, the grippers supported by mimic grasping are: the parallel two-finger gripper HGPC16A (*Parallel gripper HGPC-16-A* 2017) and the foam suction cup FM-SW 76x22 4x6 N10 (Schmalz 2022a) (Figure 4.6). It is important to note that the firmware can be easily improved with more types of grippers. Since the supported gripper are pneumatics, a valve system is also designed.

Figure 4.6 – Mimic grasping supported grippers.

4.1.3.2 Tool Firmware

The firmware is designed to interact with a companion computer with the mimic grasping server working on it. The communication is done by serial with a default baud rate of 115200. The Figure 4.7 presents the firmware finite state machine and each state is described bellow:

- **Waiting Server:** Idle initial state. It waits for a start command from the server. In the start command message, the gripper type needs to be defined. Output LED state: blink yellow;
- **Running:** Tool running rest state where the user can manipulate it. Only the grasping button is enabled allowing the "Grasping" transition. Output LED state: solid white;
- **Active Gripper:** State that active the grasping. Both buttons are enabled. If the grasping button is pressed the "Release" transition is activated. Meanwhile, the save button actives the "Save Request" transition. This state is related to the implemented actuator. Actually, this state only supports pneumatic grippers which are performed by relays usages. Output LED state: solid cyan;

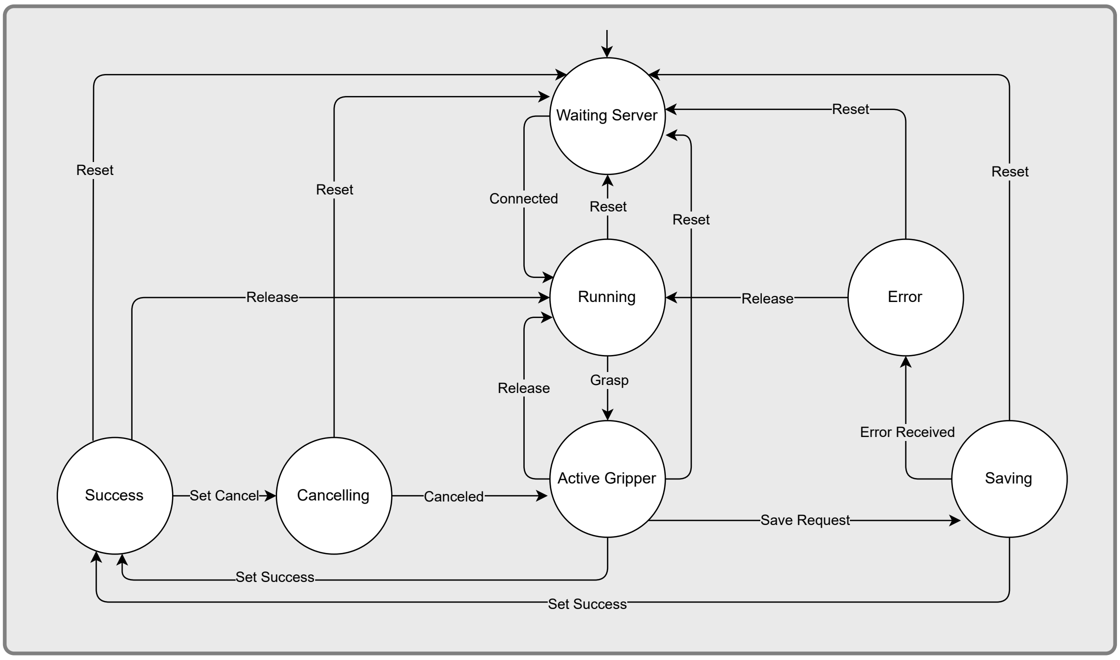


Figure 4.7 – Proposed firmware finite state machine.

- **Saving:** State which firmware requests the server to acquire data and save it. None of the buttons is enabled. If the server performs the operation successfully the "Set Success" transition will happen. Otherwise, the "Error Received" transition is activated. Output LED state: Blink Blue;

- **Error:** Only the grasping button is enabled allowing the user to release the work object and leaving the error mode ("Release" transition) to restart the operation. Output LED state: blink red;

- **Success:** State that indicates that the grasping pose is correct recorded by the server. Both buttons are enabled. Grasping button pressed actives the "Release" transition. Holding save button actives the "Set Cancel'" transition. Output LED state: solid green;

- **Cancelling:** State which the firmware requests the server to delete the last acquired grasping. The "Canceled" transition will only be activated by feedback from the server.

All states can be reset by a reset message from the server. Thus, the initial state (waiting server) will take place. The implemented state machine is a Moore Machine with its outputs described by Table 4.1.

State	Output		
	Active Gripper	Server Save Request	Server Remove Request
Waiting server	0	0	0
Running	0	0	0
Active Gripper	1	0	0
Success	1	0	0
Cancelling	1	0	1
Saving	1	1	0
Error	1	0	0

Table 4.1 – Firmware state's output

4.1.3.3 6D Mimic Interface

The 6D Mimic was first introduced by (Ferreira et al. 2016) and improved in current thesis (de Souza, Amorim, et al. 2021), with the objective to facilitate the robot programming in industrial painting procedure. The core relies in tracking a luminous marker movement attached on a painting tool. Stereo cameras are used to perform the marker identification. Therefore, the user can teaching by demonstration the painting procedure to a robot without the need to programming it. In summary, the systems is constituted of a main computer and electronic interface to control the luminous marker.

In the present thesis the 6D Mimic system is used in current Mimic Grasping use-case. The handler tool has attached the luminous marker, in the cover support 4.5, which is properly calibrated with the new tool (the relationship between the marker and the TCP).

A Pascal server is designed into 6D Mimic computer to communicate with Mimic Grasping companion computer 4.8. This server is based on 6D Mimic main pipeline, however, it is adapted to generate a grasping candidate. More specifically, when the user request to save a grasping pose (Section 4.1.3.2) an interface plugin, into Mimic Grasping Module, requests through a Transmission Control Protocol (TCP) socket the tool tracking. The tool pose is defined by the mean value of a pre-defined acquisition time, which default value is three seconds. Afterwards, the 6D Mimic computer sends the structured grasping pose through TCP back to the plugin interface which passes it to the Mimic Grasping core structure. The implemented 6D mimic server also sends a hearthbit signal ensuring that the computer communications are properly working, otherwise, an error is detected by the plugin interrupting the teaching process.

4.1.3.4 Dynamic Robot Localisation Interface

The DRL, proposed by (Carlos M. Costa et al. 2016), was initially developed as a modular robot localisation system but its application is high, being able to be used in object recognition. The configuration to perform this task builds the basis of the Object Recognition Package (OR) (Carlos M Costa 2022). This package is structured in ROS action service allowing the user to track objects' poses using a CAD reference model and a acquired point cloud image. Detailed description of the technique can be checked at (Carlos M. Costa et al. 2016) and (Carlos M Costa 2022).

The purpose of the present thesis is to deploy this package as a mimic grasping server and plugin to recognize the object pose. Herewith, the grasping candidate generated by the 6D Mimic interface 4.1.3.3 is completely defined. To perform so, Photoneo Phoxi 3D Camera S (Schmalz 2022b) is used to point cloud acquisition.

A ROS wrapper package to OR is created aiming to deploy the server to process the 3D image which comes from a Photoneo Phoxi 3D Camera S (Schmalz 2022b). Both, the server and the camera package are implemented as ROS action server, therefore to require the 3D image a goal request is emitted followed by a goal request to process the tracking. These goals are executed by the proposed Phoxi+DRL plugin which also load all necessary configurations to package and load the ROS environment. It is important to note that eh Mimic Grasping does not have any dependencies with ROS, being the ROS prerequisites in charge of the plugin. The related procedure elucidation is exposed in Figure 4.8.

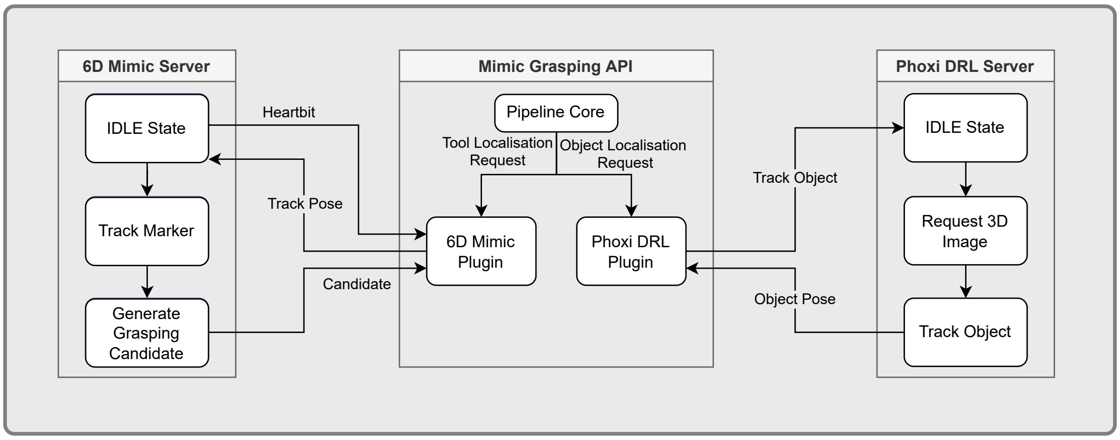


Figure 4.8 – Run-time use-case behaviour. The plugins are responsible to stabilises the communication between the localisation strategies and the Mimic Grasping pipeline. 6D Mimic and Phoxi DRL are the implemented interfaces to the proposed use-case.

4.1.3.5 Mimic Grasping Graphical User interface

A GUI is designed to allow easy interaction with Mimic Grasping API. The main window is presented in Figure 4.9 where the object 3D model is plotted and reference frames indicate the recorded grasping candidate while performing the demonstration. On the right, an output log panel is responsible to inform the mimic grasping core procedure followed by a handler firmware state indicator. Buttons allow the user to start and stop the pipeline and also export the dataset which is also plotted in 3D visualisation.

The menu interface gives access to configurations such as localisation and firmware setup. The localisation configuration is defined in Figure 4.10. In this window, the user can define which plugin will be loaded followed by the plugin-specific configuration file, the executor, and terminator commands (Section 4.1.3). These configuration can be saved to be used later by the Mimic Grasping API.

Regarding the proposed handler firmware, it is possible to communicate, configure it and test it using the firmware window 4.11. It is an important configuration menu since needs to be configured according to the gripper attached to the handler.

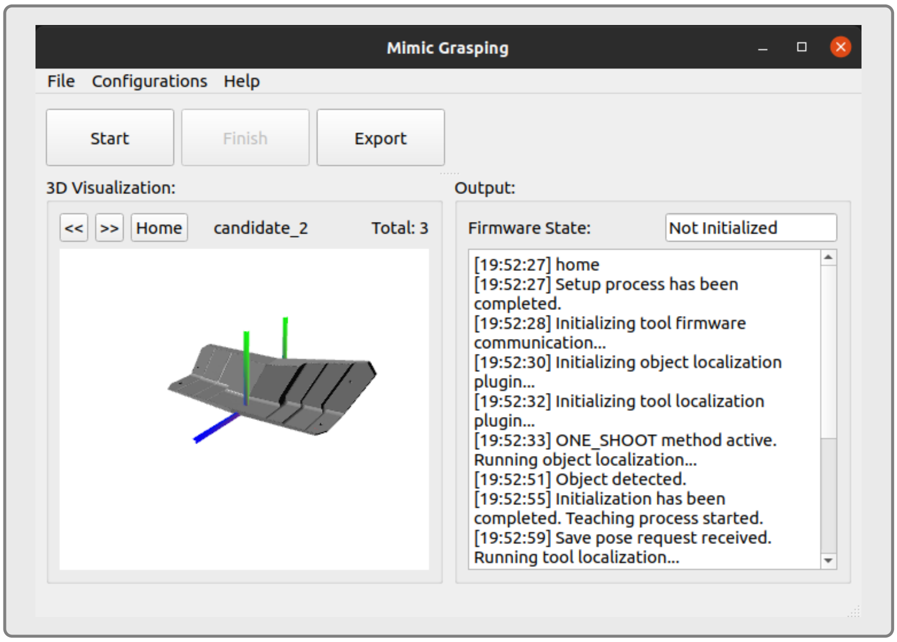


Figure 4.9 – GUI main window.

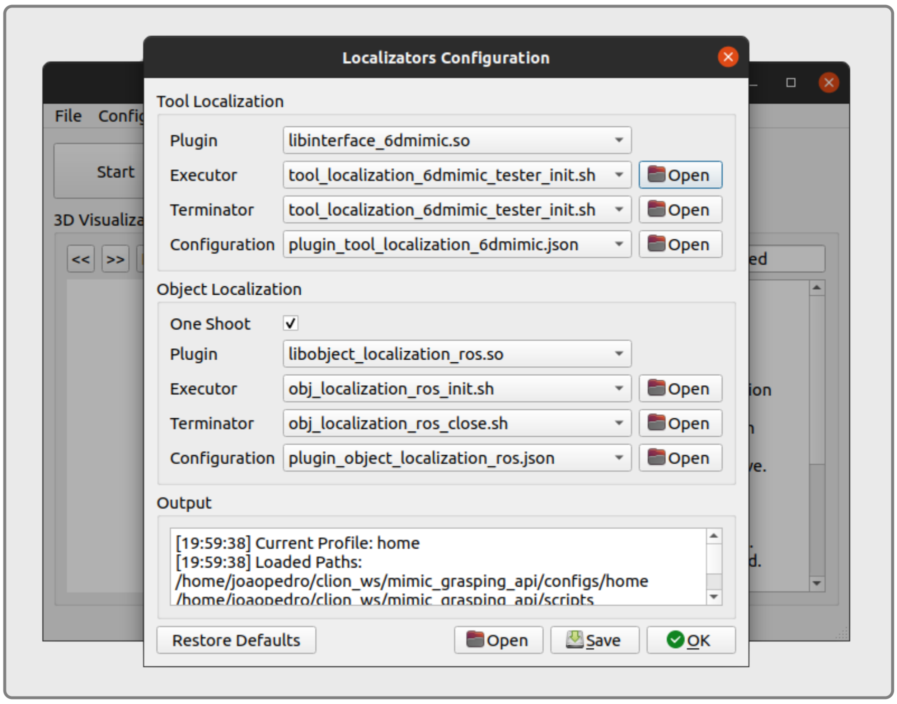


Figure 4.10 – GUI localisation configuration window.

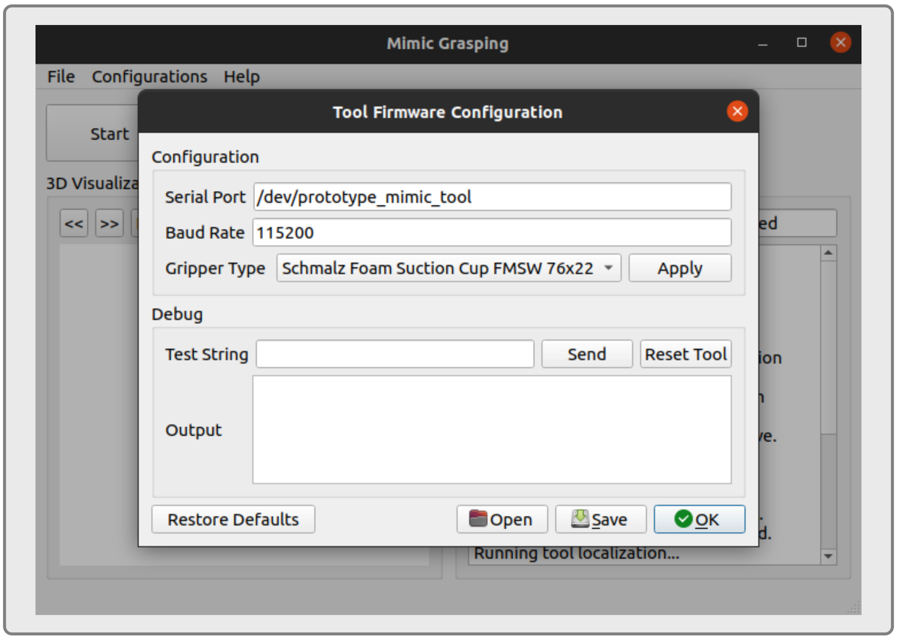


Figure 4.11 – GUI firmware configuration window.

4.1.4 Post-Processor Pipeline

The post-processor pipeline is a software structure based on ROS framework responsible to edit and manipulate the datatasets in a pipeline workflow structure defined by a YAML configuration file. The Figure 4.12 presents the Post-processor pipeline workflow.

A YAML pipeline configuration file is responsible to define the cascade of methods to be deployed, Snippet 4.5. It is mandatory that the first method be a unique loader method. The loader is responsible to import the dataset to be used by the following "post-processor‘" methods.

Since the post-processor pipeline architecture code design is based on polymorphism, which is implemented in C++, the server allows being incremented with new heuristics classes that inherit a defined parent class. Therefore new approaches can be easily integrated without modifying the core architecture and pipeline workflow.

The proposed post-processors methods are described in the Sections 4.1.4.1 to 4.1.4.5.

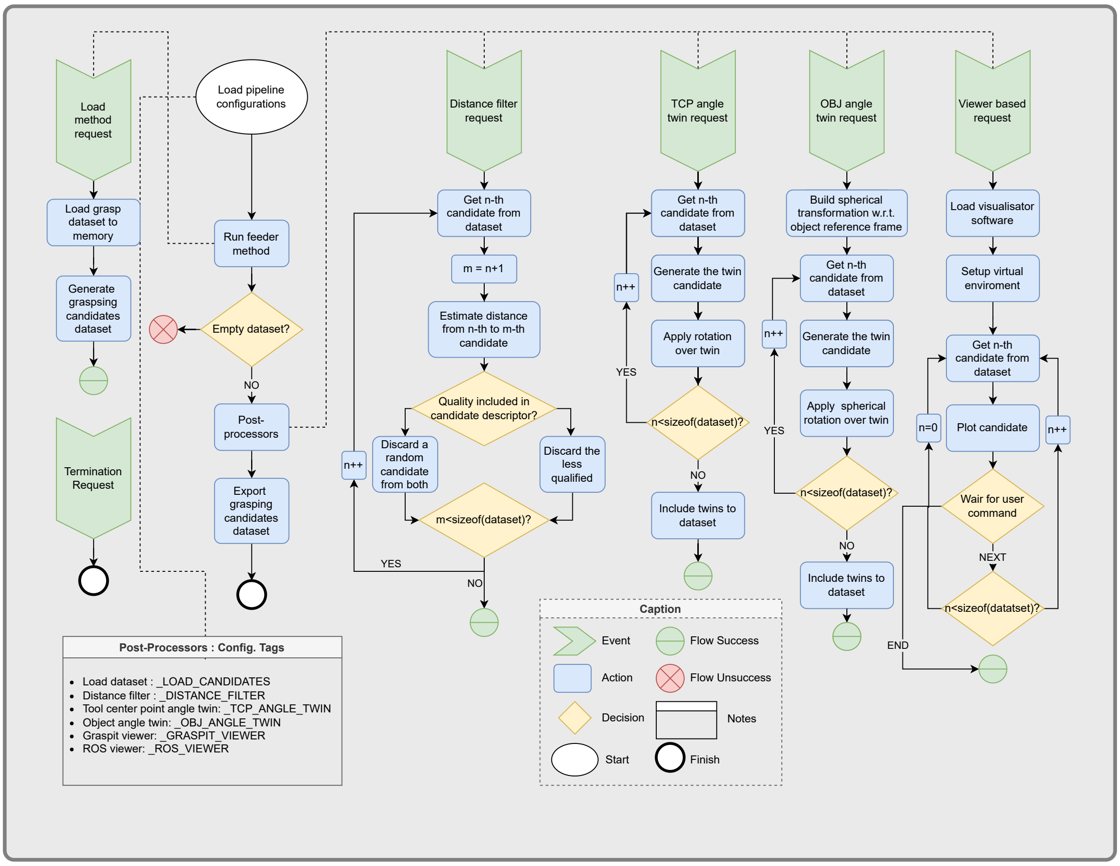


Figure 4.12 – Post-processor pipeline workflow.

4.1.4.1 Loader

The loaded allows to import a already created datatset into post-processor pipeline. It important to note that the dataset should follow the proposed convention described in Section 4.1.1. It also allows to merge several different datasets into one. The loader YAML descriptor is defined in Snippet 4.6 and its parameters is presented below:

- **model_file_name:** the object name since each dataset is defined over an specific object;
- **number_of_files:** number of datatsets to be loaded.;
- **ns_filename_of_candidates:** namespace to differ candidates from different datatsets.

```yaml
output_file_path: ''
output_log_path: ''
apm_mode: false
ros_verbosity_level: 'DEBUG'
pipeline:
    0_LOAD_CANDIDATES:
        model_file_name: support_bracket
        ns_filename_of_candidate: 'file_'
        number_of_files: 1
    1_DISTANCE_FILTER:
        abs_euclidean_distance_threshold: 0.01
        abs_roll_distance_threshold: 20
        abs_pitch_distance_threshold: 20
        abs_yaw_distance_threshold: 20
    2_TCP_ANGLE_TWIN:
        Rz: 180
        Ry: 0
        Rx: 0
```

Snippet 4.5: Post-processor pipeline configuration example.

```yaml
X_LOAD_CANDIDATES:
    model_file_name: object_name
    ns_filename_of_candidate: 'file_'
    number_of_files: 1
```

Snippet 4.6: Load dataset YAML descriptor example.

4.1.4.2 Distance Filter Post-Processor

In the grasping dataset some candidates could be in duplicate. Therefore a relevance filter based on distance is proposed to eliminate redundancies. Basically it estimates the euclidean and the angular (roll, pitch and yaw) distances between all candidates into datatset. If these distances are smaller than specific configuration thresholds a duplicate is detected and the candidate replica deleted from datatset.The pipeline descriptor for this post-processor is presented in Snippet 4.1 and its parameters described below:

```yaml
X_DISTANCE_FILTER:
    abs_euclidean_distance_threshold: 0.01
    abs_roll_distance_threshold: 20
    abs_pitch_distance_threshold: 20
    abs_yaw_distance_threshold: 20
```

Snippet 4.1: Distance filter post-processor YAML descriptor example.

- **abs_euclidean_distance_threshold**: absolute euclidean minimum distance (in meters) acceptable to define a non-duplicate candidate;
- **abs_roll_distance_threshold**: absolute roll minimum distance (in degrees) acceptable to define a non-duplicate candidate;
- **abs_pitch_distance_threshold**: absolute pitch minimum distance (in degrees) acceptable to define a non-duplicate candidate;
- **abs_yaw_distance_threshold**: absolute yaw minimum distance (in degrees) acceptable to define a non-duplicate candidate;

4.1.4.3 Tool Center Point Angle Twin

According to the gripper type, some candidates replicas could be acceptable. Considering the symmetry over the TCP reference, it is possible to define an angular twin replica without the search algorithm, thus reducing processing. E.g. two-finger grippers could have a mirror twin (180 degrees over attack axis, Figure 4.13) with same grasping properties. Therefore, the TCP angle twin post-processor is designed to apply this replica operation over all dataset. In summary, its build a candidate twin based on an angular shift (Tait-Briant ZYX-order) over the TCP reference frame. The pipeline descriptor is presented in Snippet 4.7 and its parameters defined as following:

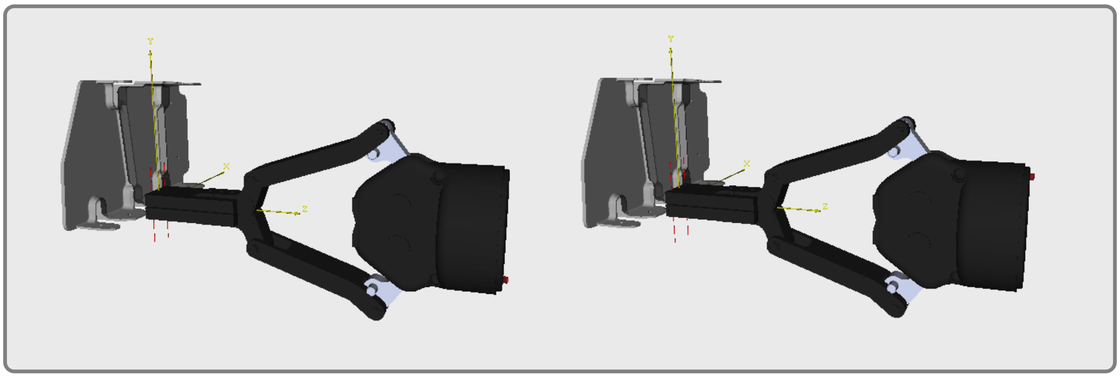


Figure 4.13 – Symmetric grasping replicas over a TCP's symmetry axis.

```
X_TCP_ANGLE_TWIN:
    Rz: 180
    Ry: 0
    Rx: 0
```

Snippet 4.7: TCP angle twin post-processor YAML descriptor example

- **Rz:** angle displacement (in degrees) over the TCP Z-axis.
- **Ry:** angle displacement (in degrees) over the TCP Y-axis.
- **Rx:** angle displacement (in degrees) over the TCP X-axis.

This post-processor generates only one symmetric replica per grasping candidate. If more is needed, this filter can be applied several times in sequence, following by Distance Filter 4.1.4.2 to remove duplicates.

4.1.4.4 Object Angle Twin

Some objects are spherical or radial symmetries, thus some grasping candidates could have symmetric representation 4.14. Therefore, this could be automatically generated by the object angle twin replica post-processor. This characteristic could improve the candidates' search. It build a candidate twin based on a spherical angular shift (azimuth and/or polar angle) over the object reference frame. The radius corresponds to the candidate position, therefore it is not a parameter. It is important to note that, the object reference frame should be defined as spherical/radial symmetry origin, otherwise this post-processor will not correctly work.

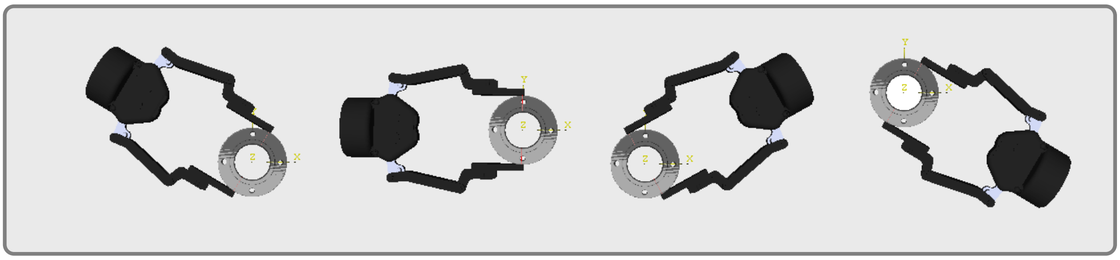


Figure 4.14 – Symmetric grasping replicas over a object's symmetry axis.

```yaml
X_OBJ_ANGLE_TWIN:
    azimuth: 30
    polar: 0
```

Snippet 4.8: Object angle twin post-processor YAML descriptor example

- **azimuth:** azimuth angle displacement (in degrees) over the object reference frame;
- **polar:** azimuth angle displacement in degrees) over the object reference frame;

To correctly deploy the spherical displacement over a candidate, the following relationship between spherical coordinates and transformation matrix is deployed:

$$T = \begin{bmatrix} cos(\Theta) & sin(\Theta) \cdot cos(\phi) & sin(\Theta) \cdot sin(\phi) \\ -sin(\Theta) & cos(\Theta) \cdot cos(\phi) & cos(\Theta) \cdot sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \tag{4.1}$$

4.1.4.5 Viewer based post-processors

Two post-processors are implemented to grasping candidates visualisation: the "GraspIt!" (Andrew T Miller et al. 2004) and the RViz (ROS n.d.[b]) viewer. The viewer post-processor pre-configure, build the visualisation world and call the respective 3D environment. It also load the datatset and set the candidates configuration to visualisation. A menu is responsible to waiting the user input to run over the dataset. This tool is important to visual inspection of the dataset.

To "GraspIt¡ support visualisation, the XML 3D model and the Polygon File Format (extension ".ply") need to be defined to gripper and object, respectively. In other hand, to RViz, a URDF 3D model is expected following by the object's Polygon File Format. The viewer module implementation should takes in consideration which gripper is used to correctly map the DOFs of each URDF joint. Therefore, the candidates' parameters "method/type" and "gripper/type" (Section 4.1.1) lead to help this mapping, since both defined how to read the array parameter DOF. Each viewer YAML descriptor are presented in Snippet 4.9 and 4.10 :

```
1_GRASPIT_VIEWER:
    path:
        gripper_models_file_path: ''
        object_models_file_path: ''
    gripper_file_name: 'robotiq_2f_140_outer_finger/
      robotiq_85_gripper.xml'
```

Snippet 4.9: "GraspIt!" viewer YAML descriptor example.

4.2 Grasping Selection

The grasping selection is a ROS package (based on ROS action server library (ROS n.d.[a])) designed for choosing the best candidate over a set of previously taught grasping poses of an object. This step is an online procedure; i.e., it is a run-time

```yaml
X_RVIZ_VIEWER:
    path:
        gripper_models_file_path: ''
        object_models_file_path: ''
    gripper_file_name: 'robotiq_140_gripper/robotiq_140_gripper.
       urdf'
```

Snippet 4.10: Rviz viewer YAML descriptor example.

process performed during the robot task execution. Thus, this operation needs to be fast and reliable. An illustration of the described procedure is presented in Figure 4.15.

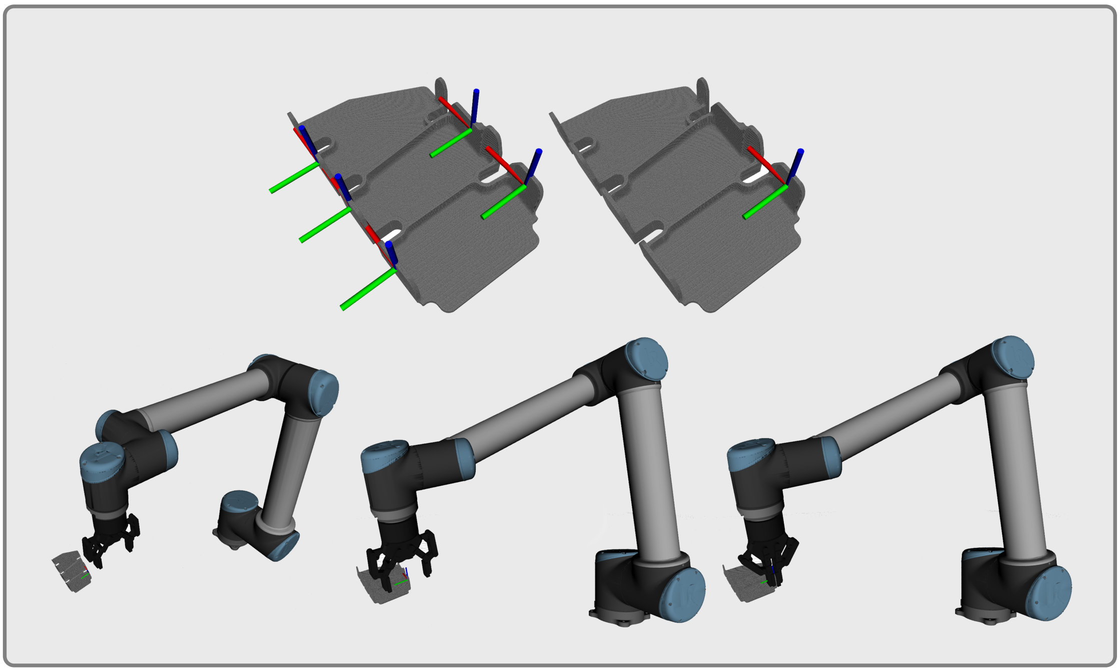


Figure 4.15 – The grasping selection process. (top left) The object with the grasping candidates frames. (top right) The best candidate is chose following the task oriented grasping heuristic. (bottom left) Robot's initial pose. (bottom middle) Approaching movement. (bottom right) Grasping.

Once the candidates dataset are loaded in the parameter server (Figure 4.16), the grasping selection pipeline estimates the best grasp candidate for allowing the robot to pick the object. The best grasping candidate is chosen according to a cascade of heuristics defined by the user in a YAML file for each object detected (Snippet 4.11). Therefore, the object needs to be identified and localized before the grasping selection process. The heuristics cascade gives a score for each grasping candidate related to a reference frame (such as the gripper). The one with the lower cost is the eligible candidate. It is possible to define a weight for each heuristic in the pipeline based on the importance level of each method in the application.

```yaml
0_depth:
    weight: 1
    distance_threshold: 2
1_roll:
    weight: 150
    min_angle_threshold: -35
    max_angle_threshold: 35
2_joints:
    chain_start: 'base_link'
    chain_end: 'gripper_urdf'
    timeout: 0.005
    urdf_param: '/robot_description'
    insist: 10
```

Snippet 4.11: A grasping selection pipeline configuration example.

Since load the database into server demands time, the pipeline allow two types of operation mode: the direct estimation and the pre-load support. The first loads the dataset into rosparam server every time a ROS action's goal request is sent. In other hand, in pre-load support the pipeline requests two ROS action's goals. The first one to load the database into server and the second one to run the estimation pipeline. It is useful to accelerate the procedure. However the user need to be sure that the loaded database correspond to the object in use when run the estimation process (Figure 4.16).

The grasping selection architecture code design is based on polymorphism and

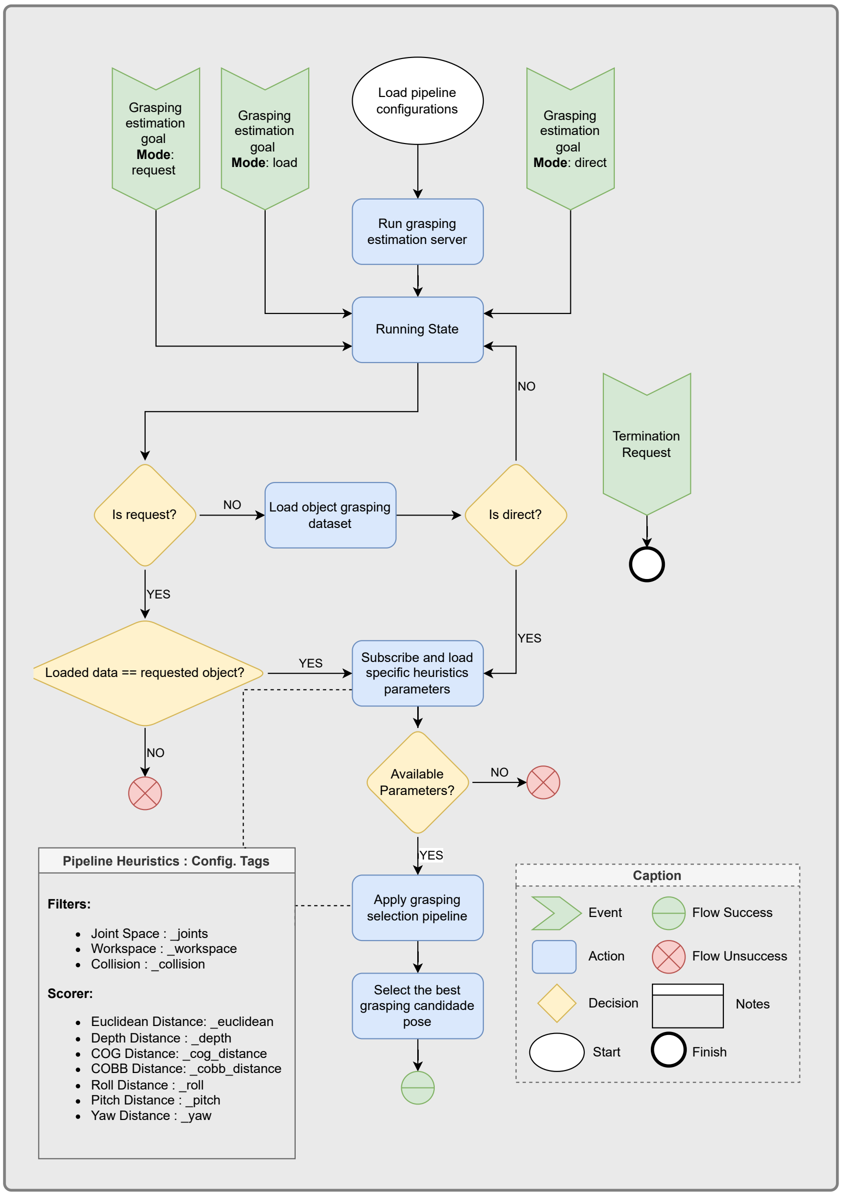


Figure 4.16 – Grasping selection pipeline workflow.

implemented in C++. The grasping selection server allows being incremented with new heuristics classes that inherit a defined parent class. Therefore new approaches can be easily integrated without modifying the grasping selection core architecture and pipeline workflow.

The implemented heuristics for the present thesis proposal are presented from Sections 4.2.1 to 4.2.10

4.2.1 Depth Distance Scorer

The Depth distance scorer is a method to set the grasping candidate cost value according with the depth distance between the TCP reference frame and the candidate. Therefore, this cost allows to select candidates close to depth from the current gripper pose. The pipeline descriptor is presented in Snippet 4.12 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **distance_threshold:** candidates with a distance bigger than the threshold value are discarded.

```
x_depth:
    weight: 1
    distance_threshold: 1
```

Snippet 4.12: Depth distance scorer pipeline descriptor example.

4.2.2 Euclidean Distance Scorer

The euclidean distance scorer sets the grasping candidate cost value according with the euclidean distance between the TCP reference frame and the candidate. This cost allows to choose near candidates from the current gripper pose. The pipeline descriptor is presented in Snippet 4.13 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.

- **distance_threshold:** candidates with a distance bigger than the threshold value are discarded.

```
x_euclidean:
    weight: 1
    distance_threshold: 1
```

Snippet 4.13: Euclidean distance scorer pipeline descriptor example

4.2.3 Center of Gravity Distance Scorer

The Center of Gravity (COG) distance scorer is a method to set the grasping candidate cost value according with the euclidean distance between the candidate and the object reference frame. This cost is important in cases of high dimensional objects, which grasping poses can cause torques over the center of gravity and, affect the equilibrium in grasping movements. The pipeline descriptor is presented in Snippet 4.14 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **distance_threshold:** candidates with a distance bigger than the threshold value are discarded.

4.2.4 Center of Bounding Box Distance Scorer

Following the same motivation of COG distance scorer4.2.3, the Center of Bounding Box (COBB) distance scorer defines the grasping candidate cost based on the euclidean distance between the candidate and the object bounding box reference frame. The pipeline descriptor is presented in Snippet 4.14 and its parameters is described below:

```
x_cog_distance:
    weight: 1
    distance_threshold: 0.15
```

Snippet 4.14: COG distance scorer pipeline descriptor example.

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **distance_threshold:** candidates with a distance bigger than the threshold value are discarded.

```
x_cog_distance:
    weight: 1
    distance_threshold: 0.15
```

Snippet 4.15: COBB distance scorer pipeline descriptor example.

4.2.5 Roll Distance Scorer

The roll distance scorer is a less effort angle displacement selector. This method set the grasping candidate cost value according with the roll distance (TCP X-axis rotation) between the TCP reference frame and the candidate. The pipeline descriptor is presented in Snippet 4.16 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **min_angle_threshold:** candidates with a distance lesser than the threshold value are discarded.

- **max_angle_threshold:** candidates with a distance bigger than the threshold value are discarded.

```
x_roll:
    weight: 150
    min_angle_threshold: −35
    max_angle_threshold: 35
```

Snippet 4.16: Roll distance scorer pipeline descriptor example

4.2.6 Pitch Distance Scorer

Similarly to roll distance scorer4.2.5, the pitch distance scorer sets the grasping candidate cost value according with the pitch distance between the TCP reference frame and the candidate. Therefore, this cost allows to select candidates with less effort in TCP Y-axis rotation. The pipeline descriptor is presented in Snippet 4.17 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **min_angle_threshold:** candidates with a distance lesser than the threshold value are discarded.
- **max_angle_threshold:** candidates with a distance bigger than the threshold value are discarded.

4.2.7 Yaw Distance Scorer

The yaw distance scorer is a method to set the grasping candidate cost value according with the pitch distance between the TCP reference frame and the

```
x_pitch:
    weight: 150
    min_angle_threshold: -35
    max_angle_threshold: 35
```

Snippet 4.17: Pitch distance scorer pipeline descriptor example.

candidate. Therefore, this cost allows to select candidates with less effort in TCP Z-axis rotation. The pipeline descriptor is presented in Snippet 4.18 and its parameters is described below:

- **weight:** a penalty weight value. The importance of the metric in selection pipeline is inversely proportional to the weight value. Therefore, this value must be higher than 1.
- **min_angle_threshold:** candidates with a distance lesser than the threshold value are discarded.
- **max_angle_threshold:** candidates with a distance bigger than the threshold value are discarded.

```
x_yaw:
    weight: 150
    min_angle_threshold: -35
    max_angle_threshold: 35
```

Snippet 4.18: Yaw distance scorer pipeline descriptor example.

4.2.8 Joint Space Filter

In run-time, some grasping candidates can lead the robot to unfeasible kinematic configurations. Thus, aiming to avoid this, the joint space filter heuristic calculates

each candidate kinematic chain and discard the ones that exceed joints thresholds. Besides each candidate, the approach pose to grasp is also evaluated. The thresholds are defined into robot model descriptor for each robot joint in degrees range. The model adopted in this pipeline is the URDF which is supported by ROS.
The pipeline descriptor is presented in Snippet 4.19 and its parameters is described below:

- **urdf_param:** ROS topic name where the URDF robot description is published;
- **chain_start:** robot first link name to be considered in kinematics solver.It should be in accordance with the robot URDF model;
- **chain_end:** robot last link name to be considered in invert kinematics solver. It should be in accordance with the robot URDF model;
- **timeout:** since it is a run-time process, and the solver is optimization process, a timeout (in seconds) should be define to end the solution refinement;
- **insist:** how many iterations to refine the solution, otherwise a random joint space will be generated.

```
x_joints:
    chain_start: 'base_link'
    chain_end: 'gripper_urdf'
    timeout: 0.005
    urdf_param: '/robot_description'
    insist: 10
```

Snippet 4.19: Joint space filter pipeline descriptor example.

4.2.9 Workspace Filter

The workspace filter is method to discard candidates that exceed a spherical workspace thresholds defined. This is useful to eliminate candidates with dangerous approach/lifting vector. For instance, if the center of a box is defined as the sphere's origin, it is possible to avoid candidate that generates approach/lifting vectors that collide with the box border, Figure 4.17. The approach positions evaluated in this heuristic are automatic generated using the offset, w.r.t grasping candidate, defined by the user.

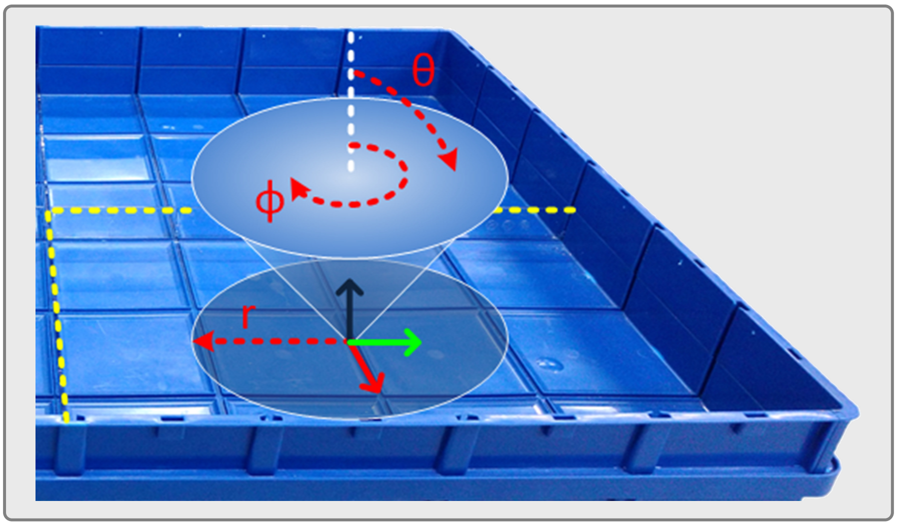


Figure 4.17 – Workspace filter 3D representation.

The pipeline descriptor is presented in Snippet 4.20 and its parameters is described below:

- **ws_base_frame:** frame that define the sphere origin. Z-axis is the reference to angles limits.
- **min_azimuth_threshold ϕ_{min}:** minimum azimuth angle (from -180 to 180 degrees);
- **max_azimuth_threshold ϕ_{min}:** maximum azimuth angle (from -180 to 180 degrees);
- **min_polar_threshold Θ_{min}:** minimum polar angle (from -180 to 180 degrees);
- **max_polar_threshold Θ_{max}:** maximum polar angle (from -180 to 180 degrees);

- **min_radius_threshold** r_{min}**:** minimum radius length (from 0 to +inf meters);
- **max_radius_threshold** r_{max}**:** maximum radius length (from 0 to +inf meters);
- **plot_ws:** tag to enable workspace 3D visualisation. It is useful to debug and calibration but it is unnecessary to perform the grasping task.

```
x_workspace:
    ws_base_frame: 'picking_box_center'
    min_azimuth_threshold: −180
    max_azimuth_threshold: 180
    min_polar_threshold: −70
    max_polar_threshold: 70
    min_radius_threshold: 0
    max_radius_threshold: .55
    plot_ws: false
```

Snippet 4.20: Workspace filter pipeline descriptor example.

4.2.10 Collision Filter

The collision filter is a method that discard candidates that cause collision between the gripper's finger and the scene (or other objects). The fingers trajectory is considered, i.e., the trajectory from open pose to close gripper's finger. The point cloud of the scene must be provided and the collision shape volume must be defined. The Figure4.18 shows an example of a collision volume (i.e. two collision boxes) defined in 3D representation.
The pipeline descriptor is presented in Snippet 4.20 and its parameters is described below:

- **cloud_topic:** input scene cloud to check for collision;
- **collision_threshold:** the number of acceptable collision points between the scene point cloud and the collision volume;

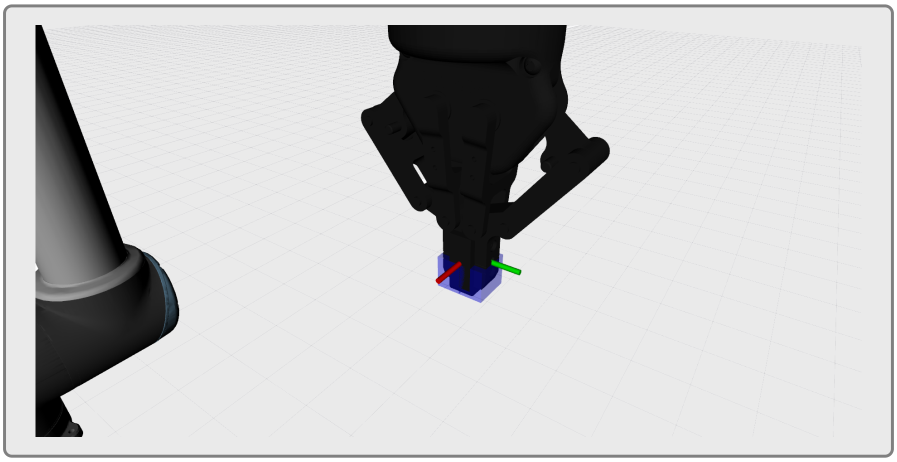


Figure 4.18 – Collision filter 3D representation. The blue boxes define the collision volume.

- **voxel_grid_filter/activate:** activate or not a down-sample voxel grid filter into input scene;
- **voxel_grid_filter/leaf_size:** voxel's grid leaf size;
- **collisions/gripper_tcp_frame** gripper tcp frame id used to define the shape/volume. The TCP must be closed;
- **collisions/show_shapes** tag to enable collision shapes 3D visualisation. It is useful to debug and calibration but it is unnecessary to perform the grasping task.
- **shapes:** collisions' shapes/volume definition such as pose and dimension.

```yaml
x_collision:
    cloud_topic: "/ambient_point_cloud"
    collision_threshold: 500
    voxel_grid_filter:
        activate: true
        leaf_sizes: [.002, .002, .002]
    collisions:
        show_shapes: false
        gripper_tcp_frame: "gripper"
        shapes:
            0_box:
                    position:
                        x: 0
                        y: 0.0055
                        z: 0.0110
                    orientation:
                        x: 0
                        y: 0
                        z: 0
                        w: 1
                    dimension:
                      x: 0.020
                      y: 0.011
                      z: 0.022
            1_box:
                    position:
                      x: 0
                      y: -0.0055
                      z: 0.0110
                    orientation:
                      x: 0
                      y: 0
                      z: 0
                      w: 1
                    dimension:
                      x: 0.020
                      y: 0.011
                      z: 0.022
```

Snippet 4.21: Collision filter pipeline descriptor example.

5 Proposal Validation

6 Conclusion

Referências bibliográficas

ABB (2022). *Dual-arm YuMI IRB 14000.* (Accessed 2022-05-06). URL: https://webshop.robotics.abb.com/us/catalog/product/view/id/84/s/dual-arm-yumi-irb-14000-assembly/category/3/.

Aleotti, Jacopo and Stefano Caselli (2012). "A 3D shape segmentation approach for robot grasping by parts". In: *Robotics and Autonomous Systems* 60.3, pp. 358–366. ISSN: 09218890. DOI: 10.1016/j.robot.2011.07.022. URL: http://dx.doi.org/10.1016/j.robot.2011.07.022.

Alonso, Marcos, Alberto Izaguirre, and Manuel Graña (2018). "Current research trends in robot grasping and bin picking". In: *The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications.* Springer, pp. 367–376.

Andrew T, Miller and Allen Peter K (2001). "GraspIt! A Versatile Simulator for Robotic Grasping". PhD thesis. Columbia University, p. 140. URL: http://www.cs.columbia.edu/%7B~%7Dcmatei/graspit/pdf/GraspIt%7B%5C_%7DRA04-complete-system.pdf%7B%5C%%7D5Cnpapers2://publication/uuid/1054081B-B727-46D2-B0FA-9364E77F53CF.

Asif, Umar, Jianbin Tang, and Stefan Harrer (2018). "EnsembleNet: Improving Grasp Detection using an Ensemble of Convolutional Neural Networks." In: *BMVC*, p. 10.

Babin, Vincent, David St-Onge, and Clément Gosselin (2019). "Stable and repeatable grasping of flat objects on hard surfaces using passive and epicyclic mechanisms". In: *Robotics and Computer-Integrated Manufacturing* 55, pp. 1–10.

Baier-Lowenstein, Tim and Jianwei Zhang (Oct. 2007). "Learning to grasp everyday objects using reinforcement-learning with automatic value cut-off". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1551–1556. ISBN: 978-1-4244-0911-2. DOI: 10.1109/IROS.2007.4399053. URL: http://ieeexplore.ieee.org/document/4399053/.

Bejczy, Antal K. (1980). "Sensors, controls, and man-machine interface for advanced teleoperation". In: *Science* 208.4450, pp. 1327–1335.

Bicchi, Antonio (1995). "On the closure properties of robotic grasping". In: *The International Journal of Robotics Research* 14.4, pp. 319–334.

Bicchi, Antonio and Vijay Kumar (2000). "Robotic grasping and contact: a review". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. IEEE, pp. 348–353. ISBN: 0-7803-5886-4. DOI: 10.1109/ROBOT.2000.844081. URL: http://ieeexplore.ieee.org/document/844081/.

Björnsson, Andreas, Marie Jonsson, and Kerstin Johansen (2018). "Automated material handling in composite manufacturing using pick-and-place systems–a review". In: *Robotics and Computer-Integrated Manufacturing* 51, pp. 222–229.

Boularias, Abdeslam, James Andrew Bagnell, and Anthony Stentz (2015). "Learning to manipulate unknown objects in clutter by reinforcement". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Castiello, Umberto (2005). "The neuroscience of grasping". In: *Nature Reviews Neuroscience* 6.9, pp. 726–736.

Castro, André L. et al. (2020). “AdaptPack Studio: an automated intelligent framework for offline factory programming”. In: *Industrial Robot: the international journal of robotics research and application*.

Chen, Jiankui et al. (2020). “Active curved surface deforming of flexible conformal electronics by multi-fingered actuator”. In: *Robotics and Computer-Integrated Manufacturing* 64, p. 101942.

Chen, Lu, Panfeng Huang, Yuanhao Li, et al. (Feb. 2020). “Detecting Graspable Rectangles of Objects in Robotic Grasping”. In: *International Journal of Control, Automation and Systems* 18.X, pp. 1–10. ISSN: 1598-6446. DOI: 10.1007/s12555-019-0186-2. URL: http://link.springer.com/10.1007/s12555-019-0186-2.

Chen, Lu, Panfeng Huang, and Zhongjie Meng (Mar. 2019). “Convolutional multi-grasp detection using grasp path for RGBD images”. In: *Robotics and Autonomous Systems* 113, pp. 94–103. ISSN: 09218890. DOI: 10.1016/j.robot.2019.01.009. URL: https://doi.org/10.1016/j.robot.2019.01.009%20https://linkinghub.elsevier.com/retrieve/pii/S0921889018307346.

Choi, Changhyun et al. (2018). “Learning object grasping for soft robot hands”. In: *IEEE Robotics and Automation Letters* 3.3, pp. 2370–2377.

Chu, Fu-Jen, Ruinian Xu, and Patricio A. Vela (Oct. 2018). “Real-World Multiobject, Multigrasp Detection”. In: *IEEE Robotics and Automation Letters* 3.4, pp. 3355–3362. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2852777. arXiv: 1802.00520. URL: https://ieeexplore.ieee.org/document/8403246/.

Ciocarlie, Matei T. and Peter K. Allen (2009). “Hand posture subspaces for dexterous robotic grasping”. In: *International Journal of Robotics Research* 28.7, pp. 851–867. ISSN: 02783649. DOI: 10.1177/0278364909105606.

Components, ROS (n.d.). *VH Hand*. (Accessed 2022-05-06). URL: https://www.roscomponents.com/en/robotic-hands/114-svh-hand.html#/svh_hand_training-no/svh_hand_left_or_right_hand-left_hand.

Costa, Carlos M (2022). *Object Recognition*. (Accessed 2022-05-24). URL: https://github.com/carlosmccosta/object_recognition.

Costa, Carlos M. et al. (2016). "Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms". In: *Robotics and Autonomous Systems* 76, pp. 113–140. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2015.09.030. URL: https://www.sciencedirect.com/science/article/pii/S0921889015002250.

CUDA Toolkit Documentation v11.0.3 (n.d.). URL: https://docs.nvidia.com/cuda/index.html.

D'Avella, Salvatore, Paolo Tripicchio, and Carlo Alberto Avizzano (2020). "A study on picking objects in cluttered environments: Exploiting depth features for a custom low-cost universal jamming gripper". In: *Robotics and Computer-Integrated Manufacturing* 63, p. 101888.

de Souza, João Pedro Carvalho, António Amorim, et al. (2021). "Industrial Robot Programming by Demonstration using Stereoscopic Vision and Inertial Sensing". In: *Industrial Robot: the international journal of robotics research and application*. ISSN: 0143-991x. DOI: 10.1108/IR-02-2021-0043.

de Souza, João Pedro Carvalho, Luís F. Rocha, et al. (2021). "Robotic grasping: from wrench space heuristics to deep learning policies". In: *Robotics and Computer-Integrated Manufacturing* 71, p. 102176. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102176. URL: https://www.sciencedirect.com/science/article/pii/S0736584521000594.

Ding, Dan, Yun-Hui Liu, and Shuguo Wang (2000). "Computing 3-D optimal form-closure grasps". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 4. IEEE, pp. 3573–3578.

Dini, Gino and Franco Failli (2000). "Planning grasps for industrial robotized applications using neural networks". In: *Robotics and Computer-Integrated Manufacturing* 16.6, pp. 451–463.

Dizioğlu, B. and K. Lakshiminarayana (1984). "Mechanics of form closure". In: *Acta mechanica* 52.1-2, pp. 107–118.

Ferrari, C. and J. Canny (1992). "Planning optimal grasps". In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE Comput.

Soc. Press, pp. 2290–2295. ISBN: 0-8186-2720-4. DOI: 10.1109/ROBOT.1992.219918. URL: http://ieeexplore.ieee.org/document/219918/.

Ferreira, Marcos et al. (2016). “Stereo-based real-time 6-DoF work tool tracking for robot programing by demonstration”. In: *The International Journal of Advanced Manufacturing Technology* 85.1-4, pp. 57–69.

Festo (n.d.). *Bernoulli grippers OGGB*. (Accessed 2022-05-06). URL: https://www.festo.com/lt/en/p/bernoulli-grippers-id_OGGB/.

Gariepy, Alexandre et al. (2019). “GQ-STN: Optimizing One-Shot Grasp Detection based on Robustness Classifier”. In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 3996–4003. ISSN: 21530866. DOI: 10.1109/IROS40897.2019.8967785. arXiv: 1903.02489.

Ghazaei, Ghazal et al. (2019). “Dealing with Ambiguity in Robotic Grasping via Multiple Predictions”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11364 LNCS, pp. 38–55. ISSN: 16113349. DOI: 10.1007/978-3-030-20870-7_3. arXiv: 1811.00793.

Goldfeder, Corey et al. (Apr. 2007). “Grasp Planning via Decomposition Trees”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. April. IEEE, pp. 4679–4684. ISBN: 1-4244-0602-1. DOI: 10.1109/ROBOT.2007.364200. URL: http://ieeexplore.ieee.org/document/4209818/.

Goodale, Melvyn A et al. (1991). “A neurological dissociation between perceiving objects and grasping them”. In: *Nature* 349.6305, pp. 154–156.

Guo, Di et al. (May 2017). “A hybrid deep architecture for robotic grasp detection”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1609–1614. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989191. URL: http://ieeexplore.ieee.org/document/7989191/.

He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Ingber, L. (1989). “Very Fast Simulated Re-Annealing”. In: 12.8, pp. 967–973.

IROS (2020). *Robotic Grasping and Manipulation Competition*. (Accessed 2022-05-02). URL: https://rpal.cse.usf.edu/competition_iros2020/.

Jacquard dataset (2018). (Accessed 2020-05-08). URL: https://jacquard.liris.cnrs.fr/.

Jain, Siddarth and Brenna Argall (May 2016). "Grasp detection for assistive robotic manipulation". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2016-June. IEEE, pp. 2015–2021. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487348. URL: http://ieeexplore.ieee.org/document/7487348/.

James, Stephen, Andrew J Davison, and Edward Johns (2017). "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task". In: *arXiv preprint arXiv:1707.02267*.

Jaśkowski, Maciej et al. (2018). "Improved GQ-CNN: Deep learning model for planning robust grasps". In: *arXiv preprint arXiv:1802.05992*.

Jia-Wei Li, Hong Liu, and He-Gao Cai (Feb. 2003). "On computing three-finger force-closure grasps of 2-d and 3-d objects". In: *IEEE Transactions on Robotics and Automation* 19.1, pp. 155–161. ISSN: 1042-296X. DOI: 10.1109/TRA.2002.806774. URL: http://ieeexplore.ieee.org/document/1177174/.

El-Khoury, Sahar and Anis Sahbani (2009). "On computing robust N-finger force-closure grasps of 3D objects". In: *2009 IEEE international conference on robotics and automation*. IEEE, pp. 2480–2486.

Kirkpatrick, S, C D Gelatt, and M P Vecchi (1983). "Optimization by Simulated Annealing". In: *Science* 220.4598, pp. 671–680.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kumra, Sulabh and Christopher Kanan (2017). "Robotic grasp detection using deep convolutional neural networks". In: *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe, pp. 769–776. ISSN: 21530866. DOI: 10.1109/IROS.2017.8202237. arXiv: 1611.08036.

Lab, Robot Learning (n.d.). *Cornell Dataset.* (Accessed 2020-05-08). URL: http://pr.cs.cornell.edu/grasping/rect%5C_data/data.php.

Large Scale Visual Recognition Challenge (ILSVRC) (n.d.). (Accessed 2020-05-19). URL: http://www.image-net.org/challenges/LSVRC/.

Lawrence, Charlotte (n.d.). *The Future of Robotics.* (Accessed 2020-05-19). URL: http://amazonpickingchallenge.org/.

Lenz, Ian, Honglak Lee, and Ashutosh Saxena (Apr. 2015). “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* 34.4-5, pp. 705–724. ISSN: 0278-3649. DOI: 10.1177/0278364914549607. arXiv: 1301.3592. URL: http://journals.sagepub.com/doi/10.1177/0278364914549607.

Levine, Sergey et al. (Apr. 2018). “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4-5, pp. 421–436. ISSN: 0278-3649. DOI: 10.1177/0278364917710318. arXiv: arXiv:1603.02199v4. URL: http://journals.sagepub.com/doi/10.1177/0278364917710318.

Li, Jia-Wei, Hong Liu, and He-Gao Cai (2003). “On computing three-finger force-closure grasps of 2-D and 3-D objects”. In: *IEEE Transactions on Robotics and Automation* 19.1, pp. 155–161.

Liu, Yun-Hui (1999). “Qualitative test and force optimization of 3-D frictional form-closure grasps using linear programming”. In: *IEEE Transactions on Robotics and Automation* 15.1, pp. 163–173.

— (2000). “Computing n-finger form-closure grasps on polygonal objects”. In: *The International journal of robotics research* 19.2, pp. 149–158.

Liu, Yun-Hui, Miu-Ling Lam, and Dan Ding (2004). “A complete and efficient algorithm for searching 3-D form-closure grasps in the discrete domain”. In: *IEEE Transactions on Robotics* 20.5, pp. 805–816.

Magnetics, Goudsmit (n.d.). *Magnetic grippers.* (Accessed 2022-05-06). URL: https://www.goudsmitmagnets.com/solutions/magnetic-handling/lifting-handling-magnets/magnetic-grippers.

Mahler, Jeffrey and Ken Goldberg (2017). “Learning deep policies for robot bin picking by simulating robust grasping sequences”. In: *Conference on robot learning* CoRL, pp. 515–524. URL: http://berkeleyautomation.github.io/dex-net..

Mahler, Jeffrey, Jacky Liang, et al. (July 2017). “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”. In: *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation. ISBN: 978-0-9923747-3-0. DOI: 10.15607/RSS.2017.XIII.058. eprint: 1703.09312. URL: http://arxiv.org/abs/1703.09312%20http://www.roboticsproceedings.org/rss13/p58.pdf.

Mahler, Jeffrey, Matthew Matl, Xinyu Liu, et al. (2017). “Dex-Net 3.0: Computing Robust Robot Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning”. In: arXiv: 1709.06670. URL: http://arxiv.org/abs/1709.06670.

Mahler, Jeffrey, Matthew Matl, Vishal Satish, et al. (2019). “Learning ambidextrous robot grasping policies”. In: *Science Robotics* 4.26, eaau4984. DOI: 10.1126/scirobotics.aau4984.

Mahler, Jeffrey, Florian T. Pokorny, et al. (May 2016). “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: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2016-June. IEEE, pp. 1957–1964. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487342. URL: http://ieeexplore.ieee.org/document/7487342/.

Miller, Andrew T and Peter K Allen (2004). “Graspit! a versatile simulator for robotic grasping”. In: *IEEE Robotics & Automation Magazine* 11.4, pp. 110–122.

Miller, AT et al. (2003). “Automatic grasp planning using shape primitives”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. IEEE, pp. 1824–1829. ISBN: 0-7803-7736-2. DOI: 10.1109/ROBOT.2003.1241860. URL: http://ieeexplore.ieee.org/document/1241860/.

Monkman, Gareth J et al. (2007). *Robot grippers*. John Wiley & Sons.

Morales, Antonio et al. (2006). "Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5663–5668.

Moreira, E. et al. (2016). "Assessment of Robotic Picking Operations Using a 6 Axis Force/Torque Sensor". In: *IEEE Robotics and Automation Letters* 1.2, pp. 768–775. DOI: 10.1109/LRA.2016.2524043.

Morrison, Douglas, Peter Corke, and Jürgen Leitner (2020). "Learning robust, real-time, reactive robotic grasping". In: *The International Journal of Robotics Research* 39.2-3, pp. 183–201.

Mousavian, Arsalan, Clemens Eppner, and Dieter Fox (Oct. 2019). "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation". In: *The IEEE International Conference on Computer Vision (ICCV)*.

Murray, Richard M et al. (1994). *A mathematical introduction to robotic manipulation*. CRC press.

Nguyen, V.-D. (1987a). "Constructing force-closure grasps in 3D". In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. Institute of Electrical and Electronics Engineers, pp. 240–245. ISBN: 0818607874. DOI: 10.1109/ROBOT.1987.1088014. URL: http://ieeexplore.ieee.org/document/1088014/.

— (1987b). "Constructing stable grasps in 3D". In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. Institute of Electrical and Electronics Engineers, pp. 234–239. DOI: 10.1109/ROBOT.1987.1088008. URL: http://ieeexplore.ieee.org/document/1088008/.

Oztop, E. and M. A. Arbib (2001). "A biologically inspired learning to grasp system". In: *Annual Reports of the Research Reactor Institute, Kyoto University* 1, pp. 857–860. ISSN: 04549244. DOI: 10.1109/iembs.2001.1019077.

Parallel gripper HGPC-16-A (Nov. 2017). FESTO.

Pas, Andreas ten, Marcus Gualtieri, et al. (Dec. 2017). "Grasp Pose Detection in Point Clouds". In: *The International Journal of Robotics Research* 36.13-14, pp. 1455–1473. ISSN: 0278-3649. DOI: 10.1177/0278364917735594. arXiv:

1706.09911. URL: http://journals.sagepub.com/doi/10.1177/0278364917735594.

Pas, Andreas ten and Robert Platt (2018). “Using Geometry to Detect Grasp Poses in 3D Point Clouds”. In: pp. 307–324. ISBN: 9783319515328. DOI: 10.1007/978-3-319-51532-8_19. arXiv: arXiv:1011.1669v3. URL: http://link.springer.com/10.1007/978-3-319-51532-8%7B%5C_%7D19.

Pelossof, Raphael et al. (2004). “An SVM learning approach to robotic grasping”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2004.4, pp. 3512–3518. ISSN: 10504729. DOI: 10.1109/robot.2004.1308797.

piab (n.d.). *piCOBOT*. (Accessed 2022-05-06). URL: https://www.piab.com/robot-and-cobot-gripping-solutions/cobots-and-robot-grippers/picobot-vacuum-gripper-unit/picobot/.

Pinto, Lerrel and Abhinav Gupta (May 2016). “Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3406–3413. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487517. URL: http://ieeexplore.ieee.org/document/7487517/.

Platt, Robert (2007). “Learning grasp strategies composed of contact relative motions”. In: *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 49–56.

Ponce, Jean and Bernard Faverjon (1995a). “On Computing Three-Finger Force-Closure Grasps of Polygonal Objects”. In: *IEEE Transactions on Robotics and Automation* 11.6, pp. 868–881. ISSN: 1042296X. DOI: 10.1109/70.478433.

— (1995b). “On computing three-finger force-closure grasps of polygonal objects”. In: *IEEE Transactions on robotics and automation* 11.6, pp. 868–881.

Prattichizzo, Domenico and Jeffrey C Trinkle (2016). “Grasping”. In: *Springer handbook of robotics*. Springer, pp. 955–988.

Przybylski, Markus, Tamim Asfour, and Rüdiger Dillmann (2011). “Planning grasps for robotic hands using a novel object representation based on the medial axis transform”. In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 1781–1788. DOI: 10.1109/IROS.2011.6048625.

Rakesh, Venkataramani et al. (2018). “Optimizing force closure grasps on 3D objects using a modified genetic algorithm”. In: *Soft Computing* 22.3, pp. 759–772.

Redmon, Joseph and Anelia Angelova (May 2015). “Real-time grasp detection using convolutional neural networks”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2015-June. June. IEEE, pp. 1316–1322. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139361. arXiv: 1412.3128. URL: http://ieeexplore.ieee.org/document/7139361/.

Rezzoug, Nasser and Philippe Gorce (2002). “A multistage neural network architecture to learn hand grasping posture”. In: *IEEE International Conference on Intelligent Robots and Systems* 2.October, pp. 1705–1710. DOI: 10.1109/irds.2002.1044001.

Roa, Máximo A and Raúl Suárez (2009). “Finding locally optimum force-closure grasps”. In: *Robotics and Computer-Integrated Manufacturing* 25.3, pp. 536–544.

— (Jan. 2015). “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38.1, pp. 65–88. ISSN: 0929-5593. DOI: 10.1007/s10514-014-9402-3. URL: http://link.springer.com/10.1007/s10514-014-9402-3.

Robotics, Empire (n.d.). *Versaball*. (Accessed 2022-05-06). URL: https://www.empirerobotics.com/products/.

Robotics, New Scale (2022). *NSR-MTM-3-URe Multi-Tool Mount System for e-Series Universal Robots (UR)*. (Accessed 2022-05-06). URL: https://www.newscalerobotics.com/products/nsr-mtm/.

Robotiq (n.d.). *2F-85 and 2F-140 Grippers*. (Accessed 2022-05-06). URL: https://robotiq.com/products/2f85-140-adaptive-robot-gripper.

ROS (n.d.[a]). *ROS Action Library*. (Accessed 2022-05-10). URL: http://wiki.ros.org/actionlib.

— (n.d.[b]). *RViz*. (Accessed 2022-05-13). URL: http://wiki.ros.org/rviz.

Rossler, B., Jianwei Zhang, and A. Knoll (n.d.). “Visual guided grasping of aggregates using self-valuing learning”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 4. IEEE, pp. 3912–3917. ISBN: 0-7803-7272-7. DOI: 10.1109/ROBOT.2002.1014336. URL: http://ieeexplore.ieee.org/document/1014336/.

Salisbury, J Kenneth and B Roth (1983). “Kinematic and force analysis of articulated mechanical hands”. In.

Santello, Marco, Martha Flanders, and John F Soechting (2002). “Patterns of Hand Motion during Grasping and the Influence of Sensory Guidance”. In: 22.4, pp. 1426–1435.

Saxena, Ashutosh, Justin Driemeyer, and Andrew Y Ng (Feb. 2008). “Robotic Grasping of Novel Objects using Vision”. In: *The International Journal of Robotics Research* 27.2, pp. 157–173. ISSN: 0278-3649. DOI: 10.1177/0278364907087172. URL: http://journals.sagepub.com/doi/10.1177/0278364907087172.

Schmalz (2022a). *FM-SW 76x22 4x6 N10.* (Accessed 2022-05-20). URL: https://www.schmalz.com/en/vacuum-technology-for-automation/vacuum-components/area-gripping-systems-and-end-effectors/vacuum-area-gripping-systems-fx-fm/area-gripping-systems-fm-sw-306896/10.01.11.00851/.

— (2022b). *Phoxi 3D Scanner S.* (Accessed 2022-05-24). URL: https://www.photoneo.com/products/phoxi-scan-s/.

Schunk (n.d.[a]). *PGN-plus-E.* (Accessed 2022-05-06). URL: https://schunk.com/us_en/gripping-systems/series/pgn-plus-e/.

— (n.d.[b]). *PWG-plus.* (Accessed 2022-05-06). URL: https://schunk.com/de_en/gripping-systems/series/pwg-plus/.

Seita, Daniel et al. (2016). “Large-scale supervised learning of the grasp robustness of surface patch pairs”. In: *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR).* IEEE, pp. 216–223.

Song, Yanan et al. (2020). “A novel robotic grasp detection method based on region proposal networks”. In: *Robotics and Computer-Integrated Manufacturing* 65, p. 101963.

Souza, João Pedro C. de, André L. Castro, et al. (2020). “AdaptPack studio translator: translating offline programming to real palletizing robots”. In: *Industrial Robot: the international journal of robotics research and application.*

Souza, João Pedro C. de, Carlos M. Costa, et al. (2021). "Reconfigurable Grasp Planning Pipeline with Grasp Synthesis and Selection Applied to Picking Operations in Aerospace Factories". In: *Robotics and Computer-Integrated Manufacturing* 67, p. 102032. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.102032. URL: http://www.sciencedirect.com/science/article/pii/S073658452030243X.

Suleman, Muhammad Umar and Mian M Awais (2011). "Learning from demonstration in robots: Experimental comparison of neural architectures". In: *Robotics and Computer-Integrated Manufacturing* 27.4, pp. 794–801.

Trottier, Ludovic, Philippe Giguere, and Brahim Chaib-Draa (2017). "Sparse dictionary learning for identifying grasp locations". In: *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, pp. 871–879. DOI: 10.1109/WACV.2017.102.

Viereck, Ulrich et al. (2017). "Learning a visuomotor controller for real world robotic grasping using simulated depth images". In: *arXiv preprint arXiv:1706.04652*.

Watson, Joe, Josie Hughes, and Fumiya Iida (2017). "Real-World, Real-Time Robotic Grasping with Convolutional Neural Networks". In: *Conference Towards Autonomous Robotic Systems*. Vol. 2, pp. 617–626. ISBN: 9783319641072. DOI: 10.1007/978-3-319-64107-2_50. URL: http://link.springer.com/10.1007/978-3-319-64107-2%7B%5C_%7D50.

Wheeler, D. S., A. H. Fagg, and R. A. Grupen (2002). "Learning prospective pick and place behavior". In: *Proceedings - 2nd International Conference on Development and Learning, ICDL 2002*, pp. 197–202. DOI: 10.1109/DEVLRN.2002.1011865.

Yun Jiang, Stephen Moseson, and Ashutosh Saxena (May 2011). "Efficient grasping from RGBD images: Learning using a new rectangle representation". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3304–3311. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980145. URL: http://ieeexplore.ieee.org/document/5980145/.

Zeng, Andy, Shuran Song, Stefan Welker, et al. (Oct. 2018). "Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots* *and Systems (IROS)*. IEEE, pp. 4238–4245. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8593986. arXiv: 1803.09956. URL: https://ieeexplore.ieee.org/document/8593986/.

Zeng, Andy, Shuran Song, Kuan-Ting Yu, et al. (Aug. 2019). "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching". In: *The International Journal of Robotics Research*, p. 027836491986801. ISSN: 0278-3649. DOI: 10.1177/0278364919868017. URL: http://journals.sagepub.com/doi/10.1177/0278364919868017.

A Appendix

Technical and Theoretical Background

A.0.1 Gripper Technologies

End effectors are hardware devices of handily mechanisms (e.g., robots and automation systems) aiming them to interact with the environment. Two classes of interaction are: passive and active. In passive interaction, the end effectors are composed of sensors (e.g., inspection, quality assurance, and surveillance applications). Meanwhile, the active interaction consists of direct interaction with the workpiece (e.g., welding, cutting, drilling, screwing, grinding, painting, and grasping different objects).

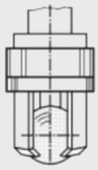
Figure A.1 – Commercial grippers examples. From top-left to down-right: magnetic (Magnetics n.d.), parallel fingers (Schunk n.d.[a]), angular fingers (Schunk n.d.[b]), adaptive two-fingers (Robotiq n.d.), dual suction (piab n.d.), Bernoulli(Festo n.d.), adaptive three-fingers (Robotiq n.d.), anthropomorphic (Components n.d.), Versaball (E. Robotics n.d.)

The grippers are a complex class of active end-effectors. They are active links to handle workpieces (Monkman et al. 2007). Grippers need to be flexible and versatile according to the application and object that they handle. Nowadays, with the development of new technologies, the variety of grippers and hardware allows more functionality, although it demands new efforts (e.g. modelling and control). In that way, hardware evaluation is also a part of the development of grasping estimation. The structure of the gripper defines a classification of grasping,

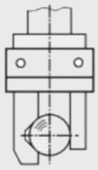
related to the handling of the objects, i.e., hold an object with an emphasis in security (enveloping grasp) or dexterity and sensitivity (dexterous manipulation). The main characteristic of the dexterous is the manipulation of the object with fingers. Meanwhile, the enveloping grasping wraps the object with the palm and the finger (Bicchi and Kumar 2000) and (Alonso et al. 2018).
Figure A.1 and Figure A.3 elucidate some technologies applied in the concept of different grippers. The most usual forms of grippers are (Monkman et al. 2007):

- **Impactive**: the gripper realizes impactive forces against the surface of the workpiece. Some examples are the finger grippers.
- **Astrictive**: the gripper generates a field responsible for producing a binding force, as an air movement (vacuum suction), magnetic or electrostatic effect.
- **Contigutive**: the gripper touches a surface making contact prehension. The adhesion may be chemical or thermal effects.
- **Ingressive**: the gripper permeates the surface of the workpiece. This ingression can be intrusive (pins) or non-intrusive (e.g., hook and loop).

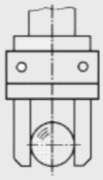
The use of automatic tool change devices is an option to improve the flexibility, and applicability of the grippers over the wide variety of objects and workpieces. Besides that, the use of handling machines or dual-arm robots in the automatic process is also a valid alternative (Figure A.4). Both solutions increase the complexity of the system, and the grasp estimation evaluation may determine which tool is best to complete the task to each object to grasp.



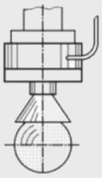
(a) Pure enclosing without clamping.



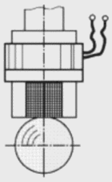
(b) Partial form fit combined with clamping force.



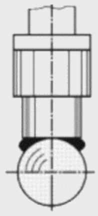
(c) Pure force closure.



(a) Holding with vacuum air (pneumatic force closure).



(b) Retention using magnetic field (force field).



(c) Retention using adhesive media.

Figure A.3 – Forms of grippers according to (Monkman et al. 2007)

Figure A.4 – (left) Multi-arm robot with different grippers (ABB 2022). (right) Multi-gripper end-effector (N. S. Robotics 2022)

A.0.2 Multi-Fingered Grasping

A multi-fingered grasp is realized over a set of contacts between the active pairs (the workpiece and the gripper). Therefore, the determination of a suitable configuration of independent grasp points is the primary step of the fingered grasp planning.

The wrench vectors describe the forces and moments that influence a rigid body's dynamic. These vectors can be used to formulate grasp locations, and a wrench vector is presented below:

$$\mathbf{w_c} = \begin{bmatrix} \mathbf{F} \\ \tau \end{bmatrix} \tag{A.1}$$

where $\mathbf{F}$ and τ are the vector representations of the forces and the moments. The wrench vectors have 3 and 6 DOFs in the case of IR^2 and IR^3, respectively.

The contact models can be categorized as friction-less contact, friction contact (also named hard finger contact), and soft contact (Murray et al. 1994). The focus of this paper will be the friction contact, since this model is sufficient for the picking application addressed in this paper.

The friction contact model considers the mechanical interaction between the active pairs. Therefore, the wrench convex depends on the friction contact forces, described by Coulomb model of friction: Considering the normal force f_n, and the tangential force f_t, static friction occurs when there is no slipping between the two surfaces of contact, that is when $|f_t| \leq \mu_t f_n$ where μ is a positive value representing the static tangential coefficient of friction. Figure A.5 shows an example of hard finger contact, the geometric representation of the Coulomb's law and the friction cone convex also defined as FC_{c_i}.

A wrench representation, w.r.t the i-th contact point (c_i), is defined as follows:

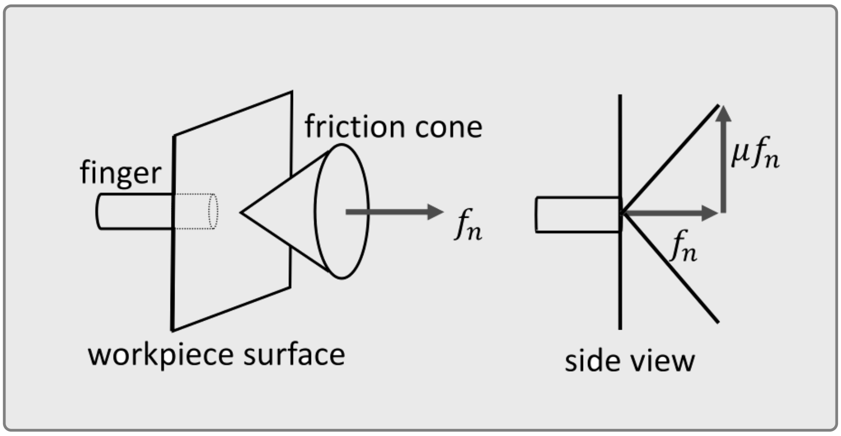


Figure A.5 – Friction contact model and the geometric representation of Coulomb's law (figure based on (Murray et al. 1994)).

$$W_{c_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{f}_{c_i}, \quad \mathbf{f}_{c_i} \in FC_{c_i} \tag{A.2}$$

where $FC_{c_i} = \mathbf{f} \in R^3 : \sqrt{f_x^2 + f_y^2} \leq \mu_t f_z,, f_z \geq 0$, and μ_t is the transversal friction coefficient in c_i.

Therefore, it is possible to define the matrix that compose the wrench vector:

$$\mathbf{W}_{c_i} = \mathbf{B}_{c_i}\mathbf{f}_{c_i}, \quad \mathbf{f}_{c_i} \in FC_{c_i} \tag{A.3}$$

where B_{c_i} is the wrench basis matrix with dimension $p \times n$ where p is the DOFs and n the number of independent forces and moments that constitutes f_{c_i}. The contact model discussed here has as reference frame the one with the origin coincident with the contact point itself. It is more convenient to refer all contacts in a grasp model to a common frame, generally the center of mass of the work piece. Therefore the wrench transformation matrix is defined as follows:

$$
{}^{o}\mathbf{T}w_{c_i} = \begin{bmatrix} {}^{o}\mathbf{R}_{\mathrm{c_i}} & 0 \\ {}^{o}\hat{\mathbf{t}}_{\mathrm{c_i}} {}^{o}\mathbf{R}_{\mathrm{c_i}} & {}^{o}\mathbf{R}_{\mathrm{c_i}} \end{bmatrix} \in IR^3 \tag{A.4}
$$

where ${}^{o}\mathbf{R}_{c_i}$ and ${}^{o}\mathbf{t}_{c_i}$ are the rotation and translation matrix of the i-th contact point (c_i) w.r.t. object frame (o). The $\hat{\mathbf{t}}$ is the linear operator representing the cross product ${}^{o}\mathbf{t}_{c_i} \times {}^{o}\mathbf{R}_{c_i}$ as bellow:

$$
\widehat{\boldsymbol{a}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{A.5}
$$

Hence, the contact map $\mathbf{G}_i$ is defined as follows:

$$
\mathbf{G}_i =^{o} \mathbf{Tw}'_{c_i} \mathbf{B}_{c_i} \tag{A.6}
$$

Note that it describes the direction of each component of the i-th applied wrench and defines the constraints of the contact. The grasp map is the matrix with all contact maps that characterize the contact model (it is also named constraint matrix):

$$
\mathbf{G} = \begin{bmatrix} {}^{o}\mathbf{Tw}'_{c_1}\mathbf{B}_{c_1} & \dots & {}^{o}\mathbf{Tw}'_{c_N}\mathbf{B}_{c_N} \end{bmatrix} \tag{A.7}
$$

Then, including the magnitude of the forces, a workpiece wrench can be written:

$$
{}^{o}\mathbf{W} = [\mathbf{G}_1, \dots, \mathbf{G}_N]\,[\mathbf{f}_{c_1}, \dots, \mathbf{f}_{c_N}]' = \mathbf{GF} \tag{A.8}
$$

where: $\mathbf{F} \in FC$ and $FC = FC_{c_1} \times \ldots \times FC_{C_N}$

The ${}^{o}\mathbf{W}$ also defines the GWS (grasp wrench space) of the grasp. It is obtained by means of the L_∞ or L_1 norm. The L_∞ defines the GWS ($\mathbf{W}_{L_\infty}$) considering the limitation of the maximum allowable normal contact force, while L_1 defines the GWS ($\mathbf{W}_{L_1}$) by the sum magnitude of the normal contact forces. The norms operation yields to:

$$
\begin{aligned}
\mathbf{W}_{L_1} &= \text{ConvexHull}\left(\bigcup_{c_i}^{N} \mathbf{w}_{p1c_i}, \ldots, \mathbf{w}_{pD_{c_i}}\right) \\
\mathbf{W}_{L\infty} &= \text{ConvexHull}\left(\bigoplus_{ci} \{\mathbf{w}_{p_1 ci}, \ldots, \mathbf{w}_{pD_{c_i}}\}\right)
\end{aligned}
\tag{A.9}
$$

where $\mathbf{w}_{pd\ c_i} \in {}^{o}\mathbf{W}$ and $\bigoplus$ is the Minkowski sum. More detail about the norm operation can be verified in (Ferrari et al. 1992).

TODO: Inserir demais conceitos descritos no deliverable Fasten

The concept of grasp closure evaluates the restraining of an object. A common assumption is the force-closure implies an equilibrium, but the inverse does not apply. A grasp has its convex hull defined by the wrenches that constitute the grasp configuration, i.e., the matrix ${}^{o}\mathbf{W}$. In a force-closure grasp, the convex hull includes the wrench space origin $\{O\}$, see Figure A.6. According to the definition presented in (Salisbury et al. 1983), if all wrenches in ${}^{o}\mathbf{W}$ positively span the entire wrench space, the grasp will be force-closure. Figure A.6 shows a grasp wrench space (GWS) and a convex hull of grasp configuration for force and non-force-closure, for a planar case with a fixed value for the moment (τ) in the z-axis. Therefore, it is considered $\mathbf{f}_{c_i} \in R^2 : f_{c_i} = (f_x, f_y)$, and the resistance to perturbation in both force axes is evaluated.

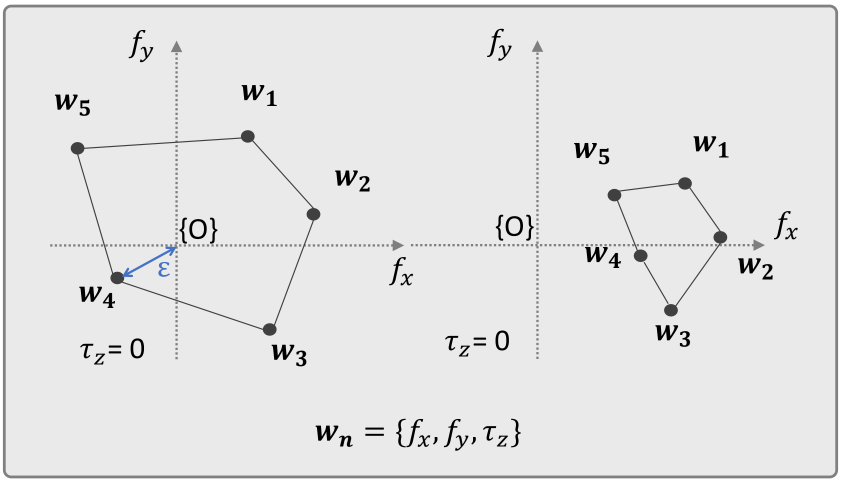


Figure A.6 – Wrench convex-hull configuration. Force-closure and the ϵ-value (left). Non-force-closure (right).

Since several configurations can reach a force-closure grasp, quality metrics like ϵ-metric evaluate which one is best. The ϵ is a normalized value that represents the wrench vector's distance to the origin ($\{O\}$), which is the shortest, i.e., the worst wrench vector to support an external perturbation. An efficient grasp, ideally, has $\epsilon = 1$. The left GWS of Figure A.6 elucidates this metric and, the readers are encouraged to a more detailed review of this grasping definition in (Ferrari et al. 1992).

A.0.3 Simulated Annealing Grasping

The SANN algorithm (Ciocarlie et al. 2009) integrated in the "GraspIt!" simulator (Andrew T et al. 2001) is one of the tools that the grasping pipeline relies on. Since it was used from the perspective of an end user, a brief explanation will be done here, and any further information can be retrieved on the referenced papers.

The SANN is a heuristic optimization algorithm based on the cooling of a set of atoms to a minimum state of energy, and it was first introduced by (Kirkpatrick et al. 1983) in a Statistical Mechanics optimization algorithm application. The "Very Fast Simulated Re-Annealing" was an improvement made by Ingber at (Ingber 1989) and used here. Since it is based on temperature, Ingber proposed that its cooling process decrease as described by Equation A.10

$$T = T_0 \cdot exp(-k^{1/D}) \tag{A.10}$$

where D is the dimensional search space, k a SANN parameter step, and T_0 is the SANN the initial temperature.

Each algorithm iteration generates new state variables following a rule of neighboring. Considering current and a new variable state as $S_{current}$ and S_{new}, this rule yields Equation A.11.

$$S_{new} = S_{current} + T \cdot (-1)^{round(Rand(0,1))} \cdot \left(1 + \frac{1}{T}\right)^{Rand(-1,1)} \tag{A.11}$$

and the probability to change the state between the current and the new one is defined by Equation A.12 where $Q(\bullet)$ represents the objective function of the optimization problem.

$$exp(\frac{Q(S_{current}) - Q(S_{new})}{T}) > Rand(0,1) \tag{A.12}$$

Regarding the multi-fingered grasp procedure, the objective function to be optimized by SANN need to be related to the hand posture **p** and, the position and orientation of the wrist **w** as follows:

$$F_{ob} = f(\mathbf{p}, \mathbf{w}), \quad \mathbf{p} \in \mathcal{R}^d, w \in \mathcal{R}^6 \tag{A.13}$$

where d is the number of intrinsic DOFs of the hand.

As discussed by (Ciocarlie et al. 2009), the hand posture is defined by *eigengrasps*, a subspace of movement based on how human generate hand postures. The *eigengrasps* reduces the DOFs of the hand based on how humans select appropriate grasps and hand postures. Studies show that humans simplify, unconsciously, the problem with a pattern in the movement. More information can be verified in (Ciocarlie et al. 2009; Santello et al. 2002). The *eigengrasp* ($\mathbf{e}_i$) is defined by hand, and it is a d-dimensional direction vector that represents the motion of a group joint space that constitute it. Therefore, a posture can be defined by Equation A.14.

$$p = \mathbf{p}_m + \sum_{i=1}^{b} a_i \mathbf{e}_i \tag{A.14}$$

with posture origin defined by $\mathbf{p}_m$ and b the total number of *eigengrasps*. Since it is a linear combination, the parameter array $\mathbf{a} = [a_0, a_1, ..., a_b]$ will be the optimization variable in Equation A.13 together with the $\mathbf{w}$. Therefore, the dimensional search space D has a reduced length, i.e. $D = sizeof(\mathbf{a}) + sizeof(\mathbf{w})$.

The optimization algorithm tries to minimize the linear and angular distance of the Interest Contact Point (ICP) that constitute the ICR (Figure A.7) adjusting the discussed optimization variables $\mathbf{a}$ and $\mathbf{w}$. The ICR is a contact region model (a predefined group of distributed ICPs) used to calculate the interaction of the algorithm, thus it is possible to create a feasible procedure. Therefore, the objective function to be minimized is describe by Equation A.15, where N is the number of total contacts in ICR, $\hat{\mathbf{n}}_i$ is the surface normal, $\mathbf{o}_i$ the distance between the ICP and the object ($i \in N$). The scalar α is a range adjustment factor between the distance and the normalized dot product of the second sum part. It is important to note that the mapping between F_{ob} and Q is realized by simulated interaction in the "GraspIt!" (Andrew T et al. 2001). A detailed description of the algorithm procedure is presented in (Ciocarlie et al. 2009).

$$Q = \sum_{i=1}^{N} (1 - \delta_i) \quad \text{with} \quad \delta_i = \frac{|\mathbf{o}_i|}{\alpha} + \left(1 - \frac{\hat{\mathbf{n}}_i \cdot \mathbf{o}_i}{|\mathbf{o}_i|}\right) \tag{A.15}$$

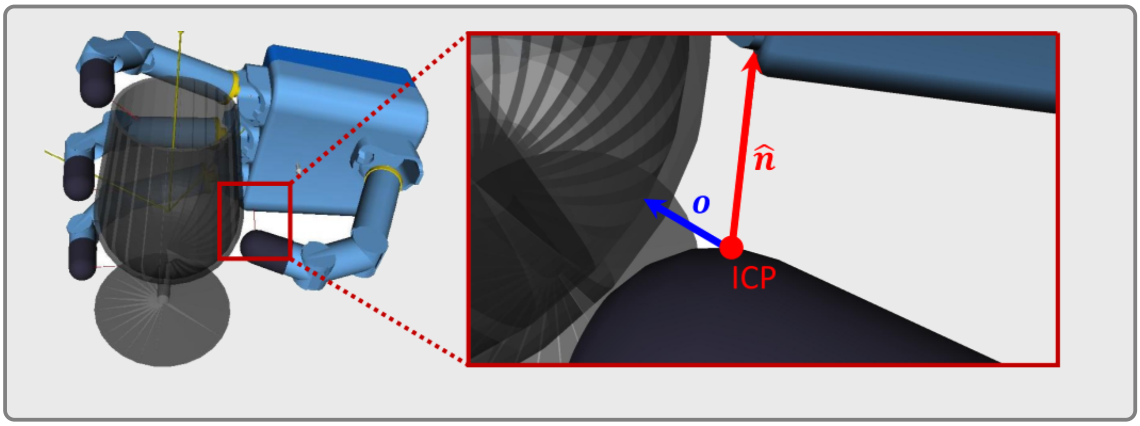


Figure A.7 – Grasp optimization process elucidation.

The algorithm proposed by (**p21:ciocarlie2009hand**), and used in this proposal, is replicated in Algorithm 1.
As already mentioned, the variables that define the states are **a**, related to the *eigengrasp*, and **w**, related to the wrist pose. The "*ObjFunc*" is described by Equation A.15. The "*Ngbr*" is the calculating of neighbouring of the state variables using the Equation A.11 and, "*Probability*" is the function to perform the probability of jump to a new state according to Equation A.12. Since this step of the algorithm is based on *Graspit!*, the "ForwardKinematics" and collision check are are parts of this tool.

```
forall variables of CurrentState do
|  CurrentState.variable = RandomValue()
end
QCurrent = ObjFunc(CurrentState);
Iterations = 0;
QSaved = 0;
while Iterations ≠ MaxIterations do
    Generate a new state as a neighbor of current state;
    repeat
        forall variables of NewState do
            Sim. Annealing neighbor generation function;
            NewState.variable = Ngbr(CurrentState.variable);
        end
        Apply ForwardKinematics(NewState);
        if collisions detected or joint limits exceeded then
            legalState = false;
        end
        else
            legalState = true
        end
    until legalState == true;
    QNew = ObjFunc(NewState);
    if QNew > QSaved then
        Insert NewState in SavedStatesList;
        QSaved = lowest ObjFunc value in SavedStateList;
    end
    Sim. Annealing probability of "jumping" to new state;
    PJump = Probability(QCurrent, QNew);
    if PJump > 0.5 then
        CurrentState = NewState;
        QCurrent = QNew;
    end
    Iterations = Iterations + 1;
end
```


Algorithm 1: SANN applied to grasping

B Appendix

Technical Data

B.0.1 Handler Tool Electronic Schematic

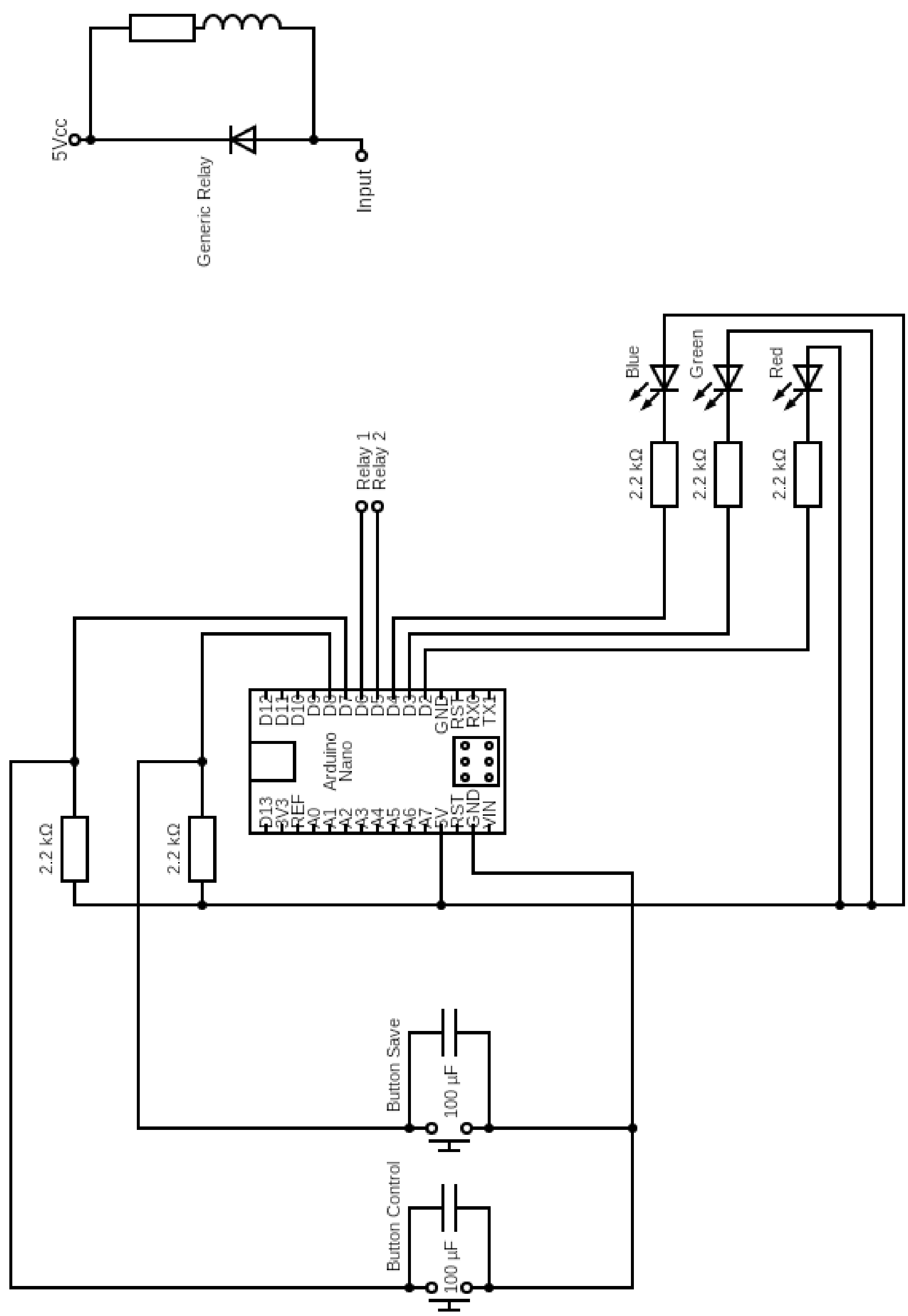


Figure B.1 – Handler Tool Electronic Schematic.