



**UNIVERSIDADE FEDERAL DA BAHIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

CAIO CRISTIANO BARROS VITURINO

**GRASPING AND IDENTIFYING OBJECTS IN
UNSTRUCTURED ENVIRONMENTS WITH DEEP
LEARNING METHODS.**

Salvador
2023

CAIO CRISTIANO BARROS VITURINO

**GRASPING AND IDENTIFYING OBJECTS IN UNSTRUCTURED
ENVIRONMENTS WITH DEEP LEARNING METHODS.**

Thesis presented to the Graduate Program in
Electrical Engineering of the Federal University
of Bahia in partial fulfillment of the require-
ments for the degree of Doctor in Electrical En-
gineering.

Supervisor: Prof. Dr. André Gustavo Scolari Con-
ceição

Salvador
2023

GRASPING AND IDENTIFYING OBJECTS IN UNSTRUCTURED ENVIRONMENTS WITH DEEP LEARNING METHODS.

TESE DE DOUTORADO

Autor: Caio Cristiano Barros Viturino
Orientador: André Gustavo Scolari Conceição - UFBA

Tese de doutorado aprovada em 01 de Dezembro de 2023 pela banca examinadora composta pelos seguintes membros:



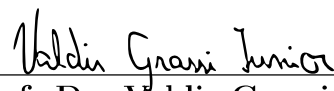
Prof. Dr. André Gustavo Scolari Conceição
Supervisor - Universidade Federal da Bahia (UFBA)



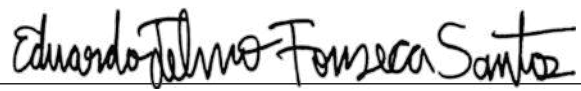
Prof. Dr. Tiago Trindade Ribeiro
Universidade Federal da Bahia (UFBA)



Prof. Dr. Eduardo Furtado de Simas Filho
Universidade Federal da Bahia (UFBA)



Prof. Dr. Valdir Grassi Junior
Universidade de São Paulo - USP



Prof. Dr. Eduardo Telmo Fonseca Santos
Instituto Federal da Bahia - IFBA

*À minha esposa Caroline Viturino e ao meu filho Enzo
Viturino*

ACKNOWLEDGMENTS

I would like first to express my deepest appreciation to my adviser Prof. Dr. André Gustavo Scolari Conceição for the opportunity to grow as a researcher in robotics and artificial intelligence field. Without his help, cooperation and encouragement, I would not have made progress in the project.

This study has received funding from SEPIN/MCTI under the 4th Coordinated Call BR-EU in CIT and from the European Unions Horizon 2020 research and innovation programme under the Grant Agreement No 777096. This study also received financial support from the CNPQ grant term number 311029/2020-5.

I would like to thank FAPESB (Fundação de Amparo à Pesquisa do Estado da Bahia) for their financial support. This study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

RESUMO

Nos últimos anos, os métodos de prensão robótica que se baseiam em aprendizado profundo têm superado os métodos tradicionais. No entanto, a maioria desses métodos utiliza prensões planares devido ao alto custo computacional associado às prensões em 6D. As prensões planares, apesar de terem um custo computacional mais baixo, apresentam limitações espaciais que restringem sua aplicabilidade em ambientes complexos, como a manipulação de objetos produzidos por impressoras 3D. Algumas técnicas de prensão robótica geram apenas uma prensão viável por objeto. No entanto, é essencial obter múltiplas prensões possíveis por objeto, pois nem todas as prensões geradas possuem soluções válidas da cinemática inversa do robô ou evitam colisões com obstáculos próximos. Para superar essas limitações, propõe-se um método de prensão robótica que é capaz de gerar múltiplas prensões seletivas 6D por objeto, evitando colisões com obstáculos adjacentes. Os testes de prensão foram realizados em uma Unidade de Fabricação Aditiva, que apresenta um alto nível de complexidade devido à possibilidade de colisões entre o efetuador final e a parte interna da impressora. Os resultados experimentais demonstram que é possível alcançar uma taxa de sucesso de 62% na prensão 6D de peças manufaturadas em ambientes confinados. Além disso, foi alcançada uma taxa de sucesso de 68% e 177 MPPH (Man Picks Per Hour) na prensão seletiva planar de objetos posicionados sobre superfícies planas. O braço robótico UR5, a câmera Intel Realsense D435 e o efetuador final Robotiq 2F-140 foram utilizados para validar o método proposto em experimentos reais.

Palavras-chave: Prensão Robótica, Redes Neurais Convolucionais, Manipuladores robóticos

ABSTRACT

In recent years, robotic grasping methods based on deep learning have outperformed traditional methods. However, most of these methods use planar grasps due to the high computational cost associated with 6D grasps. Planar grasps, despite having a lower computational cost, have spatial limitations that restrict their applicability in complex environments, such as grasping objects inside 3D printers. Some robotic grasping techniques generate only one viable grasp per object. However, it is essential to obtain multiple possible grasps per object, as not all generated grasps have viable kinematic solution or avoid collisions with nearby obstacles. To overcome these limitations, a robotic grasping method is proposed that is capable of generating multiple selective 6D grasps per object, avoiding collisions with adjacent obstacles. Grasping tests were carried out in an Additive Manufacturing Unit, which presents a high level of complexity due to the possibility of collisions between the end effector and the inside of the printer. Experimental results indicate that it is possible to achieve a success rate of 62% in the 6D grasp of manufactured parts in confined environments. In addition, a success rate of 68% and 177 MPPH (Man Picks Per Hour) was achieved in the selective planar grasp of objects positioned on flat surfaces. The UR5 robotic arm, the Intel Realsense D435 camera, and the Robotiq 2F-140 end effector were used to validate the proposed method in real experiments.

Keywords: Robotic grasping, Convolutional Neural Network, Robotic Manipulators

LIST OF PUBLICATIONS

- Viturino, C. C. B. ; Conceição, André G. S. Selective 6D Grasping with a Collision Avoidance System Based on Point Clouds and RGB+D Images. *ROBOTICA*, 2023, 1-16. DOI: 10.1017/S0263574723001364
- Viturino, C. C. B. ; Oliveira, D. M. ; Conceição, A. G. S. ; Junior, U. M. P. 6D Robotic Grasping System using Convolutional Neural Networks and Adaptive Artificial Potential Fields with Orientation Control. In: *Latin American Robotics Symposium*, 2021. DOI: 10.1109/LARS/SBR/WRE54079.2021.9605472
- Oliveira, D. M. ; Viturino, C. C. B. ; Conceição, A. G. S. 6D Grasping Based On Lateral Curvatures and Geometric Primitives. In: *Latin American Robotics Symposium*, 2021. DOI: 10.1109/LARS/SBR/WRE54079.2021.9605382
- Viturino, C. C. B. ; Conceição, A. G. S. Preensão robótica seletiva em 6D utilizando Redes Neurais Convolucionais. In: *Simpósio Brasileiro de Automação Inteligente*, 2021. DOI: 10.20906/sbai.v1i1.2597
- Viturino, C. C. B. ; Santana, K. L. ; Oliveira, D. M. ; Lemos, C. B. ; Conceição, A. G. S. Redes Neurais Convolucionais para Identificação e Preensão Robótica de Objetos. In: *XXIII Congresso Brasileiro de Automática*, 2020. DOI: 10.48011/asba.v2i1.1163
- Viturino, C. C. B. ; Junior, U. M. P. ; Conceição, A. G. S. ; Schnitman, L. Adaptive Artificial Potential Fields with Orientation Control Applied to Robotic Manipulators. In: *21st IFAC World Congress, Germany*, 2020. DOI: 10.1016/j.ifacol.2020.12.2706
- Junior, U. M. P. ; Viturino, C. C. B. ; Conceição, A. G. S. ; Schnitman, L. Sistema Anticolisão Aplicado a Manipuladores Robóticos Baseado em Campos Potenciais Artificiais. In: *14^o Simpósio Brasileiro de Automação Inteligente*, 2019. DOI: 10.17648/sbai-2019-111278

LIST OF FIGURES

1.1	Tabletop scenario for 4D grasping testing.	6
1.2	Additive Manufacturing Unit for 6D grasping testing.	6
1.3	Overview of the proposed 4D grasping pipeline (Chapter 3).	8
1.4	Overview of the proposed 6D grasping pipeline (Chapter 4).	8
2.1	Flowchart that describes the commonly adopted processes in several grasping methods as well as the subsection in which each process is presented.	11
2.2	(a) Antipodal point of a point on a circle (b) antipodal grasp definition. .	12
2.3	Types of grasp adopted by several authors. The grasping position is defined as (x, y, z) , and the orientation as θ, ψ, ϕ , corresponding to the roll (θ), pitch (ψ), and yaw (ϕ) angles.	13
2.4	c depicts an observed contact point, b constitute the 3-DOF rotation, ω is the predicted grasp width, d is the distance from baseline to base frame (SUNDERMEYER et al., 2021).	14
2.5	Grasp representation in 3D (NI et al., 2020).	14
2.6	The grasp sampling simulation used by Mousavian, Eppner and Fox (2019) and Murali et al. (2020). The FleX sim (VICENT et al., 2016) was used to sample the grasps.	21
2.7	Training data pipeline adopted by Sundermeyer et al. (2021). Grasps are sampled from object's point cloud. The objects are in stable positions. Only the grasps that are in contact with the objects are considered. . . .	21
2.8	(a) Pile and (b) Packed object configuration adopted in Breyer et al. (2021). .	22
2.9	(Left) Pile and (Right) Packed object configuration adopted in Jiang et al. (2021).	22
2.10	(a) Grasping model. (b) Best grasp and supplementary grasp (c) Sampled grasps varying from low to high inferred quality.	23
2.11	Grasping pipeline stages, including grasp synthesis, trajectory planning (before approaching), and grasp execution (NEWBURY et al., 2023). . .	30
2.12	Objects used by different authors to test grasping algorithms.	34
2.13	Objects used by Morrison, Corke and Leitner (2018) to test the performance of the robotic grasping algorithm presented.	35
2.14	(a) Level 1 objects consisting of prismatic and circular solids (b) Level 2 objects including clear plastic and household objects with a lower level of graspability. (c) Level 3 objects with adversarial geometry and material properties (d) Level 4 objects with different reflectances and material properties which affect the ability to form a vacuum seal on the object surface (MAHLER et al., 2019).	36
2.15	3D-printed objects for grasping evaluation performance. These test objects varies in terms of complex geometry (left to right), and graspability (bottom to top).	37
2.16	Phases considered for benchmarking in pick-and-place tasks (MNYUSIWALLA et al., 2020).	38
2.17	Protocol scenarios sorted by object type and amount of clutter (MNYUSIWALLA et al., 2020).	39

2.18	(Left) M-BBT and T-BBT-block templates following the test template. (Right) BBT-100 blocks randomly placed inside a bin (MORGAN et al., 2019).	40
2.19	Samples of rearrangement tasks: initial scenes (a, c) and target scenes (b, d) (LIU et al., 2021).	40
2.20	Grasping benchmark environment. The radius of the circular workspace is optional and must be reported by the authors. (BEKIROGLU et al., 2019).	41
2.21	Object locations and poses considered by the benchmark in Bekiroglu et al. (2019).	42
2.22	Layouts defined within the benchmark proposed by Bottarel et al. (2020).	43
3.1	Since GG-CNN cannot distinguish between fixed and movable objects, a grasp can be generated on fixed objects such as a bin. To generate a grasp, Generative Grasping Convolutional Neural Network (GG-CNN) uses only the area inside the red box, represented in (b) and (c). This experiment was conducted in the Laboratory of Robotics (LaR) at UFBA and is available at youtube.com/watch?v=tebXgisPew .	54
3.2	Using the GG-CNN, a grasp is not generated for objects outside the GG-CNN area (red box).	54
3.3	(Left) Grasp coordinate frame representation $\mathbf{g} = (\mathbf{p}, \phi, \omega, q)$ and the other frames related to the camera and robot. $\mathbf{p} = (x, y, z)$ describes the gripper's center point on the object, ϕ denotes the gripper angle around z axis, ω describes the gripper width and q represents the grasp quality (Right) Representation of $\tilde{\mathbf{g}} = (\mathbf{s}, \tilde{\phi}, \tilde{\omega}, q)$ on the depth image $\mathbf{I}$, where $\mathbf{s}$ indicates the gripper's center point in pixels, $\tilde{\phi}$ denotes the gripper angle and $\tilde{\omega}$ refers to the gripper width.	56
3.4	Cornell Grasping Dataset (LENZ; LEE; SAXENA, 2015) positive and negative grasps represented as rectangles as in Jiang, Moseson and Saxena (2011).	57
3.5	(a) Grasp angle representation considering $\cos(\Phi_T) > 0$ (b) grasp angle discontinuity around $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.	58
3.6	Graphical representation of the two components of a unit vector $\sin(2\tilde{\Phi}_T)$ and $\cos(2\tilde{\Phi}_T)$.	59
3.7	GG-CNN Architecture.	60
3.8	The ground truth images $\sin(2\tilde{\Phi}_T)$, $\cos(2\tilde{\Phi}_T)$, $\tilde{\mathbf{Q}}_T$ and $\tilde{\mathbf{W}}_T$, generated from the Cornell Grasping dataset. The white pixels of the images corresponding to the Grasp Quality, Grasp Angle, Grasp Width and $\sin(2\tilde{\Phi}_T)$ are equal to zero. The white pixels of the image corresponding to $\cos(2\tilde{\Phi}_T)$ are equal to one.	61
3.9	(a) Each rectangle comprises the tool center point, planar gripper angle, and gripper width. These attributes were shown in different rectangles to facilitate understanding. (b) Antipodal gripper angle is represented in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.	62
3.10	The index M of the max value in the quality image is used to acquire the value of the grasp width, angle, and the height of the grasp by using the depth measurement.	63
3.13	Test objects proposed by Mahler et al. (2017). (a) 3D Model (b) 3D-Printed objects. The objects are called: (A) Bar clamp, (B) Nozzle, (C) Part 3, (D) Gear Box, (E) Part 1, and (F) Vase.	66
3.14	Grasping pipeline.	69

3.15	Depth cloud acquired from the simulated Intel Realsense D435 camera in RViz (GOSSOW et al., 2020) (a) Depth cloud representation of the depth image $\mathbf{I}$ and the best visible grasp $\tilde{\mathbf{g}}_b^*$ (b) Depth cloud representation of the filtered depth image $\mathbf{I}_f$ acquired in stage 9 and the best visible grasp $\tilde{\mathbf{g}}_f^*$.	70
3.16	(a) UR5 Robot at the Laboratory of Robotics (LaR) at UFBA (b) the Intel Realsense D435 custom support developed (c) virtual model of the UR5 robot and the custom camera support developed.	71
3.17	The condensed version of the selective grasping ROS architecture employed. The SSD and GG-CNN nodes are used to perform the object recognition and generate the grasp pose. The pivotal SSGG-CNN Main node orchestrates interactions between the user, the Webots simulator, and the nodes linked to SSD and GG-CNN. Its role is to manage the sequence of detection and grasp generation requests, ensuring their orderly execution for correct robotic grasping and object detection.	73
3.18	Grasps executed by the proposed grasping pipeline (Figure 3.14). The Grasp trial axis represents the number of grasps executed by the robot. The dashed green line represents the number of successful grasps executed by the robot using the GG-CNN model.	75
3.19	(a) Simulated environment in Webots (b) Camera's view from the support shown in Figure 3.16 (c) Bin where the objects are placed after the grasp action.	77
4.1	Overview of the proposed grasping pipeline.	80
4.2	Objects (MAHLER et al., 2017) used to test the robotic grasping pipeline proposed in this work.	82
4.3	Dataset generation pipeline using Webots.	83
4.4	Some figures part of the dataset automatically generated in Webots using real images as background. This dataset was used to train the Mask R-CNN.	84
4.5	Detection examples performed using real object images.	85
4.6	Signed distance used to verify collisions between the point cloud and the Robotiq 2F-140 collision mesh.	86
4.7	Collision check system in an additive manufacturing system. (a) Workspace configuration used for testing (b) There are no points inside the gripper collision mesh (c) There are points of the 3D printer or the object point cloud inside the collision mesh of the gripper.	87
4.8	Relationship between the number of points of the workspace point cloud and the time required to verify the collision. It is required $0.8s$ to calculate the signed distance for $1 \cdot 10^4$ points.	88
4.9	Grasping pipeline using GraspNet (MOUSAVIAN; EPPNER; FOX, 2019), Mask R-CNN (HE et al., 2017), and a collision check system based on point clouds and collision meshes. The simulation is performed in Webots (WEBOTS, 2021) using the virtual model of the UR5 robot manipulator, Robotiq 2F-140 and Intel Realsense D435.	89
4.10	Real UR5 at the Laboratory of Robotics (UFBA) and simulated UR5 in Webots.	90
4.11	Successful and failed grasps generated in each step of the ablation study using the simulated environment.	91
4.12	Grasps performed in simulation by employing the ablation study referred in Section 4.9. (1) Grasping pipeline from stages 1 to 6, (2) Grasping pipeline from stages 1 to 10, and (3) Complete grasping pipeline. The objects employed in the experiments are shown in section 4.5. The curve 1 for the part Nozzle is under the curve 2. In other words, there were no successful grasps in the 20 attempts.	93

4.13	Time to generate a grasp for each object used in the experiments. (1) Bar clamp, (2) Gear Box, (3) Nozzle, (4) Part 1, (5) Part 3, (6) Vase. The processing times were based on the ablation study referred in Section 4.9. It is noticed that the entire grasping pipeline is time consuming due to the collision check with the point cloud. However, grasp success is considerably increased as shown in Figure 4.12. Note that the x axis of the graphs have different scales.	94
4.14	Relationship between time threshold and success rate. The lower the time threshold is, the lower is the success rate due to the time consumed by the entire grasping pipeline presented in Section 4.9.	95
4.15	Comparison between GraspNet and GG-CNN using two different grasping pipelines. The SSD512-ResNet50-COCO was employed to detect the object of interest in both pipelines. The images depict a simulation performed in Webots.	96
4.16	Grasps executed by the GraspNet and GG-CNN, employing SSD512-ResNet50 to identify the objects. It was applied the grasping pipeline detailed in Figure 4.15.	97
4.17	Time to generate a grasp for each object used in the experiments. It is important to note the different time scales in the x-axis for the GraspNet (seconds) and GG-CNN (milliseconds). In the graph, the objects are represented as (1) Bar clamp, (2) Gear Box, (3) Nozzle, (4) Part 1, (5) Part 3, (6) Vase. It is important to note that GraspNet requires more time because it generates a set of grasps rather than a single one like GG-CNN does. Besides that, GraspNet has more parameters than GG-CNN and explore the entire 6D space to generate non-planar grasps.	98
4.18	Comparison between Graspnet and GG-CNN using the grasping pipeline of the Figure 4.15.	99
4.19	Grasps performed in printed objects inside a 3D printer. Video link: youtube.com/watch?v=APXHeSNuZYU	100
4.20	Poses S and F for each object.	101
4.21	Grasps performed by GraspNet using Mask R-CNN as the object detector. To analyze the repeatability, the objects position were kept the same. S means the object pose that is easier to grasp and F is the object pose harder to grasp. These poses were randomly determined.	101

LIST OF TABLES

2.1	Specification of the most used datasets in robotic grasping applications. .	18
2.2	Comparative table comprising the dataset used in each related work, the process adopted to augment the dataset, and the amount of data generated by the augmentation process. Some authors do not perform data augmentation or do not provide any related information.	24
2.3	Networks comparative table.	29
2.4	Experimental procedures, operating system, and hardware (graphics card, robot, gripper and camera), adopted by each related work.	48
2.5	Comparative table containing the average success rate, number of parameters, and processing time to generate grasp poses using unknown objects. These objects are divided in Adversarial Objects (A.O.) (MAHLER et al., 2017), Household Objects (H.O.), and objects proposed by Viereck et al. (2017) (V.). The type refers to a real (R) or simulated (S) evaluation environment. The results are divided into open-loop (O.L.), closed-loop (C.L.), and according to the Definition of a Successful Grasps (DSGs) used in subsection 2.6.4.	50
2.6	Comparative table of the related work.	52
3.3	Evaluation of the average precision by class and mean average precision considering the IoU metric with threshold of 0.5, 0.75, and 0.5 : 0.95. . .	74
3.4	Grasp and recognition fail per object considering the SSD512-ResNet50-COCO as the object detector. 20 grasps were performed per object. . . .	76
3.5	Evaluation of the grasping reliability using the Man Picks Per Hour (MPPH) metric as suggested in Mahler et al. (2018).	76
4.1	Successful and failed grasping comparison of the ablation study using the simulated environment.	92
4.2	GraspNet and GG-CNN comparison	97

LIST OF ABBREVIATIONS AND ACRONYMS

CGN	Candidate Generation Network
CNN	Convolutional Neural Network
DSG	Definition of a Successful Grasp
EGAD	Envolved Grasping Analysis Dataset
FPS	Frames Per Second
GMM	Gaussian Mixture Model
GQ-CNN	Grasp Quality Convolutional Neural Network
GG-CNN	Generative Grasping Convolutional Neural Network
GG-CNN2	Generative Grasping Convolutional Neural Network 2
IoU	Intersection-over-union
MPPH	Mean Picks Per Hour
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
NGDF	Neural Grasp Distance Fields
QAN	Gaussian Mixture Model
RAM	Reliable Adjustment Module
ReLU	Rectified Linear Unit
ROS	Robot Operating System
SETA	Successful task Executions over Total Attempts
SGT	Simulated Grasping Trial
SGD	Stochastic Gradient Descent
SSD	Single Shot Multibox Detector
VAE	Variational Autoencoders

LIST OF SYMBOLS

Unless otherwise noted, the following mathematical symbols have the following meanings:

c	Robot's base Cartesian coordinate frame
$\mathbf{c}_i$	RGB image considering all the obstacles in the environment
$\mathbf{I}$	Raw depth image with all objects in the environment
$\mathbf{I}_f$	Filtered depth image with objects of interest
$\tilde{\mathbf{I}}_T$	Ground truth depth image input
H	Height of an image in pixels
W	Width of an image in pixels
$\mathbb{R}^{H \times W}$	Matrix of height H and width W
$\mathbb{R}^{i \times H \times W}$	Set of i matrices of height H and width W
$p[m]$	Grasp center point in $\mathbf{C}$
q	Grasp quality
s	Gripper's center point in $\mathbf{I}$
ω	Gripper width in $\mathbf{C}$
$\tilde{\omega}[m]$	Gripper width in $\mathbf{I}$
$\phi[rad]$	Planar gripper angle in $\mathbf{C}$
$\tilde{\phi}[rad]$	Planar gripper angle in $\mathbf{I}$
$\tilde{\Phi}[rad]$	Set of planar gripper angles in each pixel of $\tilde{\mathbf{G}}$
$\tilde{\Phi}_\theta[rad]$	Approximated values of $\tilde{\phi}$ for each pixel of $\mathbf{I}$
$\tilde{\Phi}_f[rad]$	Approximated values of $\tilde{\phi}$ for each pixel of $\mathbf{I}_f$
$\tilde{\mathbf{W}}[m]$	Set of planar gripper width in each pixel of $\tilde{\mathbf{G}}$
$\tilde{\mathbf{W}}_f[m]$	Approximated values of $\tilde{\omega}$ for each pixel of $\mathbf{I}_f$
$\tilde{\mathbf{W}}_\theta[m]$	Approximated values of $\tilde{\omega}$ for each pixel of $\mathbf{I}$
$\tilde{\mathbf{Q}}$	Set of planar grasp quality in each pixel of $\tilde{\mathbf{G}}$
$\tilde{\mathbf{Q}}_f$	Approximated values of $\tilde{\omega}$ for each pixel of $\mathbf{I}_f$
$\tilde{\mathbf{Q}}_\theta$	Approximated values of q for each pixel of $\mathbf{I}$

g	Grasp representation in $\mathcal{C}$
$\tilde{g}$	Grasp representation in $\mathcal{I}$
g_{θ}^*	Best grasp visible in $\mathcal{C}$
$\tilde{g}_{\theta}^*$	Best visible grasp in $\mathcal{I}$
$\tilde{g}_f^*$	Best visible grasp in $\mathcal{I}_f$
$\tilde{\mathcal{G}}$	Set of grasps $\tilde{g}$ in $\mathcal{I}$
$\tilde{\mathcal{G}}_{\theta}$	Set of estimated grasps $\tilde{\mathcal{G}}$ in $\mathcal{I}$
$\tilde{\mathcal{G}}_f$	Set of estimated grasps $\tilde{\mathcal{G}}$ in $\mathcal{I}_f$
$\tilde{\mathcal{G}}_{Tr}$	Ground truth grasp rectangles
$\tilde{\mathcal{G}}_g$	Set of grasps generated by GraspNet in 6D
$\mathcal{G}_{gf}$	Filtered grasps referenced on the camera coordinate frame
$\mathcal{G}_o$	Grasp pose that does not collide with the printer point cloud
$\mathcal{G}_{fb}$	Selected grasp on the robot's base coordinate frame
$\mathcal{G}_a$	Current gripper pose
$M(\cdot)$	Grasp function that generates a set of grasps $\tilde{\mathcal{G}}$ from an image of modality $(\cdot)$
$M_{\theta}(\cdot)$	approximation of the grasp function $M(\cdot)$, where θ indicates the network weights
$\mathcal{M}_r$	Segmentation mask generated by Mask-RCNN
$\mathcal{N}_r$	Point cloud of the detected object
$\mathcal{N}_f$	Filtered point cloud of the detected object
$\mathcal{K}_r$	Raw point cloud of the printer
$\mathcal{K}_d$	Downsampled point cloud of the printer
$\mathcal{T}_{RG}$	Homogeneous transformation between the robot frame and the gripper frame
$\mathcal{T}_{GC}$	Homogeneous transformation between the gripper frame and the camera frame
$\mathcal{T}_{CI}$	Homogeneous transformation between the 2D image coordinates and the 3D camera frame

CONTENTS

Acknowledgments	v
Resumo	vii
Abstract	ix
List of Publications	xi
List of Figures	xvi
List of Tables	xvii
List of Symbols	xxi
Chapter 1—Introduction	1
1.1 Motivation	1
1.2 Problem definition	5
1.3 Objective	7
1.4 Contributions	7
1.5 Structure of this Thesis	8
Chapter 2—Related work	9
2.1 Grasp definition	11
2.1.1 Preliminary Considerations	14
2.2 Grasping datasets and related processings	15
2.2.1 Robotic Grasping Datasets	15
2.2.2 Processing on labelled dataset	19
2.2.3 Training data generation	19
2.2.4 Preliminary Considerations	23
2.3 Network architecture and training	25
2.3.1 Training process	25
2.3.2 Evaluation metrics	28
2.3.3 Preliminary Considerations	29
2.4 Grasping pre-processing and post-processing	30
2.4.1 Pre-processing	31
2.4.2 Post-processing	31
2.4.3 Preliminary Considerations	32
2.5 Test objects	32
2.5.1 Preliminary Considerations	33

2.6	Evaluation procedures, benchmarks, and results	34
2.6.1	Grasping benchmarks	35
2.6.1.1	Benchmark protocols	36
2.6.2	Evaluation protocols adopted in simulated experiments	44
2.6.3	Evaluation protocols adopted in real experiments	44
2.6.4	Definition of grasping success	45
2.6.5	Grasping performance	46
2.6.6	Preliminary Considerations	48
2.7	Conclusion	51

Chapter 3—Selective Planar Robotic Grasping Using Convolutional Neural Networks 53

3.1	Introduction	53
3.2	Contributions	55
3.3	Problem Statement	55
3.3.1	Assumptions	55
3.3.2	Definitions	55
3.4	Generative Grasping CNN	57
3.4.1	Training dataset and related processes	57
3.4.2	Network architecture	58
3.4.3	Grasp definition	59
3.5	Very Deep Convolutional Networks for Large-scale Image Recognition (VGG)	60
3.6	Deep Residual Learning For Image Recognition	61
3.7	Single Shot Multibox Detector	63
3.7.1	Test objects	66
3.7.2	SSD Base Network, fine-tuning and dataset	66
3.8	Selective Grasping	67
3.8.1	Grasping pipeline	67
3.9	Hardware and software implementation	68
3.9.1	UR5 Robot	68
3.9.2	Robotiq Gripper 2F-140	69
3.9.3	Intel Realsense D435	69
3.9.4	Robotic Operating System Implementation	70
3.10	Pre-processing and Post-processing	72
3.10.1	Pre-processing	72
3.10.2	Post-processing	72
3.11	Experiments	72
3.11.1	Hardware and software	72
3.11.2	Performance evaluation of the object's recognition system	73
3.11.3	Performance evaluation of the robotic grasping system	74
3.11.4	Failure mode	76
3.12	Conclusion	77

Chapter 4—Selective 6D Robotic Grasping Using Convolutional Neural Networks 79

4.1	Introduction	79
4.2	Contributions	80
4.3	Problem definition	81
4.4	Object Segmentation	82
4.4.1	Mask R-CNN	82
4.4.2	Dataset and Training	82

4.5	GraspNet	83
4.6	Collision Check	85
4.7	Grasping Pipeline	87
4.8	Experimental Setup	89
4.9	Experimental Results and ablation study	90
4.10	Comparative Study	92
	4.10.1 Real hardware implementation	96
	4.10.2 Failure mode	98
4.11	Conclusion	102
Chapter 5—Conclusion		103
5.1	Main Contributions	103
5.2	Future Works	104
Bibliography		114

Chapter 1

INTRODUCTION

1.1 MOTIVATION

In recent years, significant advances have been made in the field of autonomous mobile and manipulator robots. The interdisciplinary nature of robotics has played a pivotal role in driving substantial advancements. Artificial Intelligence and computer vision have emerged as influential contributors, significantly enhancing the outcomes of contemporary research, surpassing conventional analytical and empirical methods (MORRISON; CORKE; LEITNER, 2018).

Robot manipulators that can autonomously manipulate objects of different geometries in different environments have a wide range of applications such as medicine, manufacturing, retail, service robotics, emergency support, serving food, among others. However, there are still many issues to be solved until they can safely be applied to perform these activities, including but not limited to the complexity in performing grasping in unknown objects with adversarial geometry, and the collision with the robot workspace (KROEMER, 2015; MAHLER et al., 2019; MOUSAVIAN; EPPNER; FOX, 2019).

Robotic grasping is defined by the position and orientation of the gripper so that an object can be grasped, meeting several relevant criteria, such as object shape, position, material properties, and mass, given an RGB-D image, Depth image, Point Cloud or Voxel Grid as a reference. Grasp is also defined as the process of controlling an object's motion in a desired way by applying force and torques at a set of contacts (NEWBURY et al., 2023). Robotic grasping is one of the fundamental skills in manipulating an object and is still an open area of research (MORRISON; CORKE; LEITNER, 2020b; RIBEIRO; GRASSI, 2019). When applying a robotic grasping technique, it is necessary to get an accurate definition of what is a successful grasp. This definition varies according to the technique used. Besides that, there may be several successful grasps poses in different object regions. Therefore, it is crucial to select the positive grasp that represents the greatest success rate (LENZ; LEE; SAXENA, 2015; REDMON; ANGELOVA, 2015).

Robotic grasping involves several areas of robotics, such as perception, motion planning, and trajectory execution. Perception is based on sensors to predict the pose that the gripper needs to reach in 3D space, motion planning refers to the process planning a path that satisfies the constraints of motion dynamic and obstacle avoidance and trajectory execution uses a controller to execute the planned path (SUN et al., 2023). Consequently, its implementation in practice is a challenge. This challenge becomes even greater when the robot performs grasping on objects of different geometries with an unlimited amount of positions, since it requires a high dexterity (KUMRA; KANAN, 2017; TOBIN et al., 2017). Besides that, grasping has shifted from considering relatively simple, isolated objects to grasping geometrically and visually challenging objects in a cluttered environment (MORRISON; CORKE; LEITNER, 2019).

Detailed instructions can be provided to a robot to perform specific tasks. This method is known as an analytical or geometric method. Analytical robotic grasping methods are referred to as hand-designing features and have been widely used in the past (MAITIN-SHEPARD et al., 2010; KRAGIC; CHRISTENSEN, 2003). These methods require the development of a mathematical grasping model that includes the geometry, kinematics, and dynamics related to the robot and the object, which is not previously known (KOBBER; PETERS, 2010). In addition, the surface properties and friction coefficients are not available in priori (BOHG et al., 2013). Therefore, these parameters cannot be accounted for unknown objects.

Analytical methods also consider that the position of the object and the contact locations between the end effector and object are entirely known (PRATTICCHIZZO; TRINKLE, 2008). In some analytical methods, grasping poses are previously calculated using a point cloud registration method, which matches the 3D mesh point cloud and the real object models through geometric similarities (CIOCARLIE et al., 2014; HERNANDEZ et al., 2016). Despite the satisfactory performance in known environments, analytical methods are not feasible in unknown, dynamic and unstructured environments, considering unknown objects.

Rather than analytical methods, empirical methods have the advantage of removing partially or entirely the need for a complete analytical model (KONIDARIS et al., 2012). Such techniques focus on using experience-based strategies employing machine learning. These approaches generate grasp candidates through trial and error and classify them by using some metric. Empirical strategies usually imply the existence of previous experience of grasping, provided with the aid of heuristics and learning (BOHG et al., 2013). This training process requires the use of real robots (LEVINE et al., 2016), simulations (KAPPLER; BOHG; SCHAAL, 2015) or direct assessment in images (LENZ; LEE; SAXENA, 2015).

Pinto and Gupta (2016) acquire training data through experience, making over 50.000 grasping attempts in 700 hours of training. In contrast, recent grasping techniques have focused on associating human-labeled positive grasps with grasping regions in RGB+D images (LENZ; LEE; SAXENA, 2015) or just depth images (MORRISON; CORKE; LEITNER, 2018).

In unstructured environments, robots must continuously and independently analyze data in the workspace to support decision-making and possibly react to unwanted and unexpected events (KATZ; KENNEY; BROCK, 2008). Moreover, robotic grasping is inherently a problem of perception; therefore, the control algorithm must have a high computational efficiency (LENZ; LEE; SAXENA, 2015). Recent research concerned with robotic grasping has yielded convincing results due to the advancement of deep learning-based computer vision techniques (MORRISON; CORKE; LEITNER, 2018).

Deep-learning techniques have provided a considerable advance in robotic grasping applied to unknown objects. Through these techniques, it is possible to extract features from objects that correspond to a specific grasping pose or a set of grasping poses. The results achieved exceed the analytical and empirical models (JOHNS; LEUTENEGGER; DAVISON, 2016; LENZ; LEE; SAXENA, 2015; MAHLER et al., 2017; MORRISON; CORKE; LEITNER, 2018).

Several robotic grasping techniques, based on Convolutional Neural Network (CNN), generate multiple grasping poses (LENZ; LEE; SAXENA, 2015; MAHLER et al., 2017; PINTO; GUPTA, 2016; WANG et al., 2016; MOUSAVIAN; EPPNER; FOX, 2019). These techniques determine the best grasp among the generated grasps based on its score or kinematic viability. In other words, the final grasp can be determined by the highest score inferred by the network or the one in which the inverse kinematics, given the gripper pose, is feasible.

Deep learning-based methods are mainly divided into two categories such as sampling-based methods and regression-based methods (NEWBURY et al., 2023). Sampling methods individually evaluate the grasp using the encoded information. In these methods, the grasp is sampled and then evaluated using a quality estimation function, which is usually a CNN. The quality term is usually a scalar value that represents the probability of success of the grasp. The uniform and Gaussian distributions are commonly applied to sample grasps. Johns, Leutenegger and Davison (2016) were among the pioneers in discovering that the application of Gaussian distributions enhances grasping performance by effectively eliminating outliers. To generate a set of grasps poses, different sampling methods may be applied, such as Euclidean space, priors, configuration space, latent space, and multiple views. The Euclidean space requires post-process to refine the grasps

and remove infeasible grasps while latent space generates more robust grasps but still needs post-process methods to remove grasps in collision with other objects and refine the grasps positions (MOUSAVIAN; EPPNER; FOX, 2019).

Johns, Leutenegger and Davison (2016) used a CNN to detect robotic grasps on static uncluttered objects. Rather than just considering an individual grasp pose, this method finds a score for each grasp in a discretized grasp map and selects the best grasp according to uncertainties related to the grasp pose. In this research, the CNN was trained in a simulated environment using rendered depth images of synthetic objects. The gripper approaches the object orthogonally to the plane composed of the RGB+D sensor (a method known as planar grasping).

Mahler et al. (2017) used a synthetic dataset containing 3D object models to train a CNN denoted Grasp Quality Convolutional Neural Network (GQ-CNN). This CNN infers the grasp position directly from the depth image generated by an RGB+D sensor. The robot first separate the objects using a push policy when it is detected that there is no sufficient clearance to grasp an object.

Lenz, Lee and Saxena (2015) and Wang et al. (2016) applied an inference method to find an optimal grasp pose for an object which maximizes the chances of grasping it. The Cornell database (LENZ; LEE; SAXENA, 2015) was used in these researches. As opposed to previous methods, they treat robotic grasping as a problem of object detection. This method handles information in the context of feature learning, using data from RGB+D cameras. Despite the results, potential sampled robotic grasps can be ignored. Although the algorithm was tested in cluttered environments, the objects used in the experiments were distant from each other, which may facilitate the grasp.

Gualtieri et al. (2016) developed a 6D robotic grasping technique for unknown objects. This generates the position and orientation of the gripper when grasping an object. The method used in this paper applies an open-loop controller and only works when the camera is at a certain distance from the object (to see the whole object). Open-loop controller is commonly referred in robotic grasping as a method that does not use visual feedback to perform the grasping. In other words, the grasping is generated from a certain position and orientation of the gripper and then executed without constant feedback. Despite the high success rate in grasping cluttered objects, it is necessary to scan the object space to be grasped beforehand; therefore, the grasping time is considerably increased.

Levine et al. (2016) were one of the first to implement closed-loop robotic grasping using deep-learning techniques on cluttered objects. However, the CNN used has many layers, one million parameters, and requires months of training in various robots. In addition, the camera is fixed, making it difficult to adapt to different scenarios such as table height. The frequency obtained ranged from 2 to 5 Hz. Viereck et al. (2017) overcomes this limitation by decreasing the number of layers of the CNNs and placing the camera on the gripper. The model used in this research learns the distance to the nearest feasible grasp on the object. Despite achieving a maximum processing frequency of 5 Hz, it was insufficient to attain a significant percentage of successful closed-loop grasps. The network parameters were trained using OpenRAVE simulation and the kinematics solver IKFast (DIANKOV; KUFFNER, 2008).

Morrison, Corke and Leitner (2018) have proven through real experiments that the number of CNN parameters directly affects the performance of the robotic grasp. This performance is essential for grasping in unstructured and dynamic environments. Morrison, Corke and Leitner (2018) solved the computational performance problem found in Viereck et al. (2017) by generating grasping poses for all pixels in an image through a relatively small 6-layer CNN and 62.000 network parameters. The maximum processing frequency reached was 50 Hz when generating new positions, enough for the closed-loop grasping method to be applied. In this method, the robotic grasping is defined by $g = (p, \phi, \omega, q)$, where p is the point on the object represented in the robot workspace, ϕ is the gripper orientation, ω is the grasp width, and q is a parameter that represents the

grasp quality. Therefore, it is possible to perform a complete robotic grasp, unlike the method presented by Viereck et al. (2017), which ignores the grasp width. Despite the performance of this CNN, the grasp is planar and only generates one grasp per object.

When some objects are positioned in a cluttered way, it may occlude the view of other objects to be grasped by the robot. To solve this problem, Pas et al. (2017) has created a point cloud inference method based on CNN. In this method, a trajectory planning algorithm proposed by Schulman et al. (2014) is used to generate a fixed trajectory during the grasping process. This trajectory allows the RGB+D camera to generate depth data from occluded parts of objects at different angles. This depth data is joined together to create a better representation of the object. Results showed a 9% increase in success rate compared to the use of more than one static sensor in the grasping process.

Morrison, Corke and Leitner (2019) develop a method, called Multi-View Picking (MVP), based on Pas et al. (2017). However, when building a point cloud of a set of objects, the robot follows a predefined trajectory to scan the entire object environment. When performing a scan, the sensor gets a full point cloud of objects, eliminating occlusions. Results showed a 12% increase in success rate over single fixed viewpoint methods. In addition, the proper exploration of the object area has been shown efficient enough to increase the chances of successful grasping. Despite the improved performance, the time consumed to pick an object is increased significantly. Morrison, Corke and Leitner (2019) used objects of adversarial geometries proposed by Mahler et al. (2017).

Ribeiro and Grassi (2019) presented a CNN architecture to predict planar grasp poses using an RGB image. The grasps are based on a rectangular representation presented in Jiang, Moseson and Saxena (2011). The rectangle representation is composed of five parameters corresponding to the rectangle center (x, y) , the gripper width w , the gripper plate size h , and the gripper angle θ . The Cornell database (LENZ; LEE; SAXENA, 2015) was augmented to improve the generalization of the network architecture used. 297338 instances were generated from only 885 images in which 82% were used for training, 1% for validation, and 17% for testing. They developed a network with 4 convolutional layers and 2 fully connected layers, containing 1.548.569 parameters. The evaluation was performed using the Intersection over Union (IoU) or Jaccard Index bigger than 25%. Ribeiro and Grassi (2019) achieved a performance of 74.07 FPS (13.5ms per inference) using the GeForce GTX 1050 Ti, an accuracy of 94.8% in the Image-Wise division, and 86.9% in the Object-Wise division.

Ribeiro, Mendes and Grassi (2021) developed a CNN capable of performing dynamic grasps using the Cornell database (LENZ; LEE; SAXENA, 2015). The grasping is performed by applying a visual servo controller to adapt to changes in the object pose. The controller is learned from scratch using a CNN capable of end-to-end visual servoing. The image of the target is obtained *a priori*, and it is used as a setpoint by the visual servoing control. The control loop is executed as long as the L1-norm of the control signal is greater than a certain threshold. The evaluation was performed using the Intersection over Union (IoU) or Jaccard Index bigger than 25%. A success rate of 85.7% was achieved in a dynamic environment without clutter.

Comparing grasping algorithms is a complex task since the methods differ in terms of software, hardware, and task (pick-and-place, shelf picking, stow, etc.) (MAHLER et al., 2018). Some authors tried to standardize the benchmark procedure by proposing an environment and objects for the testing phase (BEKIROGLU et al., 2019; LIU et al., 2021; CALLI et al., 2015; MNYUSIWALLA et al., 2020; MORGAN et al., 2019). Nevertheless, the procedure itself is not enough to compare different grasping algorithms. The evaluation metric is also important. In this thesis, metric means the evaluation methodology to define the grasping effectiveness, considering different parameters as time and probability of success and failure. There are commonly used metrics in the industry to measure the performance of grasping algorithms such as Mean Picks Per Hour (MPPH), Successful task Executions over Total Attempts (SETA), Mean Time Between Failure

(MTBF), Mean Time To Repair (MTTR) and Availability (MAHLER et al., 2018). These metrics allow for decoupling the algorithm performance and identifying the bottleneck of the system pipeline.

In the context of additive manufacturing systems, it is necessary to apply a robust robotic grasping technique capable of yielding a diverse set of 6D grasps (COSTA et al., 2020; ARRAIS et al., 2019). This is necessary as some grasps may not be kinematically possible or collides with objects in the robot’s volumetric space. Deep learning-based grasps techniques provided a great tool to improve the performance of grasps in unknown objects. However, grasps are usually performed in environment that offer a low risk of collision with objects. Techniques to avoid collisions between the robot’s gripper and the environment are still an open area of research.

Weng et al. (2023) proposed a neural network called Neural Grasp Distance Fields (NGDF) to predict the distance between a query pose and close grasps, representing the manifold of grasps as a continuous level set to avoid collisions with the obstacles in the environment. In consequence, the proposed method can plan the grasp and the motion during the optimization problem without having to decouple the grasp synthesis, motion planning, and collision avoidance. However, a dataset is needed for each gripper model. Therefore, a neural network must be trained for every new gripper type.

1.2 PROBLEM DEFINITION

Recent grasping techniques are commonly focused in generating grasps and do not allow the selection of a specific object to be grasped (MORRISON; CORKE; LEITNER, 2018; MOUSAVIAN; EPPNER; FOX, 2019). Instead, they generate random grasps based on a quality prediction of an input data considering all the objects in the workspace. Therefore, given a set of objects in the workspace, the grasp generator may grasp any of them. Nevertheless, it is not desired since the robot may grasp part of a fixed object such as a bin instead of the movable objects. Furthermore, it is not possible to select a specific object to be grasped using only these grasping methods.

This thesis employs two distinct environments, a tabletop and a constrained space, to assess the efficacy of the grasp generation methodologies. The former, represented a planar surface populated with multiple objects (Figure 1.1). Conversely, the constrained space is exemplified by an Additive Manufacturing Unit, specifically a 3D printer. This setting presents increased complexity owing to potential collisions between 3D printer parts, proximate objects and the gripper (Figure 1.2).

This thesis addresses robotic grasping in static unstructured environments with unknown objects arbitrarily positioned. To clarify, attributes such as size, shape, weight, position, orientation, static and dynamic friction, among other physical parameters, are not predetermined in this context. In this thesis, the objects chosen for experimentation were first proposed by Mahler et al. (2017). These objects are recommended for benchmarking owing to their geometrically challenging features, such as smooth curved surfaces and narrow openings.

Employing complex objects allows for a more comprehensive assessment of the proposed system in realistic settings, in contrast to relying on basic shapes such as cubes or cylinders. Traditional robotic manipulators in industrial applications predominantly deal with standardized items. However, the rise of warehouse automation and domestic robotics necessitates advanced capabilities in grasping a broader array of objects. Highlighting the significance of this, major corporations endorse the advancement of robotic grasping systems in unstructured environments through prominent competitions, including the Amazon Picking Challenge (CORRELL et al., 2016) and the IROS Robotic Grasping and Manipulation Competition (SUN et al., 2016). These challenges rigorously evaluate the adeptness of robotic systems in intricate environments, including both unknown environments and objects.

4D GRASPING



Figure 1.1: Tabletop scenario for 4D grasping testing.

6D GRASPING



Figure 1.2: Additive Manufacturing Unit for 6D grasping testing.

Recent studies (NEWBURY et al., 2023; MORRISON; CORKE; LEITNER, 2019) frequently evaluate grasping techniques within planar workspaces such as tables or bins. Such environments are often selected for evaluating grasping efficacy due to their relative simplicity compared to more constrained scenarios. It's important to note that methods that generate 4D grasps, also known as planar grasps, work well on flat surfaces but might not be ideal for constrained spaces such as inside the 3D printer. This limitation comes from the fact that the gripper can only grasp an object orthogonal to the surface. In other words, the gripper can only grasp an object from the top. In constrained spaces, a 6D

grasp generator is recommended since the robot can grasp the object from all directions.

1.3 OBJECTIVE

This thesis delineates the development of 4D and 6D selective grasping pipelines using Deep Learning techniques, implemented in tabletop and constrained environments. The grasps target objects of unknown shape and physical attributes. To discard the grasps in collision with the environment, a novel collision avoidance algorithm utilizing point clouds and signed distance fields has been introduced. Moreover, a heuristic-based approach was proposed to filter and prioritize grasps based on the gripper's position and orientation, and the grasp score inferred by the 6D grasping generation network. The research also compares the proposed 4D and 6D robotic grasping pipelines considering the same environment and objects. An automatic object detection dataset generation pipeline was also developed to facilitate the training of the object recognition and segmentation networks.

A thorough assessment, backed by experimental data, elucidates the grasping performance. Validations for both conditions were conducted using the Intel Realsense D435 RGB+D camera, an UR5 Robot Arm Manipulator, and a Robotiq 2F-140 gripper.

1.4 CONTRIBUTIONS

This thesis has the following main contributions:

- Development of a 4D grasping pipeline using a Convolutional Neural Network Grasp Generator and an Object Detection Network to selectively grasp objects on a tabletop scenario using an RGB+D sensor (Figure 1.3);
- Development of a selective grasping pipeline based on Variational Autoencoders and an Object Segmentation Network to generate 6D grasps using an RGB+D sensor, avoiding collisions between the robot's gripper and the environment (Figure 1.4);
- Development and validation of a low computational complexity collision avoidance system to discard grasps in collision with the environment and a heuristic method to filter the best grasps;
- Integration of an object recognition and instance segmentation method, a 6D grasping generator, and a new collision detection system;
- Development of an automatic object recognition dataset generation pipeline using a simulator; and
- Validation of the proposed system using an UR5 Robot Arm Manipulator, an RGB+D camera Intel Realsense D435, and the gripper Robotiq 2F-140.

Additionally, an ablation study is conducted on the grasping pipeline stages to analyze how the size of the workspace point cloud affects collision check time. The object detection and segmentation algorithm using deep learning for specific manufactured objects is validated in real experiments, proving the effectiveness of a simulator-generated synthetic dataset. A detailed comparison between 4D and 6D grasping techniques is also presented, highlighting the advantages and disadvantages of each method, using the same set of objects.

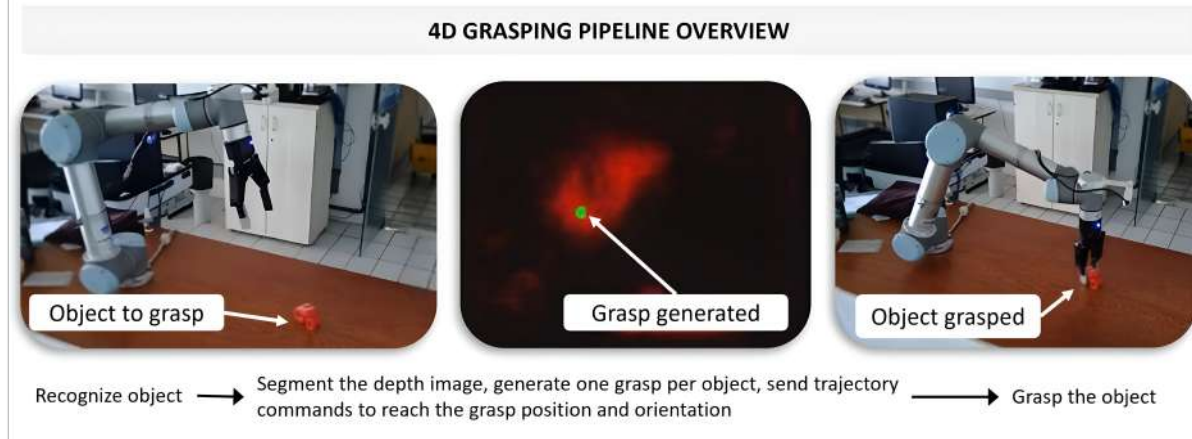


Figure 1.3: Overview of the proposed 4D grasping pipeline (Chapter 3).

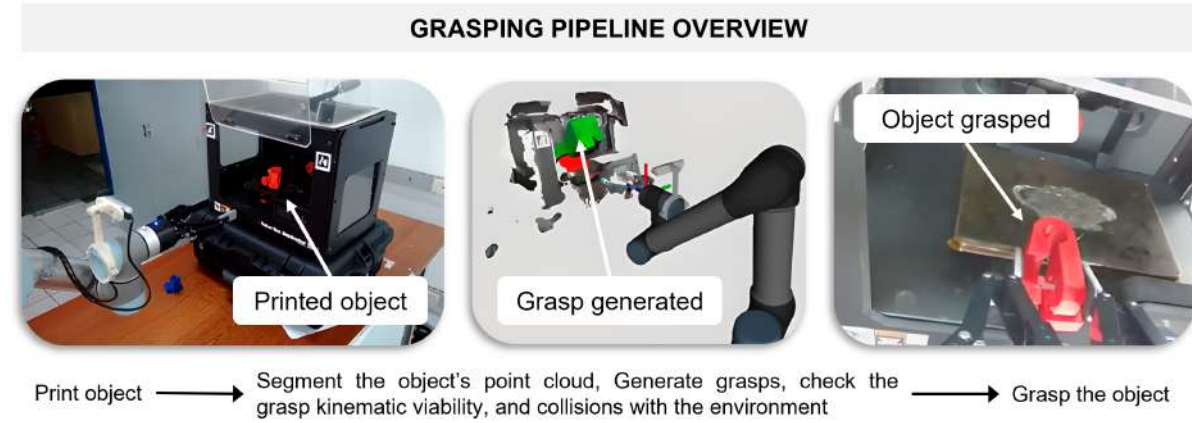


Figure 1.4: Overview of the proposed 6D grasping pipeline (Chapter 4).

1.5 STRUCTURE OF THIS THESIS

The remainder of this thesis is organized as follows:

Chapter 2: A detailed overview of recent grasp techniques is provided;

Chapter 3: A grasping pipeline capable of performing grasps on unknown and uncluttered objects is presented;

Chapter 4: A multiple 6D grasp generator is integrated with an object recognition system to perform selective grasps. A collision check system is proposed to verify collisions with nearby obstacles in an Additive Manufacturing Unit. An automated dataset creation pipeline using simulation is presented;

Chapter 5: Finally, the conclusion, main contributions and future works are presented.

Chapter 2

The purpose of this chapter is to provide a review of robotic grasping techniques using deep learning. This review is subdivided into sections regarding important aspects of robotic grasping. This review serves as a tool to understand the impact of each procedure adopted by several authors.

RELATED WORK

As Morrison, Corke and Leitner (2018) state, it is not proper to directly compare results between robotic grasping experiments due to the extensive grasp detection techniques used. Each experiment differs from the other by using different types and numbers of objects, physical hardware, robot arms, grippers, and cameras. Mahler et al. (2018) is a guest editorial composed by reputed researchers and industry professionals in robotic grasping in the United States, Germany, and Australia. The editorial states that although there are openly shared codes and data, it is highly complex to compare or replicate experimental results to identify which aspects of each approach work best. Although Mahler et al. (2018) propose standardization of data to be given by the authors, they do not offer a principled benchmark methodology that clearly defines each step in the grasping pipeline as Bekiroglu et al. (2019) do.

Direct comparisons are not fair due to the disparities in the adopted hardware, software, experimental procedures, and the environment (NEWBURY et al., 2023; MAHLER et al., 2018). Besides that, the vision system is also confounded by lighting variation, shadows, reflections, surroundings, and other factors. Due to the lack of documentation in the evaluation regarding grasping methods, Mahler et al. (2018) proposes a procedural model to simplify and increase the effectiveness of grasping’ methods comparison. It was proposed the use of a metric¹ called Mean Picks Per Hour (MPPH), commonly used in industry to test the speed where robots can grasp objects. Nevertheless, this metric can be affected by the trajectory planning and the robot motion, leading to the wrong interpretation of the grasping precision. Mahler et al. (2019) prove that a grasping method could achieve a high MPPH and low precision. Consequently, MPPH metric does not evaluate exclusively the grasping method itself.

Therefore, as Bekiroglu et al. (2019) state, it is difficult to evaluate the influence of a grasp planning algorithm independently of the vision system, robot arm, and hand. Different authors tried to establish protocols to standardize the grasping performance evaluation, including what type of objects should be used, how they should be placed in the robot workspace and metrics calculation (BEKIROGLU et al., 2019). It is an agreement between recent works that there are no tools to build meaningful comparisons, preventing systematic analysis (MORGAN et al., 2019). Also, it is difficult to evaluate each system’s component such as object recognition, segmentation, motion planning, and hardware design (MORGAN et al., 2019; LIU et al., 2021).

Although Bekiroglu et al. (2019) presented a step-by-step solution for evaluating the grasping performance, they do not provide a metric to analyze the type of object in terms of difficulty to grasp as shown in Mahler et al. (2019). The transparency, porosity, and deformability of the objects highly affect the grasping performance, and these should be accounted for in the evaluation (MAHLER et al., 2019). Besides that, for the learning-based grasping methods, the objects must be divided into the novel and seen objects. Novel objects mean objects that were not used to train the grasping algorithm. Although

¹It is important to note that in the robotic grasping field, metric means the evaluation methodology to define the grasping effectiveness.

Mahler et al. (2019) divided the objects in terms of grasping difficulty, this parameter was set based on the reliability across the policy adopted by the authors. It does not represent the actual grasping difficulty because different grasping methods would result in different reliabilities. Therefore, this metric is not directly comparable.

The level of clutter also influences grasping performance. Mnyusiwalla et al. (2020) evaluated grasping methods using the same test environment and objects in different clutter levels, documented through images. They found that the higher the clutter the lower is the grasping performance.

There are famous image recognition challenges in computer vision such as COCO (LIN et al., 2015) and VOC (EVERINGHAM et al., 2010a) which contributed to the enormous advancements in the field. But there are no similar challenges for robot grasping because it is significantly more challenging due to the mechanical, sensorial, and algorithmic innovations combined in any solution (BEKIROGLU et al., 2019). Liu et al. (2021) created a robot setup, called OCRTOC (Cloud-Based Competition and Benchmark for Robotic Grasping and Manipulation), to perform remote experiments of standardized table organization scenarios in varying difficulties to provide a fair environment grasping performance comparison.

System-level benchmarks also depend on the application itself. There are benchmarks for pick-and-place in tabletop scenarios (CALLI et al., 2015), pick-from-the-self scenario (LEITNER et al., 2017), bin-picking (MNYUSIWALLA et al., 2020). There are extensive grasping applications, and it is impossible to derive a benchmark that covers all of them (MNYUSIWALLA et al., 2020).

The type of the end-effector is also an important factor that affects the grasping performance. Mnyusiwalla et al. (2020) evaluated four types of end-effectors on the same set of objects. They found that the grasping performance highly depends on the gripper type. The challenges behind the hardware design are also reported in Sun et al. (2021).

In addition to the challenging comparison between the grasping methods, the grasping performance can be strongly affected by the network architecture and its parameters (MORRISON; CORKE; LEITNER, 2018). The grasping methods applied in each research vary considerably by employing different experimental procedures, datasets, network architectures, metrics, pre-processings, post-processings, etc. Therefore, it is crucial to review the motivation underlying each pipeline step. It is noted poor documentation in some research, which can lead to even more challenging comparisons.

Figure 2.1 shows a flowchart that describes the commonly adopted processes in several grasping methods as well as the sections in which each process is presented in this chapter:

- Section 2.1 shows the specificities related to the grasp definition;
- Section 2.2 describes the dataset generation method and its related pre-processes and post-processes. Some grasping datasets are generated from simulation if they are not manually labeled beforehand;
- Section 2.3 details several network architectures, training procedures, cost functions, weight initialization, and more network-related information;
- Section 2.4 presents the pre-processing and post-processing commonly applied to improve the grasping performance;
- Section 2.5 shows the objects often used for evaluating the grasping performance in real experiments. Robotic grasping methods are not tested in the same way as object recognition networks. It requires real or simulated tests with complete hardware including the robot, sensors, objects, and the controller;

- Section 2.6 indicates the performance results of each related work and the procedures and benchmarks employed for the grasping performance evaluation; and
- Section 2.7 concludes this chapter.

Each section includes preliminary considerations, which summarize important observations.

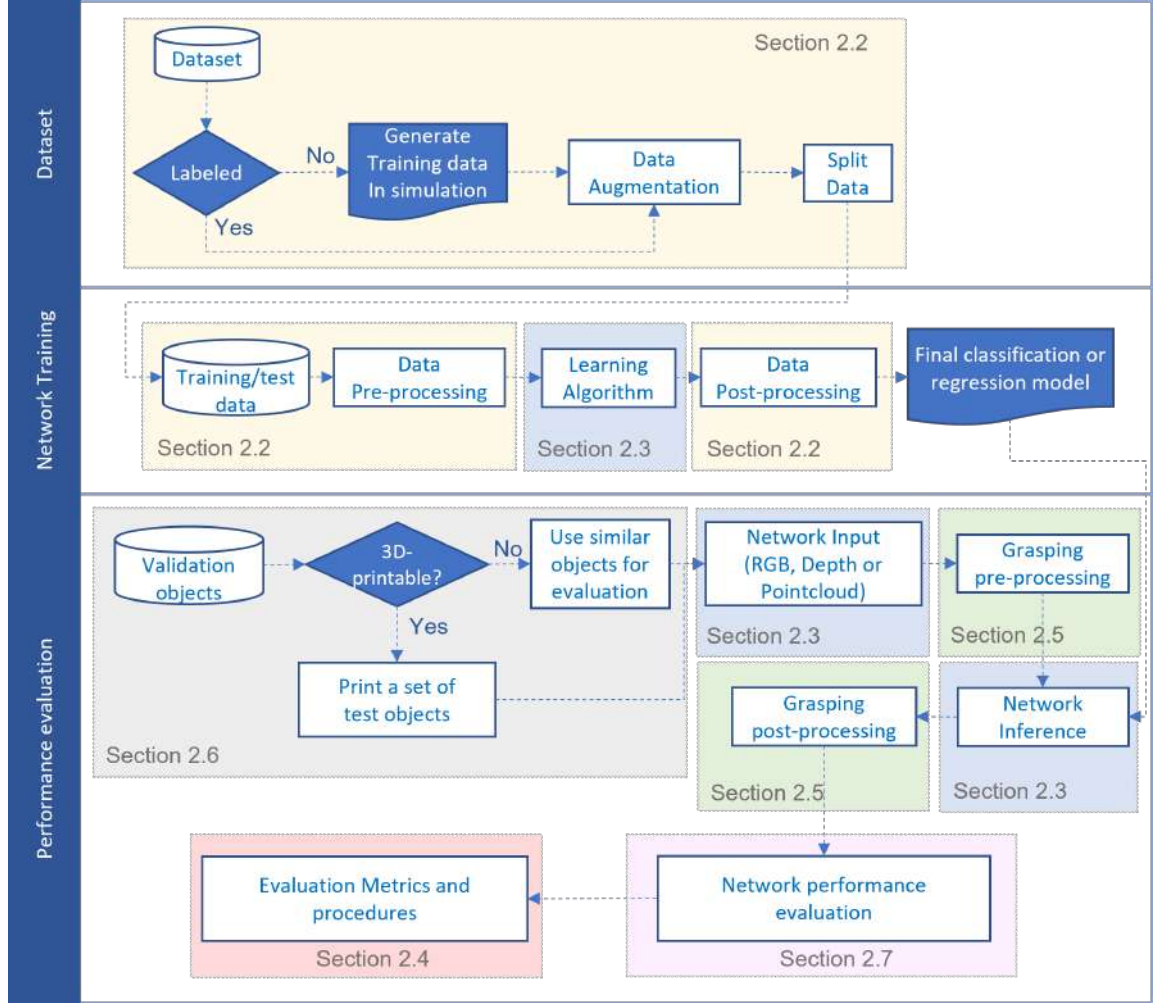


Figure 2.1: Flowchart that describes the commonly adopted processes in several grasping methods as well as the subsection in which each process is presented.

2.1 GRASP DEFINITION

There are several ways to define a robotic grasp, and it highly depends on the method used. Johns, Leutenegger and Davison (2016) define a grasp as a pose, comprising of three parameters (x, y, θ) , where x and y are the positions related to the gripper's grasping axis in the image coordinate relative to the center of the image and θ is the gripper orientation. This grasp definition is commonly applied when the method refers to planar grasping. The gripper pose is obtained in the robot's base coordinate using a sequence of homogeneous transforms,

$$g = T_{RG}(T_{GC}(T_{CI}(\tilde{g}))), \quad (2.1)$$

where g is the gripper pose in the robot base frame, $\tilde{g}$ is the gripper pose in the image coordinate frame, T_{RG} is the homogeneous transformation between the robot frame and the gripper frame, T_{GC} is the transformation between the gripper frame and the camera frame, and T_{CI} is the transformation between the 2D image coordinates and the 3D camera frame.

Mahler et al. (2017) defined the grasping pose as a set of four parameters: (x, y, z) , and ϕ , the planar gripper angle. Jiang, Moseson and Saxena (2011) developed a grasp representation based on a rectangle corresponding to a pair of parallel edges of a robotic gripper. This grasp rectangle is composed of five parameters: x and y coordinates of the upper-left corner of the rectangle, gripper width ω corresponding to the antipodal grasp, gripper finger's tip width h and grasp angle ϕ . This representation is commonly adopted in related works (LENZ; LEE; SAXENA, 2015; RIBEIRO; GRASSI, 2019). Pinto and Gupta (2016) employed a similar representation but fixed the length of the grasp width and gripper finger's tip instead. Gualtieri et al. (2016) represented the grasp pose as a 6D pose, comprising the position and orientation in $\mathbb{R}^3$.

Definition 2.1.1. *The term antipodal grasp refers to the term antipode also used in plane geometry. Given a point O lying in the center of circle C , the antipode O' of O is the other point lying on the conic through which the line $\overrightarrow{OC}$ passes (Figure 2.2a). In robotic grasping, antipodal grasp is used to define the gripper angle Φ , the Tool Center Point (TCP), and the finger's plate position represented as antipodal points (Figure 2.2b).*

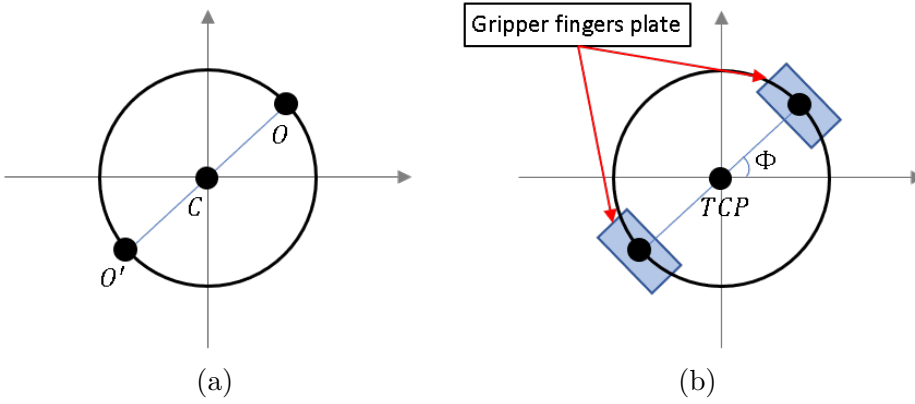


Figure 2.2: (a) Antipodal point of a point on a circle (b) antipodal grasp definition.

Levine et al. (2016) also defined a grasp by the position (x, y, z) , and orientation ϕ of the gripper. Although they have not used grasp representation such as Mahler et al. (2017), during the grasp process, the gripper remains planar to the surface where the objects are located.

Morrison, Corke and Leitner (2018) and Morrison, Corke and Leitner (2020b) define a grasp as six parameters: x, y, z coordinates of the grasp point, gripper angle ϕ , gripper width ω and q which is a pixel-wise quality measure of the grasp. The grasp is first obtained in image coordinates given a 2.5D depth image and is later transformed into the robot base coordinates by

$$g = T_{RC}(T_{CI}(\tilde{g})), \quad (2.2)$$

where T_{CI} transforms from the image coordinate to the camera frame and T_{RC} transforms from the camera frame to the robot's base frame.

2.5D depth image is defined as a simplified representation of 3D data in which every pixel (x, y) on the camera's image plane contains only the depth value, measuring its distance to the scene (CHONG; TEOH; ONG, 2016).

Some authors defined the grasp by $(R, T) \in SE(3)$ where $R \in SO(3)$ and $T \in \mathbb{R}^3$ are the rotation and translation of grasp g (MURALI et al., 2020; MOUSAVIAN; EPPNER; FOX, 2019; MAHLER et al., 2019, 2016). Grasps are defined in the object reference frame, whose origin is the center of mass of the point cloud observed. This method is known as 6D or non-planar grasp. Contrary to Morrison, Corke and Leitner (2018), it is not necessary to transform the grasp representation from the image coordinates to the camera frame, since the grasp is already obtained in the camera coordinate frame. Figure 2.3 summarizes the types of grasping commonly adopted.

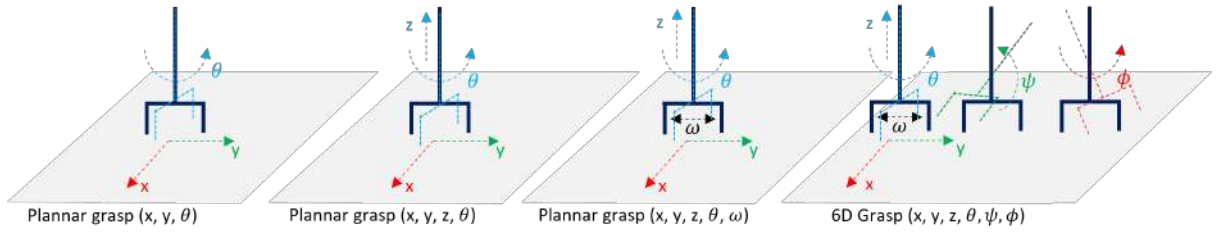


Figure 2.3: Types of grasp adopted by several authors. The grasping position is defined as (x, y, z) , and the orientation as θ, ψ, ϕ , corresponding to the roll (θ), pitch (ψ), and yaw (ϕ) angles.

Sundermeyer et al. (2021) state that the grasp representation is crucial to solving the grasping task using learning-based methods to generalize well to unseen objects and handle the high-dimensional output space well. The authors consider only the grasps that lie on surfaces that can be observed with a depth sensor. Given that the grasp point c in Figure 2.4 is predictable, it is possible to reduce the 6-DOF learning problem to estimating the 3-DOF grasp rotation $R_g \in \mathbb{R}^{3 \times 3}$ and grasp width $\omega \in \mathbb{R}$ of a parallel-yaw gripper.

In Sundermeyer et al. (2021), the grasp $g \in G$ is defined as the orientation and position $(R_g, t_g) \in SE(3)$ and grasp width $\omega \in \mathbb{R}$ as

$$t_g = c + \frac{\omega}{2}b + da, \quad (2.3)$$

and

$$R_g = \begin{bmatrix} | & | & | \\ b & a \times b & a \\ | & | & | \end{bmatrix}, \quad (2.4)$$

where $a \in \mathbb{R}^3$, $\|a\| = 1$ is the approach vector, $b \in \mathbb{R}^3$, $\|b\| = 1$ is the grasp baseline vector, and $d \in \mathbb{R}$ is the constant distance from the gripper baseline to the gripper base. The reduced dimensionality greatly facilitates the learning process compared to methods that estimate grasp poses in unconstrained $SE(3)$ spaces.

Breyer et al. (2021) defines a grasp g by the position t and orientation r of the gripper with respect to the robot's base coordinate system. The gripper opening ω is also considered. For each pose, a quality $q \in [0, 1]$ capturing the probability of grasp success is added.

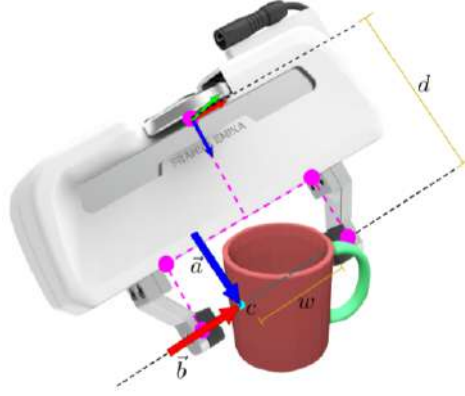


Figure 2.4: c depicts an observed contact point, b constitute the 3-DOF rotation, w is the predicted grasp width, d is the distance from baseline to base frame (SUNDERMEYER et al., 2021).

Jiang et al. (2021) define a 6-DoF grasp g as the grasp center position $t \in SO(3)$ of the gripper, and the opening width $\omega \in \mathbb{R}$ between the fingers. A scalar grasp quality $q \in [0, 1]$ estimates the probability of grasp success. For an arbitrary point $p \in \mathbb{R}^3$, the occupancy $b \in [0, 1]$, indicating whether a point is occupied by any other objects in the scene, is also considered.

Ni et al. (2020) denote grasps in 3D space as $g = (p, n, r, d)$ where $p \in \mathbb{R}^3$ is the grasp point, $n \in \mathbb{R}^3$ is the unit approaching vector, $r \in \mathbb{R}^3$ is the opening vector, and $d \in \mathbb{R}$ is the approaching distance of two fingers relative to grasp point p (Figure 2.5).

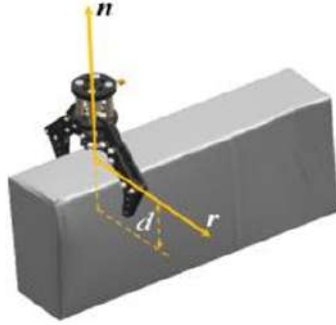


Figure 2.5: Grasp representation in 3D (NI et al., 2020).

2.1.1 Preliminary Considerations

Some approaches define the grasp position in $\mathbb{R}^2$ on the surface where the objects are located (RIBEIRO; GRASSI, 2019; JOHNS; LEUTENEGGER; DAVISON, 2016; LENZ; LEE; SAXENA, 2015; PINTO; GUPTA, 2016), whereas others define the grasp position in $\mathbb{R}^3$, considering the height of the object on the surface (MAHLER et al., 2017; MORRISON; CORKE; LEITNER, 2018, 2019, 2020b). Both aforementioned grasp representations consider a planar grasp; therefore, the orientation is only defined in one axis of rotation, orthogonal to the grasping rectangle.

The quality measure of the grasping pose can also be considered as part of the grasp definition (MORRISON; CORKE; LEITNER, 2018, 2020b). Although it benefits the acquisition of an acceptable grasp pose, it is not guaranteed to obtain a feasible or the best pose. Mousavian, Eppner and Fox (2019) do not use only the grasp score but also verify the kinematic feasibility of the generated grasp poses as well as if the grasp position leads to collision with other objects in the environment.

Furthermore, planar grasp techniques restrict its real applications. To perform bin picking, planar grasp techniques may be well suited. Nevertheless, for picking objects in constrained spaces, it is necessary to apply non-planar grasp techniques (MOUSAVIAN; EPPNER; FOX, 2019).

2.2 GRASPING DATASETS AND RELATED PROCESSINGS

This section focuses on the investigation of each dataset applied in robotic grasping methods as well as the data pre-processing preceding the network training. Subsection 2.2.1 briefly details each dataset applied in different robotic grasping researches. Subsection 2.2.2 describes the techniques employed to augment a labeled dataset and the procedures adopted to adequate the training data to fit the network architecture adopted by each related research. Subsection 2.2.3 indicates the techniques adopted to sample grasps from an existing not labeled dataset (such as 3D object’s mesh dataset) through offline simulations. Subsection 2.2.4 depicts important aspects of the related work in terms of dataset manipulation.

2.2.1 Robotic Grasping Datasets

KIT (KASPER; XUE; DILLMANN, 2012) is a dataset of 3D data acquired with a high-accuracy laser scanner model Minolta VI-900. The objects are rotated in the scanning processing in order to get different viewpoints of the object. The resolution of the depth images is 640×480 pixels. Color stereo images with 1392×1038 pixels are also captured from different angles using the camera Marlin 145C2 with known intrinsic parameters.

The 3DNet (WOHLKINGER et al., 2012) is a dataset that is divided into 4 other databases with 10, 60, 100, and 200 object classes, varying the difficulty in object recognition. It comprises 360 synthetic objects suitable for robotic manipulation, 1600 scenes of single objects on a planar surface in random poses, and several instances per class captured with an RGB+D sensor. For each scene, a color image, a point cloud, and a bounding box with the class label are provided. This dataset is generally used for object class recognition and 6D pose estimation.

BigBird dataset (SINGH et al., 2014) contains depth scans and RGB images of 125 real objects. 600 registered point clouds from a Primesense Carmine 1.09 sensor are provided. The objects are scanned by 600 different viewpoints. Transparent objects are not well scanned due to the depth measurement limitation. The segmentation masks for each of the images and point clouds are also provided.

Kappler, Bohg and Schaal (2015) presented a dataset with 700 object instances in 80 different categories, including drugstore products, groceries, music instruments, toys, tools and household instruments. Each object contains 500 grasps, totalizing 300k different grasps. This dataset also contains point clouds from different viewpoints. The mesh models used are from the 3DNet dataset (WOHLKINGER et al., 2012).

The Cornell Grasping Dataset (LENZ; LEE; SAXENA, 2015) has 885 depth images of real objects available and 5110 human-labeled positive and 2909 negative grasps. This dataset has the advantage of having several labeled grasps available per image. The Cornell Dataset represents antipodal grasps as rectangles using pixel coordinates, aligned to the position and rotation of a gripper as in Jiang, Moseson and Saxena (2011).

The ModelNet dataset (WU et al., 2015) is a large-scale collection of 3D mesh models

representing a range of common objects. This dataset is specifically applied for deep learning experimentation comprising 127,915 synthetic objects, 662 object categories, and 10 categories with annotated orientation.

The Yale-CMU-Berkeley (YCB) dataset (CALLI et al., 2017) contains 600 high-resolutions RGB images, 600 RGB+D images and five sets of textured three-dimensional geometric models. The real objects were scanned using BigBIRD and Google Scanners and are available to researchers upon request. In total, 77 objects were scanned and made available online. The objects sets were divided into 5 categories: Food items, kitchen items, tool items, shape items, task items.

The Jacquard Dataset (DEPIERRE; DELLANDRÉA; CHEN, 2018) contains RGB and depth images with antipodal grasp rectangles represented as pixels coordinates. These grasp rectangles are collected in simulation and do not require manual labeling. This dataset is usually applied in Simulated Grasping Trials (SGTs), where the successful grasps are obtained through simulations. This dataset contains 54,485 rendered images labeled with almost 5 million grasp annotations. It is also possible to access the simulated environment in which it was created, allowing the application of SGTs.

Mahler et al. (2017) presented a new dataset, denoted Dexterity Network 2.0 (Dex-Net 2.0). This dataset comprises more than 6.7 million point clouds of 1,500 synthetic objects. The images are not multiple-labeled such as the Cornell dataset. The grasp rectangles were associated with each image through simulations; therefore, several hours of tedious hand labeling were avoided.

Most datasets apply only to a specific type of gripper. To avoid this limitation, Mahler et al. (2019) developed a new dataset called Dexterity Network (Dex-Net) 4.0, extending the gripper-specific models of Dex-Net 2.0 (MAHLER et al., 2017) and 3.0 (MAHLER et al., 2018). This dataset is comprised of 5 million grasps computed over 1664 unique 3D objects in simulated clutter.

Morrison, Corke and Leitner (2020a) developed a dataset specifically for robotic grasping applications. It uses evolutionary algorithms to generate a set of objects that varies in terms of geometry complexity and graspability. This dataset contains 2000 3D object mesh, including a set of 49 printable objects that represent many semantic classes. In this dataset, objects are labeled according to their complexity and difficulty, providing a range in both dimensions. A set of 1 million precomputed grasp poses is provided. Each object is comprised of 100 antipodal grasps labeled with a robust grasp-quality metric (MAHLER et al., 2017). The code to create a custom object dataset from the Envolved Grasping Analysis Dataset (EGAD) is also provided.

Fang et al. (2020) presented a dataset called GraspNet. This dataset contains 97.280 RGB-D images of real world objects and over one billion 6D grasp poses manually annotated. The images were taken from the Intel Realsense D435 camera and Kinect V4 Azure. The grasp evaluation is done by analytical computation. The authors scanned 88 objects and generated 4 datasets with 190 scenes in total, called train (100 scenes), test seen (30 scenes), test similar (30 scenes) and test novel (30 scenes). Each scene contains 256 RGB-D images and millions of grasps for 10 objects randomly sampled.

Zhang et al. (2022) introduced the dataset REGRAD (Large-Scale Relational Grasp Dataset for Safe and Object-Specific Robotic Grasping in Clutter). The object is composed of 2D images and point clouds. The dataset contains 200 object instances, 55 categories and 50K different object models. The labels include 6D pose of each object, bounding boxes, 2D segmentations, point cloud segmentations, Manipulation Relationship Graph (MRG) indicating the grasp order, collision-free and stable 6D grasps of each object and rectangular 2D grasps. The dataset contains 118.1K different scenes.

Fang et al. (2022) presented a dataset called DexGraspNet with 1.32M grasps and 5355 objects for robotic hands. The dataset contains 133 object categories and more than 120 diverse grasps per object. The objects encompassed within the dataset are de-

rived from hand-scale categories, collected from both synthetic and real datasets such as YCB (CALLI et al., 2017), BigBIRD (SINGH et al., 2014), Grasp (KAPPLER; BOHG; SCHAAL, 2015), KIT (KASPER; XUE; DILLMANN, 2012), and Google’s Scanned Object Dataset (DOWNS et al., 2022). These objects are then processed—normalized, cleaned, remeshed into manifolds, and further assessed for collision meshes to ensure the dataset’s quality and utility. DexGraspNet’s generation leverages a highly efficient synthesis method anchored on a deeply accelerated differentiable force closure estimator, which robustly and efficiently synthesizes stable and diverse grasps on a large scale.

Table 2.1 shows a comparison between the datasets applied in robotic grasping techniques. It comprises important aspects of datasets such as the modality of the data provided and the type of supported grippers, divided by (1) vacuum gripper, (2) two-fingered gripper, (3) three-fingered gripper, (4) four-fingered gripper, and (5) five-fingered gripper. It also shows the labeling process and the number of ground-truths available as well as image resolutions when provided. Unfilled fields mean that the information was not provided, or it is not applicable for this dataset.

Table 2.1: Specification of the most used datasets in robotic grasping applications.

	Data Type	Type of Gripper	Annotated dataset	Grasp Type	Ground Truths	Multiple Labels	Multiple Views	Image Resolution	Number of Objects	Number of Images
REGRAD (ZHANG et al., 2023)	RGB/Point Cloud	2	✓	3D+	100M	✓	✓		200	900K
DexGraspNet (Fang et al., 2022)	CAD	5	✓	3D+	1.32M	✓	✓		5355	
EGAD (MORRISON et al., 2020a)	CAD	Adaptable	✓	3D+	1M	✓			2000	
Dex-Net 4.0 (MAHLER et al., 2019)	Depth	1, 2	✓	3D+	13K	✓	✓	96x96	1664	5M
Jacquard (DEPIERRE et al., 2018)	RGB+D	2	✓	3D, 4D	1.1M	✓	✓		11K	54K
Dex-Net 2.0 (MAHLER et al., 2017)	Depth	2	✓	3D+	6.7M		✓	640x480	1500	6.7M
YCB Dataset (CALLI et al., 2017)	CAD/RGB+D			3D+			✓	1920x1080	77	600
Wohlkinger et al. (2012)	CAD		✓	3D+	✓	✓	✓		700	
ModelNet (WU et al., 2015)	CAD			3D+					128k	
Cornell (LENZ et al., 2015)	RGB+D	2	✓	3D, 4D	8019	✓	✓	640x480	240	1035
BigBird (SINGH et al., 2014)	RGB+D			3D, 4D					125	600
3DNet (WOHLKINGER et al., 2012)	CAD			3D+					1500	
KIT (KASPER et al., 2012)	Depth	3, 4, 5	✓	3D, 4D			✓	1392x1038		
KIT (KASPER et al., 2012)	RGB	3, 4, 5	✓	3D, 4D			✓	640x480		

Type of gripper: 1 - vacuum gripper, 2 - two-fingered gripper, 3 - three-fingered gripper, 4 - four-fingered gripper, and 5 - five-fingered gripper.

Grasp Type: The type of grasping prevalent in most work utilizing the dataset. The grasping type 3D+ means that the dataset can be used to train from 3D to 6D grasps.

Multiple Labels: Multiple labels or grasps per object.

Multiple Views: Grasps annotated considering the same object from different views.

2.2.2 Processing on labelled dataset

Lenz, Lee and Saxena (2015) extracted an RGB+D image from each grasping rectangle of the Cornell Grasping Dataset and used this to generate features for each rectangle. This image is rotated to align the gripper orientation and re-scaled to 24×24 pixels to fit the network input. These images have 7 channels. The first three are the YUV color space representation, the fourth is the depth measurement and the last three are the position of the point cloud in $\mathbb{R}^3$. Each modality was regularized separately to match as closely as possible the statistics of each data to avoid biases. Lenz, Lee and Saxena (2015) found that distorting image features may cause non-graspable rectangles to be wrongly determined as graspable, decreasing the grasping performance.

Morrison, Corke and Leitner (2018) used the Cornell Dataset since there are multiple labeled grasps available per image. Besides that, the data better suits the pixel-wise grasp representation adopted. The Cornell Dataset was augmented by using random crops, zooms, and rotations to generate a set of 8840 depth images. This augmentation generated 51.100 grasp samples. It was noticed that the Cornell Dataset produces a more reasonable grasp estimation if compared to a unique image to represent a grasp as in Mahler et al. (2017). Only the positive labeled grasps were considered for training.

Morrison, Corke and Leitner (2018) cropped each Cornell Dataset depth image into a 300×300 image and generated three other 300×300 images from it. The images consist of the grasp quality, grasp angle, and grasp width. Each pixel in the grasp quality image is obtained by dividing each ground-truth grasp rectangle into three equal parts. The center of this rectangle is represented as the maximum grasp quality. Each pixel in the image representing the grasp angle is obtained through two vector components in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The grasp width values are calculated in pixels in the interval $[0, 150]$ for each grasp rectangle and normalized to the interval $[0, 1]$ before the training process.

Morrison, Corke and Leitner (2020b) used the Cornell Dataset to train a grasp network. Nevertheless, the Jacquard Dataset (DEPIERRE; DELLANDRÉA; CHEN, 2018) was used to evaluate the network performance through SGTs. Real experiments proved the relevance of data augmentation for the proposed method to be transferred to a real robot. Cornell and Jacquard’s dataset considers only RGB and Depth Images obtained from a single camera viewpoint. Therefore, random crops, rotations, and zooms were applied in the training process to obtain different viewpoints and depth invariances.

Ribeiro and Grassi (2019) used a trimming window of 320×320 resolution to crop the RGB images in Cornell Dataset under the restriction that no part of the object is outside the window area. The trimming windows were then resized to 224×224 to fit the network input. By applying this data augmentation, 297.388 images were generated.

2.2.3 Training data generation

Since the dataset used by Johns, Leutenegger and Davison (2016) for training the network is not already labeled as in Cornell dataset, a different approach was adopted to generate the training data. By using SGTs, a generic two-finger gripper is randomly positioned on the object’s surface and closes at a constant velocity and force for a fixed time. To the grasp is given a score of one if the object is successfully lifted by 20 cm above the surface,

and zero if the object falls. The overall score for the pose is calculated as an average of five attempts. To perform the simulations, 1000 synthetic objects were randomly selected from the ModelNet Dataset. The physical parameters such as finger torque, coefficients and friction, and the object density were manually tweaked in the simulation. The depth rendered images of the synthetic objects were randomly rotated and shifted to achieve 1000 new augmented images per original image. A noise model was added to each image since it was acquired in the simulation.

Pinto and Gupta (2016) developed a large-scale experimental study that increases the amount of data for learning to grasp. 50 *k* data points were collected over 700 hours of trial and error experiments. This dataset considers positive as well as hard-negatives (grasps that appear correct but fail). The objects used for generating the dataset vary in terms of graspability difficulty. During the collection of data, the robot moves 25 *cm* above the object, and the system filters the region of interest using an off-the-shelf Mixture of Gaussians (MOG). The grasp position and orientation are randomly sampled from that region of interest. The grasp is considered a success if the force sensor of the gripper detects a grasping and the object is lifted by 20 *cm*. The image corresponding to the successful grasp is then saved as a 277×277 pixels to fit the network input. The data was augmented by using random rotation. 150 objects were used to collect data points and 800 images were randomly sampled and evaluated for every trial.

Gualtieri et al. (2016) sampled thousands of 6D antipodal grasp candidates. These grasps are generated in simulation considering different cameras' views of the object. The grasp is considered successful if part of the point cloud of the object is contained between the fingers after the grasp approach. The BigBird dataset was used since the mesh is registered with the point cloud. Despite this, since the data from the BigBird dataset is acquired from a real camera, it contains noise that can worsen the grasping performance. 216 *k* grasps candidates were generated by using this technique.

Mahler et al. (2017) presented a new dataset, denoted Dex-net 2.0, with 1500 objects in which 1371 are synthetic models from the 3DNet dataset (WOHLKINGER et al., 2012) and 129 are laser scans from the KIT object dataset (KASPER; XUE; DILLMANN, 2012). For each synthetic object, a set of planar parallel-jaw grasps is generated on the object's surface given a generic gripper model. These grasps are binary labeled as success or fail based on a collision-free position and a robust quality metric (LIU; HE; CHANG, 2010) by using SGTs. Depth images are also generated for the object's mesh, considering random objects and the camera's planar position. Each depth image generated is rotated, translated, cropped, and scaled to align the grasp to the image's center, creating 6.7 million 32×32 pixels grasp depth images. Since the depth image is rendered in simulation, a noise model is added. This dataset is augmented by rotating each image by 180° and reflecting the image about its vertical and horizontal axes.

Mousavian, Eppner and Fox (2019) and Murali et al. (2020) sampled grasps in simulation using the software FleX (VICENT et al., 2016) (Figure 2.6). Candidate grasps were sampled based on the object geometry using the surface normal as a reference to align the gripper's z-axis. The distance between the gripper and the object are sampled uniformly between zero and the gripper's finger length. Grasps in a collision or grasps that don't have a volume between the fingers are discarded. Objects from six categories, including

boxes, cylinders, bowls, bottles, and mugs were used. A total of 10.816.720 grasps were sampled and 7.074.038 were positive grasps (successful grasps). In the simulation process, a free-floating parallel-jaw gripper and a free-floating object without gravity were used. Surface friction and object density are kept constant. After grasping the object, the gripper performs a pre-defined shaking motion. The grasping is successful if the object does not fall. From the 7.074.038 grasps sampled, only 2.104.894 grasps were successful (19.4%).



Figure 2.6: The grasp sampling simulation used by Mousavian, Eppner and Fox (2019) and Murali et al. (2020). The FleX sim (VICENT et al., 2016) was used to sample the grasps.

Sundermeyer et al. (2021) used the ACRONYM dataset (EPPNER; MOUSAVIAN; FOX, 2021), which consists of 8872 meshes from the ShapeNet dataset (CHANG et al., 2015). 17.7 million simulated grasps were sampled under varying friction. Only the grasps that are within a radius of 5 mm are considered as positive grasps. The main goal is to generate a set of grasps in the entire scene to increase the grasp coverage.

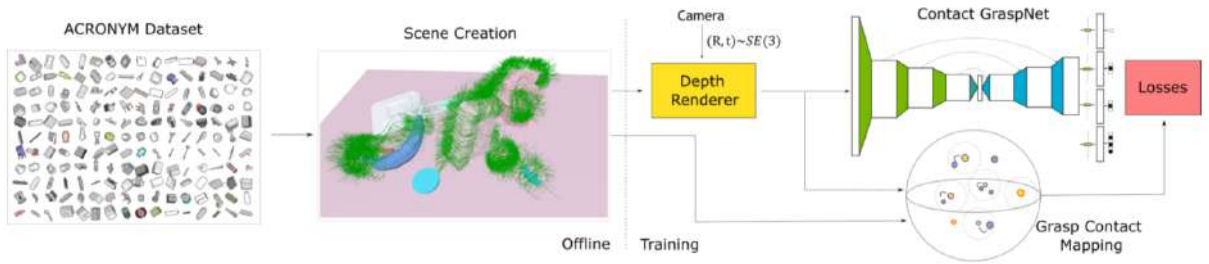


Figure 2.7: Training data pipeline adopted by Sundermeyer et al. (2021). Grasps are sampled from object’s point cloud. The objects are in stable positions. Only the grasps that are in contact with the objects are considered.

Breyer et al. (2021) generated a diverse set of labeled grasps by creating different piles of objects in clutter and the whole configuration space of the gripper. They created virtual scenes for simulated grasping trials using “pile” and “packed” objects. In the Pile configuration, the objects are dropped into a box placed on a flat surface. In the Packed configuration, a subset of taller objects is placed upright at random locations avoiding collisions with other objects (Figure 2.8). Pile configuration favors top-down grasps and Packed favors side-grasps.

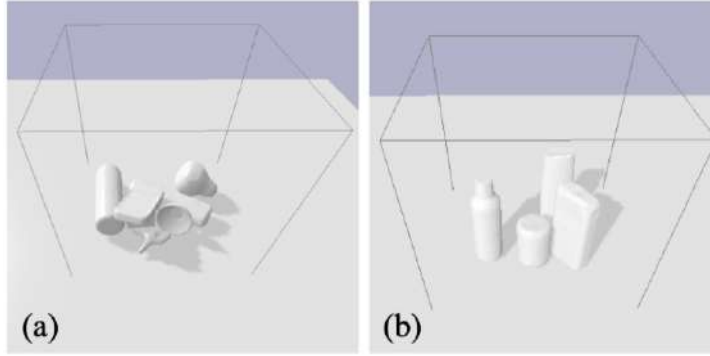


Figure 2.8: (a) Pile and (b) Packed object configuration adopted in Breyer et al. (2021).

Breyer et al. (2021) reconstructed a point cloud by fusing synthetic depth images. The virtual viewpoint of the camera was sampled using spherical coordinates. The authors sampled a point and associated normal from the reconstructed cloud. They tested six different grasps orientations about the normal vector by spawning the gripper at the given pose. Grasps with collision were labeled as negative. The grasp width was also stored. Only 120 points were sampled for each cloud. They generated a dataset with two million grasps by discarding redundant negative samples and limiting the angle by 90° about the normal vector.

Jiang et al. (2021) collected ground-truth grasps from physical trials in simulation using 303 objects for training and 40 objects for testing. The objects were organized in Pile and Package configuration as in Breyer et al. (2021) (Figure 2.9). Once the scene is created, grasp’s centers and orientations are sampled near the surface of the objects and these grasps are executed in simulation. The grasps are uniformly distributed on the object’s surface.



Figure 2.9: (Left) Pile and (Right) Packed object configuration adopted in Jiang et al. (2021).

Natarajan, Brown and Calli (2021) also generated synthetic data using simulation. For a given start pose, each compass direction was explored for a grasp. If no grasps were found, a further exploration of four random steps from each compass direction was performed three times. The camera path as well as the shorted working path exploration were saved. This was repeated for 1,000 random initial poses of six objects.

Ni et al. (2020) sampled single grasps in simulation. The goal was to obtain grasps g , metrics $Q_c(g)$, supplementary grasps $\tilde{g}$ and negative grasp points and normals (p_{neg}, n_{neg}) . The quality metric $Q_c(g)$ is the distance between the grasp origin and the nearest point in the boundary of $\mathcal{BG}$. $\mathcal{BG}$ is the convex hull of the primitive grasp wrenches (MISHRA; SCHWARTZ; SHARIR, 1987). N points were randomly sampled on the object's surface and their normals n were also calculated. The grasps were rotated for N_r times along the approaching vector n . A collision detection is performed for each grasp generated. If no grasps were generated for a given sampled point, this point and its normal are considered as negative and added to (p_{neg}, n_{neg}) . The grasp with maximum metrics value is considered as the best grasp (Figure 2.10). The object's density and friction are kept fixed in the data generation.

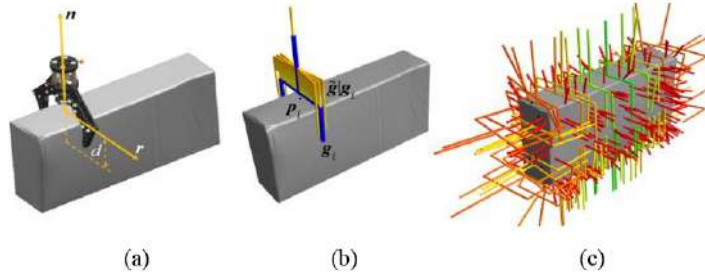


Figure 2.10: (a) Grasping model. (b) Best grasp and supplementary grasp (c) Sampled grasps varying from low to high inferred quality.

2.2.4 Preliminary Considerations

Table 2.2 shows the dataset used by each related work, the procedures applied in order to augment it, and the amount of augmented data. Unfilled fields mean that the author did not provide the related information.

Most grasping datasets just give a general gripper geometry (FANG et al., 2020). Consequently, grasp synthesis algorithms commonly ignore this information and let the collision check process for the motion planning algorithm. This approach is computationally expensive and waste computational time, since many grasps will be in collision with the environment and must be discarded. This process is usually done by neural networks (MURALI et al., 2020) that learns how to prone grasps that are in collision with the environment or analytically using the gripper geometry as proposed in this thesis.

Some grasping approaches require considerable time to generate training data (JOHNS; LEUTENEGGER; DAVISON, 2016). The physical parameters are frequently tweaked by hand when training the CNN in simulation. Despite this, the grasp score associated with the grasp performed in data generation highly depends on the friction between the gripper and the object and can lead to false scores (JOHNS; LEUTENEGGER; DAVISON, 2016). Nevertheless, a synthetic object can be positioned at a random orientation in the data generation stage using simulation, in contrast to the Cornell Dataset (MORRISON; CORKE; LEITNER, 2018, 2020b; RIBEIRO; GRASSI, 2019), BigBird (SINGH

Table 2.2: Comparative table comprising the dataset used in each related work, the process adopted to augment the dataset, and the amount of data generated by the augmentation process. Some authors do not perform data augmentation or do not provide any related information.

	Dataset	Data Augmentation	Amount of augmented data
Jiang et al. (2021)	Simulation	-	-
Breyer et al. (2021)	Simulation	-	2M
Sundermeyer et al. (2021)	ACRONYM	-	17.7M
Ni et al. (2020)	Simulation	-	-
Morrison, Corke and Leitner (2020a)	EGAD	-	-
Morrison, Corke and Leitner (2020b)	Jacquard	-	54k
Mousavian, Eppner and Fox (2019)	Simulation	-	2M
Morrison, Corke and Leitner (2019)	Cornell	Crops, zooms, and rotations	51k
Mahler et al. (2019)	Dex-Net 4.0	Rotations and reflections	-
Ribeiro and Grassi (2019)	Cornell	Trimming Window	297k
Morrison, Corke and Leitner (2018)	Cornell	Crops, zooms, and rotations	51k
Mahler et al. (2017)	Dex-Net 2.0	rotations and reflections	6.7 M
Johns, Leutenegger and Davison (2016)	ModelNet	Shift and Rotation	1M
Lenz, Lee and Saxena (2015)	Cornell	-	-
Pinto and Gupta (2016)	Experience	Rotations	-
Gualtieri et al. (2016)	BigBird	-	216k

et al., 2014), Jacquard (DEPIERRE; DELLANDRÉA; CHEN, 2018), and other grasping datasets that offer only RGB and Depth images.

Data distribution is relevant for archiving a high grasping performance since the more diverse the dataset is, the higher the success rate will be in a physical robot (MAHLER et al., 2019). Datasets such as ModelNet, 3DNet, and KIT are commonly employed for machine-learning-based grasp generation. Despite this, these datasets were built on the premise of object recognition applications. Furthermore, they contain a low variety of object classes, leading to poor grasping performances when considering unknown and complex objects (MORRISON; CORKE; LEITNER, 2020a).

According to Pinto and Gupta (2016), creating a grasp dataset using hand labeling ground-truth grasps is challenging and biased since an object can be grasped in multiple ways. Nevertheless, generating training data from offline simulation instead of hand-labeled data can lead the grasping algorithm to choose a graspable area that is not adequate. For instance, a knife might be grasped by the blade during the training or a hot glue gun by the barrel (GUALTIERI et al., 2016).

The number of augmented data varies significantly in each grasping method. Some research adopts a high number of data (RIBEIRO; GRASSI, 2019) compared to other (MORRISON; CORKE; LEITNER, 2018, 2020b). Nonetheless, it seems reasonable to generate significantly more data when the number of parameters of the network architecture is higher (RIBEIRO; GRASSI, 2019).

According to Morrison, Corke and Leitner (2020b), data augmentation techniques, such as zooms, rotations, and crops can be applied to RGB and depth images to get different camera viewpoints. Nevertheless, these operations do not correspond to different viewpoints, but the same image is translated, cropped, or rotated. Johns, Leutenegger

and Davison (2016) states that synthetic depth images are more realistic than RGB images, as these often struggle to model illumination and texture with sufficient realism.

Pinto and Gupta (2016) stated that labeling a dataset by hand such as in Cornell Grasping Dataset does not represent well the grasps. The reason is that other parts of the object not labeled as positive cannot be considered as negative grasp only because it was not labeled. Nevertheless, it is proved that it is not necessary to consider the negative grasps to achieve a high grasping performance (MORRISON; CORKE; LEITNER, 2018).

Gualtieri et al. (2016) used a point cloud of the object to be grasped to generate 6D grasp poses. Despite the performance stated, this method struggles with filtering only the point cloud of the object of interest, leading the grasping to poor performance in clutter.

According to Mahler et al. (2017), aligning the grasp horizontally in the center of the image removes the necessity to learn rotational invariance. It discards the need to discretize a set of orientations as in (JOHNS; LEUTENEGGER; DAVISON, 2016). Nevertheless, centering the grasp in an image prevents the dataset to have multiple labels per image as found in Cornell Dataset.

Mousavian, Eppner and Fox (2019) and Ni et al. (2020) generate data in simulation considering only one object at a time. Consequently, the grasping learning model will not learn to avoid collisions with nearby objects and will be required a collision avoidance method to work with the grasp generation model. Jiang et al. (2021), Breyer et al. (2021) and Sundermeyer et al. (2021) generated datasets with objects organized in pile or packed. Hence, the grasp model undergoes learning to comprehensively analyze the entire environment and prevent collisions while executing the grasping action.

2.3 NETWORK ARCHITECTURE AND TRAINING

According to Mahler et al. (2018), the training proceedings used for machine learning should be reported to allow for a reasonable comparison between the grasp methods. Therefore, subsection 2.3.1 focuses on the comparison between the network training process adopted in each related work. Subsection 2.3.3 depicts relevant considerations about the training process in the related works.

2.3.1 Training process

Johns, Leutenegger and Davison (2016) trained a CNN to classify every inferred pose in a discretized grasp map in terms of its score. This network takes a 240×240 pixels depth image as input. The CNN architecture is composed of five convolutional layers followed by max pooling, and two fully connected layers. The gradient descent optimization method was applied to mini batches using a Softmax classifier. The training took an average time of 10 minutes per object and one week to complete since more than one million augmented images were used.

Lenz, Lee and Saxena (2015) proposed a two-stage cascaded CNN detection system based on deep learning. The first stage of the CNN is faster due to fewer parameters of the architecture. This stage filters the grasps on the objects by scoring potential grasps in an

RGB+D image. The second stage is a deeper CNN that is more precise and rejects false positive grasps by inferring the best-ranked grasp. An RGB+D image is used as input for this network. A regularization method is applied to improve the network generalization performance.

Pinto and Gupta (2016) presented a pre-trained CNN based on AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) with 18 M parameters to generate a robotic grasp. This network is responsible for sampling image patches and predicting the grasping angle. It is accomplished by classifying 18 discretized angle bins. The network architecture comprises five convolutional layers and two fully connected layers. The training took 700 hours and collected 50 k data points.

Gualtieri et al. (2016) used CAD models for training a CNN to classify grasps. This network comprises four convolutional layers. The network architecture used was based on the LeNet (LECUN et al., 1998). The network training was performed in Caffe (JIA et al., 2014). Approximately 85% of the training data was used in training.

Mahler et al. (2017) presented a CNN architecture, denoted Grasp Quality Convolutional Neural Network (GQ-CNN). The GQ-CNN architecture comprises four convolutional layers in pairs followed by three fully connected layers. The input of the network is divided into two: one for a depth image representing the grasp and the other for the height from the grasp pose. The stochastic gradient descent with momentum was used in training with the Rectified Linear Unit (ReLU) activation function for the output layer. The weights of the network were initialized by a probabilistic Gaussian model.

Morrison, Corke and Leitner (2018) presented a CNN with a fully convolutional topology, denoted Generative Grasping Convolutional Neural Network (GG-CNN). The GG-CNN generates a grasp pose in a pixel-wise grasp map in terms of a quality image. This network takes a 300×300 pixels depth image as input. The L2 loss function and the Adam optimization (KINGMA; BA, 2014) were used in the training. 80% of the dataset was used in training and 20% in evaluation. The GG-CNN outputs four 300×300 pixels images. One image is related to the grasp quality, two images are concerned with the grasp angle in terms of its x and y components, and one image is related to the grasp width. The network architecture was defined by evaluating the performance of 95 similar network architectures in respect of the grasping time and success rate.

Ribeiro and Grassi (2019) presented a CNN architecture to predict the best grasp position in uncluttered environments. This network takes as input 224×224 pixels RGB images. The network architecture comprises two convolutional layers, followed by batch normalization, max pooling, convolution filters, and two fully connected layers with a dropout rate of 50%. The activation function ReLU is used except for the output layer, in which the linear function is used. The training was divided into image-wise, where the dataset is randomly organized into sets, and object-wise, where all similar images are organized in the same set. Approximately 84.5% of the training dataset was used in training, 0.8% was used for validation, and 14.7% was used for testing.

Morrison, Corke and Leitner (2020b) presented a new version of the GG-CNN, denoted GG-CNN2. GG-CNN2 is a fully convolutional network based on the semantic segmentation architecture (YU; KOLTUN, 2015). It uses the same input and output as the GG-CNN. The GG-CNN2 is a result of several tests performed by modifying its

architecture and evaluating its performance through a SGT simulator. In this simulation, the convolutional layers, filter sizes, dilation parameters, data augmentation, and output filtering are modified in order to achieve a better success rate and less grasping time. This network is trained using the Jacquard dataset with 95% left for training and 5% for testing. The L2 loss function and the Adam optimization (KINGMA; BA, 2014) were used in the training.

Mahler et al. (2019) developed a CNN to learn a grasping policy in synthetic training datasets using analytic models. This network maximizes the chance to grasp each object in a cluttered bin considering a fixed time per grasp. This method considers uncertainties in the environment related to the object’s properties such as geometry, friction, and the center of mass. The loss function used in training takes the gripper type into account (parallel or suction gripper). Therefore, each object is grasped by the appropriate gripper. The network architecture comprises four convolutional layers and two fully connected layers and used an 80-20 training-to-validation image-wise split of the Dex-Net 4.0 dataset.

Mousavian, Eppner and Fox (2019) developed a grasping technique called GraspNet, which is a 6D grasping CNN based on Variational Autoencoders (VAEs) (KINGMA; WELLING, 2013). VAEs are deep generative models developed based on classic Autoencoders. This network consists of an encoder and decoder that map the input data to a latent space of reduced dimensionality. The encoder and decoder use the PointNet++ (QI et al., 2017) network architecture to extract spatial characteristics from each point of the object’s point cloud and the robot’s end effector for each grasp generated, whether successful or not.

GraspNet (MOUSAVIAN; EPPNER; FOX, 2019) has two modules: generator and evaluator. The generator module relies on different samples of a latent space and the object and gripper’s partial point cloud to produce different grasps. The evaluator module accepts or rejects the grasps based on their probability of success. In addition, there is a grasp refinement process, which applies a grasp shift, given by Δg , to increase the grasp of success rate. GraspNet was trained from grasps obtained in the software FleX (VICENT et al., 2016). In the training process, each object is rendered from a random view and 64 grasps are sampled using stratified sampling to make sure that sampled grasps have enough diversity. Both generator and evaluator are trained using Adam optimizer with a learning rate of 0.0001.

GraspNet for target-driven (MURALI et al., 2020) improves the GraspNet (MOUSAVIAN; EPPNER; FOX, 2019) by adding a collision network called CollisionNet to predict a clutter-centric collision score. The ground truth labels are generated in simulation with a collision checker assuming full state information. CollisionNet is optimized using cross entropy loss. The goal of this method is to avoid collisions with nearby obstacles when considering clutter scenes.

Ni et al. (2020) presented a CNN based grasping network based on PointNet++ (QI et al., 2017). The input of the PointNet++ is the point cloud of the object to be grasped. Two layers of one dimensional convolutional filters were added on the top of the PointNet++ network. The output of these two convolutional layers has the dimension 8192×9 , corresponding to a score, category, normal and rotational blocks. The network

was trained by stochastic gradient descent using fixed learning rate of 10^{-4} , momentum of 0.9, and weight decay of 2^{-5} .

Breyer et al. (2021) employed the set abstraction and feature propagation layers proposed in PointNet++ (Qi et al., 2017) to build an asymmetric U-shape network. The network takes $p = 20,000$ ($p \in \mathbb{R}^3$) points as input and predict grasps for only $m = 2048$ farthest points of the input. The network has four heads with two 1D-Conv layers each and per-point outputs $s \in \mathbb{R}$, $z_1 \in \mathbb{R}^3$, $z_2 \in \mathbb{R}^3$, $o \in \mathbb{R}^{10}$, from which the grasp representation is constructed. The vectors representing the grasp approach and surface normal are orthonormalized using Gram-Schmidt orthonormalization to facilitate the regression of 3D rotations. The loss is evaluated at all outputs points using binary cross entropy. A set of pre-defined points representing the gripper are transformed considering the inferred grasping points. Breyer et al. (2021) formulate the 6-DoF grasp loss as a weighted minimum average distance between ground truth and inferred gripper points.

Sun et al. (2023) proposed a grasping pipeline with three stages called Candidate Generation Network (CGN), Reliable Adjustment Module (RAM), and Gaussian Mixture Model (QAN). CGN is a neural network, based on PointNet++ (Qi et al., 2017), that predict grasp scores through features extracted from point clouds. RAM is responsible for adjusting the grasp width and position to avoid collision with nearby obstacles. QAN evaluates the grasp quality and selects the best grasp. The losses are based on the graspability classification, calculated using cross-entropy, approach direction, calculated using mean squared error, and in-plane rotation angles, calculated using log-likelihood loss. The network was trained using Stochastic Gradient Descent (SGD). The learning rate was set to 0.02 and the momentum to 0.9. The network was trained for 100 epochs with a batch size of 128.

Lundell et al. (2023) suggested the use of Conditional Variational Encoders (CVAE) to generate constrained 6D grasps for completing specific tasks such as squeezing out liquid from a bottle. The grasps are generated on a specific part of the object. PointNet++ (Qi et al., 2017) is used to extract feature of the object. Eleven features were extracted from the point cloud such the 3D position of each point, the binary feature indicating if the point belongs to the target area or not, and 7-dimensional grasp pose representation. A grasp evaluator was also implemented to prone out bad quality grasps based on the object and gripper point cloud.

2.3.2 Evaluation metrics

Analogous to the object detection task, the grasping task incorporates distinct metrics for assessing network performance within the evaluation phase. It is important to note that the metrics employed during the grasping test (referenced in subsection 2.6.1) differ from those used in the context of object detection. This distinction is based on the necessity of rigorously testing the grip synthesis method, which includes applying real-world or simulated grasping motions using robotic equipment to enable the evaluation of grasp success rates. As a result, this evaluation process differs from the testing procedures applied to the object detection task.

The most common metrics used to evaluate the grasping task are (NEWBURY et al.,

2023):

- Intersection-over-union (IoU): Ratio between the intersection and the union of the predicted and ground truth bounding boxes (MORRISON; CORKE; LEITNER, 2018);
- Coverage: Percentage of sampled ground truth grasps that are within a threshold distance of any the generated grasps;
- Grasps prediction accuracy: Percentage of grasps outcomes correctly predicted (number of successful grasp predictions over the number of predictions);
- Precision: Percentage of true positive grasp predictions similar to the precision metric used in object detection or the number of true positive grasp predictions over the number of selected positive grasps; and
- Precision@k: Average precision of the top-k ranked grasps. Ap_μ denotes the average precision for k ranges with a force closure parameter lower than μ (SUN et al., 2023).

These metrics are simpler to employ during the training or evaluation process given that it is challenging to determine which grasping, out of all the ones generated (in the case of sampling-based grasping methods), is successful, especially in the case of cluttered sceneries. It would not be practical in a real-world situation to put back the objects in the scene exactly as they were before the grasp was applied.

2.3.3 Preliminary Considerations

Table 2.3 summarizes the network’s input size, activation function, weight initialization, optimizer, number of parameters, and the library used for training.

Table 2.3: Networks comparative table.

	Input Size	Output Layer Configuration	Weight Initialization	Optimizer	Number of parameters	Framework
Sun et al. (2023)	Point cloud	-	-	SGD	-	PyTorch
Breyer et al. (2021)	-	-	-	Adam	-	-
Morrison, Corke and Leitner (2020a)	300x300	ReLU	Xavier	Adam	66.000	PyTorch
Mousavian, Eppner and Fox (2019)	-	-	-	Adam	-	Tensorflow
Mahler et al. (2019)	96x96	-	Gaussian	GD	-	-
Ribeiro and Grassi (2019)	320x320	ReLU	Xavier	Nadam	1.5 million	Tensorflow
Morrison, Corke and Leitner (2018)	300x300	ReLU	Xavier	Adam	62.400	Keras
Mahler et al. (2017)	32x32	ReLU	Gaussian	GD	18 million	Tensorflow
Johns, Leutenegger and Davison (2016)	240x240	Softmax	-	GD	7.3 million	Tensorflow
Pinto and Gupta (2016)	227x277	Softmax	Gaussian	-	18 million	-
Gualtieri et al. (2016)	-	ReLU	Pretrained	-	-	Caffe
Lenz, Lee and Saxena (2015)	24x24	-	Unsupervised Feature Learning	Log-sum-exponential	-	MATLAB

Hand-engineered, deterministic or analytic methods are avoided by recent research since deep learning methods using CNN have achieved an outstanding performance.

These results are observed even in real experiments with unknown and dynamic objects (LENZ; LEE; SAXENA, 2015; MORRISON; CORKE; LEITNER, 2018; MAHLER et al., 2018).

Pinto and Gupta (2016) state that the network would fail to generalize to new objects if the amount of data used in training is fewer than the number of parameters of the network. Despite this, Morrison, Corke and Leitner (2018, 2019, 2020b) have proved in real experiments, that the number of data does not necessarily need to be larger than the number of parameters for object generalization. The network architecture and the post-processing have proved to be the key points of a good grasping performance.

Depth images are commonly applied instead of RGB images since it better represents the object features (GUALTIERI et al., 2016; MOUSAVIAN; EPPNER; FOX, 2019). In addition, color images are often avoided due to object color variations and ambient luminance. To avoid this limitation, some grasping methods have adopted depth images instead (MORRISON; CORKE; LEITNER, 2018, 2019).

2.4 GRASPING PRE-PROCESSING AND POST-PROCESSING

This section reviews the methods used to improve the network inference in real or simulated environments. These methods are related to the procedures employed to the network input, as described in subsection 2.4.1, or the network output, as described in subsection 2.4.2.

Figure 2.11 shows the grasping stages in a pipeline presented in (NEWBURY et al., 2023). The first stage is the grasp synthesis, which is responsible for generating a set of grasps. The second stage is the trajectory planning, which is responsible for planning the robot's trajectory to approach the object. The third stage is the grasp execution and the last one involves trajectory planning as well. The pre-processing and post-processing methods are commonly applied to the network input and output in the first stage of this pipeline. However, trajectory planning can also be applied to filter the grasps generated by the network as a post-process method Gualtieri et al. (2016) or used to get multiple view of the objects as a pre-process method Morrison, Corke and Leitner (2019).

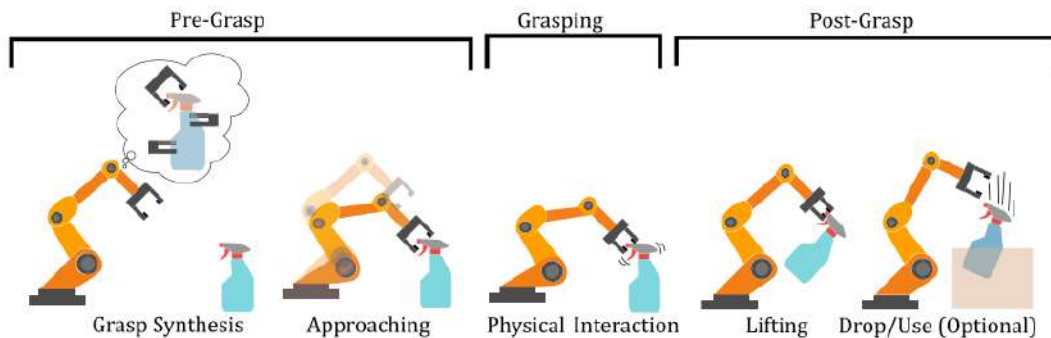


Figure 2.11: Grasping pipeline stages, including grasp synthesis, trajectory planning (before approaching), and grasp execution (NEWBURY et al., 2023).

2.4.1 Pre-processing

Morrison, Corke and Leitner (2018) applied the inpaint operation (BRADSKI, 2000b) to the network input depth images. It removes the values where the depth measurements are not available due to fails in sensing. This processing was also adopted by Johns, Leutenegger and Davison (2016). The inpaint operation replaces the pixels or an average of pixels with the nearest non-zero value.

Mahler et al. (2017) generate a discrete set of antipodal grasps, sampled uniformly at random in the image space for the object’s surfaces normals. Firstly, areas of a high gradient in the depth image are found. Then, a set of candidate antipodal grasps are generated by using a rejection sampling through pairs of pixels. The antipodal grasps are converted to 3D by discretizing the grasp height relative to the table surface. Due to limitations related to this derivative method, Mahler et al. (2017) also presented a derivative-free optimization to obtain a robust grasp by iteratively re-sampling grasps from a learned distribution using a Gaussian Mixture Model (GMM).

In order to compute the best grasp during the picking attempt, Morrison, Corke and Leitner (2019) developed a controller, denoted Multi-View Picking (MVP). It combines visual grasp predictions from multiple camera views along a trajectory. It is accomplished by reducing the entropy associated with the grasp prediction in clutter. Before the grasp attempt, it adapts the robot’s trajectory in real-time in order to explore areas of higher entropy. This exploration increases grasping success when considering objects in clutter. The trajectory is defined as a set of K discrete points in which a visual grasp detection observation is performed. The initial trajectory’s position is defined at z_{max} and the final trajectory’s position is equivalent to the grasp pose.

Sun et al. (2023) decouples the orientation as viewpoint classification and in-plane rotation prediction. Each approach vector are classified into V predefined viewpoints. The grasping scoring is used to predict the approaching direction, where the score of the approaching direction represents the success rate of grasping in this direction. The point clouds are transformed to the viewpoint which has the highest score and used to estimate the in-plane rotation angle.

2.4.2 Post-processing

Johns, Leutenegger and Davison (2016) applies a covariance matrix to represent the gripper’s pose uncertainty. The grasp function is convolved with a probability density function given this uncertainty. This is performed by smoothing the grasp function in 3-dimensions, the position (x, y) and the orientation θ , with a kernel corresponding to a Gaussian distribution. In addition, trilinear interpolation is performed over the pose space to improve the precision beyond the pose discretization level. This method is also adopted in Morrison, Corke and Leitner (2018) since the authors noticed that using the Gaussian kernel found in Johns, Leutenegger and Davison (2016) improved the grasping performance by 10% because outliers are removed, increasing the robustness of the grasps.

In order to mitigate the imprecision found in the Baxter robot, Pinto and Gupta (2016) re-rank a sample of the top 10 grasps identified in the neighborhood of the image patches generated. The average of the best angle scores for the neighborhood patches is

assigned as the new patch score for the grasp configuration. This method improved the precision of the grasping system.

Gualtieri et al. (2016) used the trajectory planning denoted TrajOpt (SCHULMAN et al., 2013) to generate an offline trajectory. A constraint was applied to TrajOpt to align the line-of-sight axis of the camera towards the bin’s center, keeping a distance of 40 *cm* from it. The IKFast and OpenRave (DIANKOV; KUFFNER, 2008) was used to generate the inverse kinematics and check collisions.

When the CNN has a high frequency, the fast-shifting between grasp poses should be avoided. Considering this, Morrison, Corke and Leitner (2018) obtain three grasps from the highest local maxima and consider the best grasp acquired in the previous iteration. Since the robot’s movement is slow compared to the control loop, the frames do not considerably change. The grasp system is started by tracking the global maxima at every grasp attempt.

2.4.3 Preliminary Considerations

In some approaches, the grasp quality is simultaneously predicted over a set of discrete grasp candidates (JOHNS; LEUTENEGGER; DAVISON, 2016; PINTO; GUPTA, 2016). Although it is time-consuming, the execution time can be reduced by reducing the grasp candidates (LENZ; LEE; SAXENA, 2015; WANG et al., 2016). It is achieved by conciliating the number of sampled grasps with execution time. Besides that, several potential grasps are ignored.

The process of generating grasps from depth image gradients is commonly applied when the network does not return the best grasp but rank the identified grasps in the image (MAHLER et al., 2017). Besides that, it may be necessary to select a feasible grasp if the CNN generates a set of grasps (MOUSAVIAN; EPPNER; FOX, 2019).

2.5 TEST OBJECTS

This section describes the objects used in real experiments in order to evaluate the grasping performance.

Johns, Leutenegger and Davison (2016) used a set of known 20 everyday objects of different shapes, frictional coefficients, and sizes as shown in Figure 2.12a. Lenz, Lee and Saxena (2015) used a diverse set of 35 objects within a size of $0.3\text{ m} \times 0.3\text{ m} \times 0.3\text{ m}$ and weighing at most 2.5 *kg* from their offices, homes and lab (Figure 2.12b). The objects have simple geometry and were chosen based on the gripper width limit. Similarly, Gualtieri et al. (2016) used common household objects that weigh less than 500 *g* and have at least one part that fits within the gripper used (Figure 2.12c).

Pinto and Gupta (2016) divided the test objects into two categories: a set of objects used for training the network and another set of objects never seen in training. Both sets of objects consist of household objects with simple shapes, as shown in Figure 2.12d.

In order to increase the grasping difficulty in real experiments and standardize the evaluation performance in real experiments, Mahler et al. (2017) proposed a set of eight objects with adversarial geometric features such as smooth curved surfaces and narrow

openings (Figure 2.12e). The objects are easily reproducible with 3D printing; therefore, eliminating the need for purchasing matching objects. These objects also fit most grippers on the market such as Robotiq 2F-80/2F-140. The household objects (Figure 2.12f) were also employed for testing the grasping performance.

According to Morrison, Corke and Leitner (2018), experiments are commonly performed with objects that are not easily reproducible. These objects are very easy to grasp or unfeasible small, large or heavy for many robot arms and grippers such as the objects found in ACRV Picking Benchmark (APB) (LEITNER et al., 2017) and Yale-CMU-Berkeley (YCB) Object Set (CALLI et al., 2015). In order to make the experiments reproducible, Morrison, Corke and Leitner (2018) used the adversarial objects found in Mahler et al. (2017) (Figure 2.12g). They also used a set of household objects with different surfaces, sizes and grasp difficulties found in Leitner et al. (2017) and Calli et al. (2015) (Figure 2.12h). The household objects found in Viereck et al. (2017) (Figure 2.13) were also employed to compare the robotic grasping technique used.

Mahler et al. (2019) separated objects into four sets with 25 objects each in terms of their difficulty to grasp. The first set (Figure 2.14a) consists of prismatic and circular solids. The second set (Figure 2.14b) contains common household objects weighting up to 500 g. The third set (Figure 2.14c) consists of 25 novel objects with few accessible and perceptible grasps due to adversarial geometry, transparency, specularly, and deformability. The fourth set comprises objects that cannot be grasped with Dex-Net 4.0 due to reflectance and material properties which make the object imperceptible to a depth camera (MAHLER et al., 2019).

The test objects can affect the assessment of grasp and the lack of standardization can make comparisons between grip techniques difficult. Researchers commonly adopt unique household or commercial products that are not easily reproducible or available to use in evaluation. Given this limitation, Morrison, Corke and Leitner (2020a) created a set of 49 3D-printable objects for performance evaluation that varies in shape and grasp complexity (Figure 2.15). Nevertheless, the objects lack semantic meaning which may not generalize well for real-world applications (NEWBURY et al., 2023).

2.5.1 Preliminary Considerations

The grasping tested performance highly depends on the dataset used (MORRISON; CORKE; LEITNER, 2020b). Therefore, it is essential to standardize the test objects used for reasonable performance comparison. In addition, commonly adopted test objects are not applicable for all robotic grippers. Therefore, a set of test objects that can easily be printed and scaled to fit the most gripper’s limitation while maintaining a fair comparison is required (MORRISON; CORKE; LEITNER, 2020a).

Some objects are not perceived well by depth cameras, such as those that are reflective, transparent, or black (MAHLER et al., 2019). Narrow objects can also be difficult to grasp since collisions may happen with the gripper. This is a consequence of a few similar objects used in training. Curved objects are also difficult to grasp since the object slides from the gripper (MORRISON; CORKE; LEITNER, 2020a).



(a) Simple geometry objects used by Johns, Leutenegger and Davison (2016)



(b) Simple geometry objects from home, lab, and offices (LENZ; LEE; SAXENA, 2015)



(c) Household objects used by Gualtieri et al. (2016)



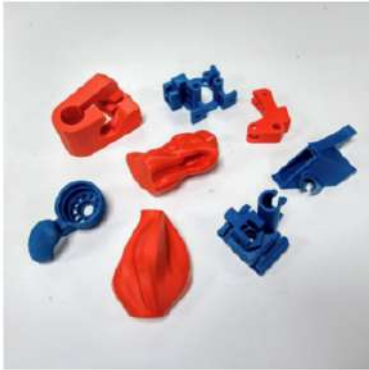
(d) Simple household geometry objects (PINTO; GUPTA, 2016)



(e) Eight objects with adversarial geometry (MAHLER et al., 2017).



(f) Household objects (MAHLER et al., 2017).



(g) Eight adversarial objects (MAHLER et al., 2017).



(h) Household objects (LEITNER et al., 2017).

Figure 2.12: Objects used by different authors to test grasping algorithms.

2.6 EVALUATION PROCEDURES, BENCHMARKS, AND RESULTS

According to Mahler et al. (2018), the performance may be categorized based on the target application, such as warehouse picking, pick and place, industrial kitting, stowing, assembly, etc. To standardize the grasping performance measurement, some authors presented benchmarking processes that are also discussed in this section. This section is organized in the following way:

- Subsection 2.6.1 describes some grasping benchmarks commonly adopted in researches. These benchmarks are protocols created to fairly evaluate grasping methods;



(a) Household objects presented by Viereck et al. (2017).

(b) Household objects reproduced by Morrison, Corke and Leitner (2018).

Figure 2.13: Objects used by Morrison, Corke and Leitner (2018) to test the performance of the robotic grasping algorithm presented.

- Subsections 2.6.2 and 2.6.3 covers the procedures employed in simulated and real experiments by distinct authors. Although these procedures are not commonly standardized, making it difficult to compare the grasping performances, it is important to note how the authors are evaluating their grasping performances;
- Subsection 2.6.4 establish different Definition of a Successful Grasp (DSG) usually chosen;
- Subsection 2.6.5 details the performance achieved by each grasping method according to the DSG and evaluation procedure or benchmark adopted; and
- Subsection 2.6.6 describes observed aspects of related works in terms of the DSG adopted, performance achieved, and experimental procedures employed.

2.6.1 Grasping benchmarks

As previously mentioned, there are image recognition challenges in computer vision such as COCO (LIN et al., 2015) and VOC (EVERINGHAM et al., 2010a) that assisted the evolution of object detection techniques over the last years. Unfortunately, there are no similar challenges for robot grasping due to its mechanical, sensorial, and algorithmic complexity (BEKIROGLU et al., 2019). Despite it, Liu et al. (2021) provided a remote robot platform testing environment to allow fair comparisons between grasping algorithms called OCRTOC (Cloud-Based Competition and Benchmark for Robotic Grasping and Manipulation). Nevertheless, the grasping evaluation is not as easy as the COCO or VOC challenges. It requires manual organization of the environment, which reduces the availability for researchers worldwide. Different authors tried to establish protocols to standardize the grasping performance evaluation, including the type of objects that should be used, how objects should be placed in the robot workspace and metrics calculation (BEKIROGLU et al., 2019).

The type of benchmark procedure highly depends on the grasping application. There are benchmarks for pick-and-place in tabletop scenarios (CALLI et al., 2015; BEKIROGLU



Figure 2.14: (a) Level 1 objects consisting of prismatic and circular solids (b) Level 2 objects including clear plastic and household objects with a lower level of graspability. (c) Level 3 objects with adversarial geometry and material properties (d) Level 4 objects with different reflectances and material properties which affect the ability to form a vacuum seal on the object surface (MAHLER et al., 2019).

et al., 2019), pick-from-the-self scenario (LEITNER et al., 2017), bin-picking (MNYUSIWALLA et al., 2020). There are robot manipulation benchmarks such as Collins et al. (2019), but these do not evaluate the grasping performance itself since it is mainly focused on measuring the gap between simulated and real robot motions.

The goal of the benchmarking proposed by Bekiroglu et al. (2019) is to analyze only the performance of the grasping algorithm in tabletop environments, excluding the motion planning and other factors that could influence the grasping performance. The main question asked by the authors is: “How do we evaluate the influence of a grasp planning algorithm independently of the vision system, arm and hand?”. The answer is complex since authors use different hardware, pipelines, software, etc. Despite that, they make an effort to increase the comparability of grasping methods.

There are robotic grasping challenges such as Amazon Picking (CORRELL et al., 2016) and the IROS Robotic Grasping and Manipulation Competition (SUN et al., 2016). Nevertheless, this challenge evaluates the grasping method at a system level. In other words, we cannot assure that the robot achieved a bad performance due to the grasping method since the hardware design and perception system could highly influence this result (CALLI et al., 2021).

2.6.1.1 Benchmark protocols Mnyusiwalla et al. (2020) proposed a benchmark protocol, based on the YCB dataset evaluation protocol (CALLI et al., 2015), to evaluate grasping methods in pick-and-place systems, inspired by a task in the logistic domain by picking up fruits and vegetables from a container and placing them in an order bin.

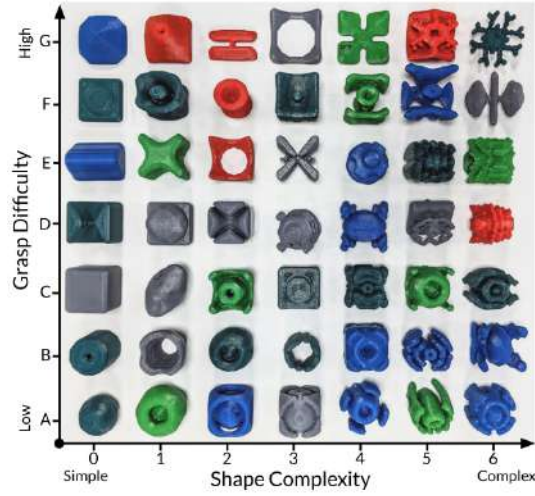


Figure 2.15: 3D-printed objects for grasping evaluation performance. These test objects varies in terms of complex geometry (left to right), and graspability (bottom to top).

They were inspired by the grocery use case of Ocado, the world’s largest online-only supermarket. The focus of the research found in Calli et al. (2015) is to test different end-effectors in cluttered and geometrically constrained set-ups. The goal is to autonomously pick one-by-one all objects placed in a non-mixed storage container, transport and place them in a delivery container in the minimum possible time. Since it is very difficult to find why the grasping failed, the authors break down the grasping task into pre-grasping, grasping, transport, and placement.

Figure 2.16 comprises the following stage:

1. Pre-grasping phase: is the phase in which the end-effector moves to the vicinity of the object to be grasped.
2. Grasping phase: System’s attempt to grasp the object. Ends when the object loses contact with the bin.
3. Transport phase: Motion of the system when the hand fully supports the object, up to the position when the gripper is over the placement area.
4. Placement phase: The system intentionally releases the object inside the target bin from a maximum allowed height (20 cm) over the bin.

Calli et al. (2015) state that although the object may fail in the transport phase due to an unstable grasping in the grasp phase, this evaluation methodology helps track the issue using per-phase analysis. The bins must have an opening of exactly 60 cm $\times$ 40 cm ($L \times W$) and a minimum height of 15 cm. The bin position is not standardized since the reachability and workspace of the robotic arm vary. Nevertheless, the pose of the bin must be fixed with respect to a static coordinate frame and must be reported. Calli et al. (2015) standardized the set of objects of the experiments. They used a net bag of limes, mango, loose-leaf salad bag, cucumber, and punnet (small plastic box) of



Figure 2.16: Phases considered for benchmarking in pick-and-place tasks (MNYUSIWALLA et al., 2020).

blueberries. Besides that, the authors proposed a set of pre-defined environments with different clutter levels (Figure 2.17). The scenarios are designed to mimic the initial placement of objects corresponding to optimal packing. The system must not know any object pose inside the bin a priori.

The objects dropped outside the storage container should not be introduced again, and no external intervention is allowed during the execution. The system must stop picking when it is empty.

Morgan et al. (2019) proposed a pick-and-place benchmarking based on the Box and Blocks Tests (BBTs) and its variations modified-BBT (M-BBT) and targeted-BBT (T-BBT). BBT test consists of two containers separated by a barrier, with one container holding 150 colored wooden blocks 2.18. Within one minute, the robot must transfer the blocks to the empty container, ensuring that the end effector crosses the division between the containers. The BBT benchmarks were applied because they increasingly challenge the robot grasping pipeline, including motion planning, the perception system, etc.

The standard BBT has been utilized for clinical tests in physically impaired individuals. This test provides norms for able-bodied individuals, ages 20-92 in 12 ages groups (318 females and 310 males). It allows meaningful comparison between humans and robots in pick-and-place operations since there are extensive public data of individual tests.

There are protocols established by Morgan et al. (2019) to use the BBT in robotics. The bins must be positioned on a support surface in a reachable area for the robot arm. The bin's location is optional. The lid and the containers must have the same length. Each test has its block organization inside the bin. The bin to be filled is optional (left or right). The end-effector must start in a position outside of either container. The blocks must be part of the YCB set (CALLI et al., 2015). They must have the same size, colors,



Figure 2.17: Protocol scenarios sorted by object type and amount of clutter (MNYUSIWALLA et al., 2020).

weight, and texture of the blocks from the YCB dataset. Markers cannot be placed on any of the blocks. If the center of the bin moves more than 2.54 cm, or the cube rotates more than 10° , the task must be restarted and the score for that task execution is zero.

Liu et al. (2021) created a robot benchmark setup, called ORCTOC (Cloud-Based Competition and Benchmark for Robotic Grasping and Manipulation) to perform remote experiments of robotic grasping algorithms. This benchmark is focused on object rearrangement, which is a canonical robotic task as it includes object grasping, recognition, manipulation planning, and reasoning. The complexity of a rearrangement task depends on the object features, goal definition (exact or coarse), motions for the rearrangement (random or ordered), and the test environment (real or simulated). Coarse goal definition means that the goal pose of the object is not important, such as placing the object in a bin (Figure 2.19).

The objects in the ORCTOC benchmark system are selected from a subset of objects that can be easily found in daily life. Some objects used in testing are scanned to provide the training data for the challengers to build their own solutions. Five tasks are offered to the users with increasingly clutter, therefore increasingly difficulty. This benchmark only evaluates the grasp success rate, error, and improvement percentage from the last winner of the benchmark considering the grasp success rate as a base.

Bekiroglu et al. (2019) defined a protocol for executing grasps under repeatable conditions in terms of objects and their placements. In addition, they proposed a success criterion for measuring the grasp robustness. Objects of the YCB dataset (CALLI et al., 2015) must be used in this benchmarking. The objects should be placed within the



Figure 2.18: (Left) M-BBT and T-BBT-block templates following the test template. (Right) BBT-100 blocks randomly placed inside a bin (MORGAN et al., 2019).

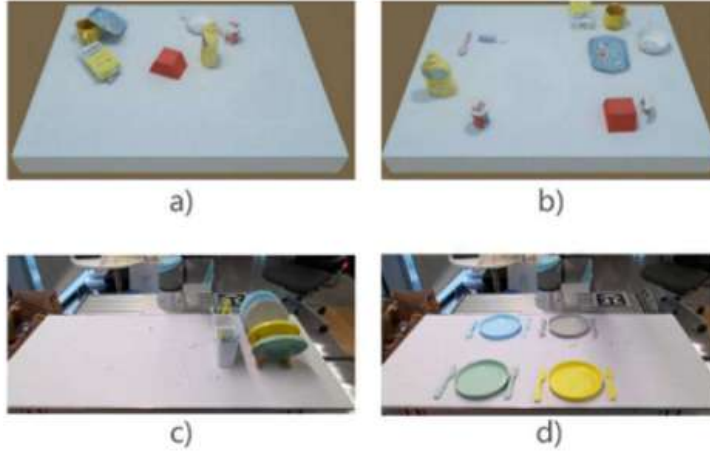


Figure 2.19: Samples of rearrangement tasks: initial scenes (a, c) and target scenes (b, d) (LIU et al., 2021).

robot’s reachable space. The robot must start the test with a 90 – 90 configuration as shown in Figure 2.20. r can be modified by the use for grasping large and heavy objects. The camera’s position is also optional. The object’s pose must respect the sequence of poses stated by the authors of the benchmark (Figure 2.21). The benchmark performed in Bekiroglu et al. (2019) includes lift, rotational, and shaking tests. Also, the time for grasping each object is considered.

Bottarel et al. (2020) proposed a set of printable layouts of predefined grasping scenarios equipped with localization markers to enhance test reproducibility. It was also proposed a protocol for testing the robot reachability and calibrating the vision system. The benchmark system also generates scores, grasp stability evaluation, and gives the possibility to evaluate the grasping algorithm using objects in isolation or in clutter. The objects of the YCB dataset (CALLI et al., 2015) are used. By using this benchmark, the reproducibility is significantly increased.

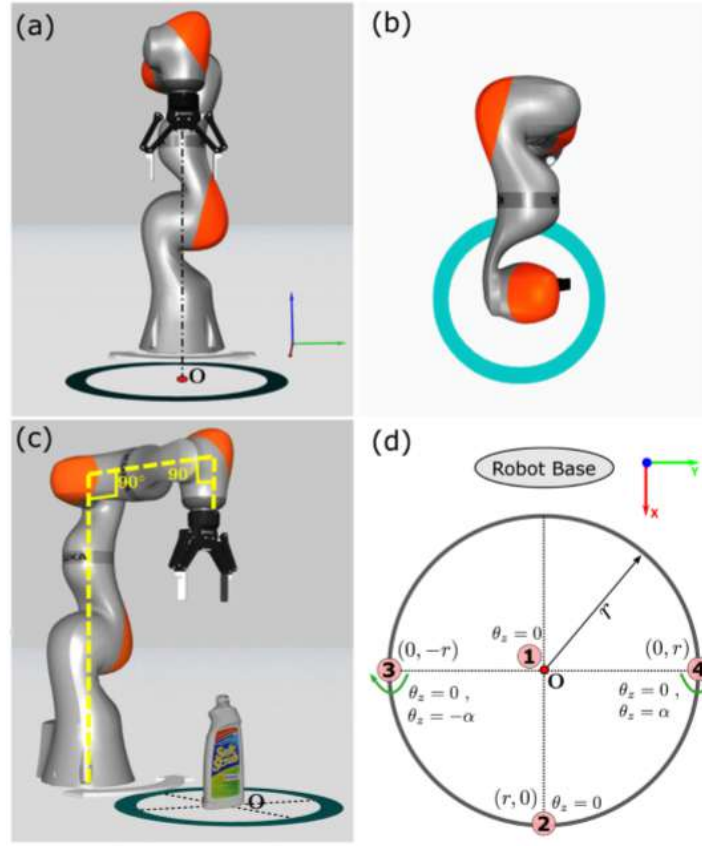


Figure 2.20: Grasping benchmark environment. The radius of the circular workspace is optional and must be reported by the authors. (BEKIROGLU et al., 2019).

Newbury et al. (2023), Fujita et al. (2020), and Calli et al. (2015), proposed the use of four grasping evaluation metrics:

- Grasping Success rate: Ratio between the number of objects successfully grasped and the initial number of objects in the bin;
- Mean Picks Per Hour (MPPH): Most used metric in logistics business for measuring human and robot grasping efficiency;
- Successful task executions over total attempts (SETA): Estimation of the success probability of a single grasp;
- Mean Time Between Failure (MTBF): Calculated by dividing the operating time by the failure count of the system;
- Mean Time To Repair (MTTR): Obtained by dividing the total time of failure by the failure count of the system;
- Availability: Measured by dividing $MTBF/(MTBF + MTTR)$; and

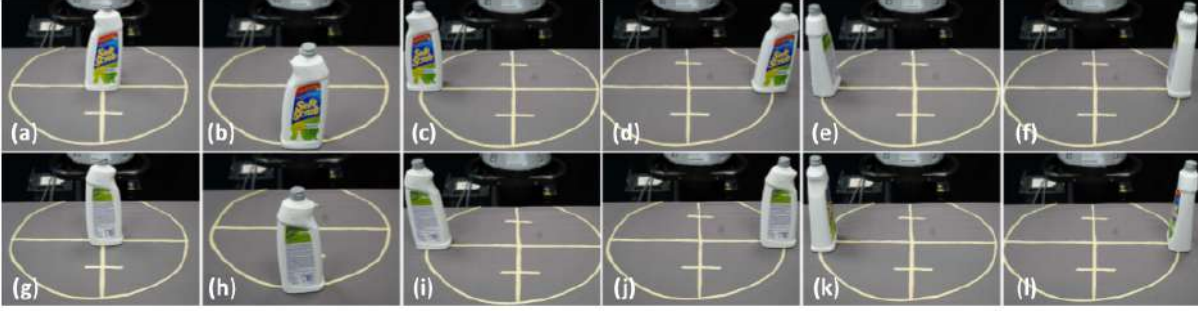


Figure 2.21: Object locations and poses considered by the benchmark in Bekiroglu et al. (2019).

- Completion/clearance rate: Percentage of objects that are removed from the clutter (structured or not).

In their study, Newbury et al. (2023) emphasize the importance of assessing the time efficiency of grasping techniques. To illustrate, situations may arise where opting for a quicker robot with a 75% success rate is more favorable than selecting a slower robot achieving 95% success, provided that occasional failed grasps do not yield significant drawbacks. Consequently, the authors allege that utilizing MPPH as the designated metric for this objective is reasonable. However, it should be noted that this metric's reliability is significantly influenced by factors such as robot hardware, trajectory planning, and computational resources.

Calli et al. (2015) recommends the application of the aforementioned metrics over three consecutive runs for each experiment. The average value of all three runs must be reported. The total failures that happen in each one of the phases, including pre-grasping, grasping, transport, and placement should be reported. Fujita et al. (2020) compared the aforementioned metrics against the score adopted in the Amazon Picking Challenge 2017. They studied the correlation between the team's performance and each metric. The goal was to find which technologies are important for the competition and future practical use.

According to Fujita et al. (2020), MPPH evaluates the task in a system level, since it also considers trajectory planning time besides the grasping algorithm. Although MPPH considers the average probability of success and the operation time, teams of the Amazon Picking Challenge 2017 that had a better operation time achieved better scores. By applying different metrics, it is possible to understand individually each aspect of the system.

When evaluating new grasping methods, authors are highly encouraged by the robotic grasping community to:

- Provide the lighting level (lux) of the environment (if the method is dependent on lighting) (MAHLER et al., 2018);
- Report the camera registration procedure, hardware and software used, and USB bandwidth (MAHLER et al., 2018);

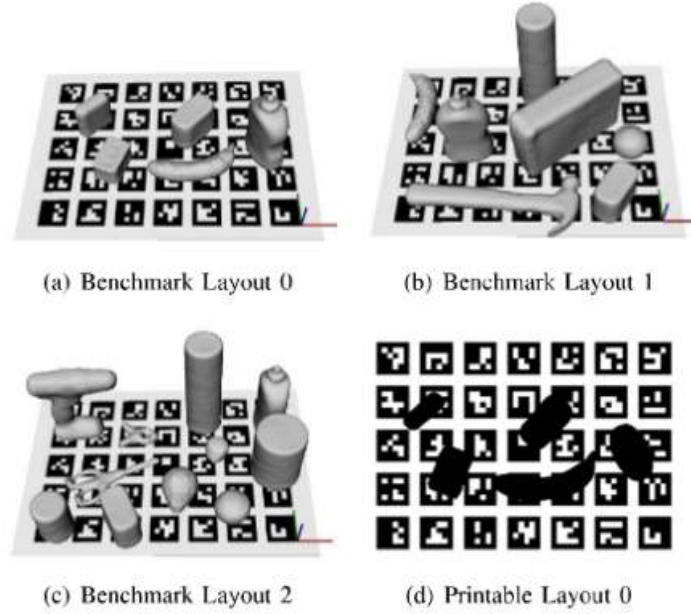


Figure 2.22: Layouts defined within the benchmark proposed by Bottarel et al. (2020).

- Make use of existing published datasets when possible (CALLI et al., 2021);
- Utilize a standard template that summarizes the protocol and benchmark (CALLI et al., 2021); and
- Provide multimedia files that illustrate and demonstrate the protocol (CALLI et al., 2021).

Depending on the grasp application, the experiments adopted for testing grasping method performance is usually divided into (NEWBURY et al., 2023; MORRISON; CORKE; LEITNER, 2018; VIERECK et al., 2017):

- Grasping individual objects, where only one object is grasped at a time;
- Grasping objects in a piled cluttered environment, where multiple objects are randomly placed in bins or boxes;
- Grasping objects in a structured environment, where multiple objects are spread out in a scene such that they are not touching each other;
- Grasping a specific class of objects such as household objects;
- Grasping objects with adversarial geometry;
- Grasping in open-loop. In this experiment, the objects are static. This method is also called one-shot grasp (VIERECK et al., 2017); and
- Grasping in closed-loop. In this experiment, the controller constantly updates the grasping pose in real-time (MORRISON; CORKE; LEITNER, 2020b).

2.6.2 Evaluation protocols adopted in simulated experiments

Johns, Leutenegger and Davison (2016) performed experiments in both simulated and real environments to investigate how well the training adapts to real data. A set of 1000 randomly selected object meshes from the ModelNet dataset (MND) (WU et al., 2015) was used for testing. Firstly, the object is placed on the surface at a random position and orientation within the camera’s field of view. The gripper is sent to the pose corresponding to the maximum grasp score in the depth image and a grasp attempt is performed. The Dynamic Animation and Robotics Toolkit (DART) (LEE et al., 2018) was employed in testing.

2.6.3 Evaluation protocols adopted in real experiments

Johns, Leutenegger and Davison (2016), Morrison, Corke and Leitner (2018), and Morrison, Corke and Leitner (2020b) fixed a robot on a table and rigidly mounted a camera onto the wrist of the robot. The camera was positioned perpendicular to a table at a fixed height.

Lenz, Lee and Saxena (2015) performed experiments in a static uncluttered environment. The gripper is firstly positioned 14 *cm* backward from the grasp position, and then move towards it. Then, the gripper is closed and moved 30 *cm* upwards. The experiments are performed in an open-loop. Gualtieri et al. (2016) used two statically mounted depth sensors to build a depth cloud of the object to be grasped. The gripper opening is restricted from 3 to 7 *cm* width. The objects are mixed in a box before being placed in the workspace and the robot performs the grasp until no objects are found in a bin. The experiments are performed in open-loop.

Johns, Leutenegger and Davison (2016) performed experiments in a static uncluttered environment. The objects were placed at five random positions and orientations within a graspable area in the robot workspace, but the way objects are placed into the robot’s workspace was not informed. The distance between the gripper fingers was kept fixed at 10 *cm* before the gripper is closed at the grasp pose. The experiments were only performed in open-loop control since the grasp is not updated over the path to the grasp pose.

Mahler et al. (2017) conducted experiments in a static environment with individual or cluttered objects using an open-loop controller. The camera is fixed in the environment. Before each grasping experiment, a human operator shakes the objects in a box and places them on the workspace.

Morrison, Corke and Leitner (2018) performed experiments in a static environment with individual objects and a dynamic environment with cluttered objects. Since the camera does not acquire depth measurements from a distance closer than 150 *mm*, the target grasp is not updated after this distance. Besides that, the depth camera used is unable to provide any valid depth data on many black or reflective objects. The gripper used has a maximum stroke of 175 *mm* but it was limited to 70 *mm* to increase the grasp precision. Before placing the objects in the workspace, they are shaken in a cardboard box to remove the bias related to the object pose. The grasping experiments were divided into open-loop and closed-loop grasping. The closed-loop grasping was dynamically executed by manually moving a sheet where objects are placed. In the open-loop grasp experiment,

the objects are grasped while kept fixed in the environment.

Mahler et al. (2019) executed more than 2500 grasp attempts on a physical robot system with a parallel-jaw and vacuum gripper. 50 test objects were used in the experiments. In the experiments, all objects are placed in a bin and the grasp is performed until the bin is empty or the number of grasp attempts reached the limit. A fixed 3D depth camera was used. Morrison, Corke and Leitner (2019) validated the approach using a Franka Emika Panda robot equipped with a custom 3D-printed gripper finger. The Intel Realsense D435 depth camera was used. These experiments consider cluttered objects in a bin. In each grasping experiment, 20 objects are chosen at random, and the robot performs the grasps until all objects have been removed.

Morrison, Corke and Leitner (2020b) is an extension of the research presented in Morrison, Corke and Leitner (2018). The objects are shaken in a cardboard box before placing them into the robot’s workspace in order to avoid bias associated with the grasping attempt in individual or cluttered scenarios. In the open-loop experiments, the camera is positioned 350 *mm* above and orthogonal to the planar surface. The one-shot grasping is performed in this case. The robot moves to a pre-grasp position, 170 *mm* above the surface. From this position, the gripper moves down slowly to grasp the object while closing the gripper and stopping the movement if any unexpected collision is detected. In closed-loop experiments, the camera is positioned at 400 *mm* above the surface and the objects are manually moved using labels as a reference of the start and final position of the movements.

2.6.4 Definition of grasping success

In this research, DSG refers to a method to determine if the grasp performed was successful or not. It is important to emphasize that authors commonly adopt only one DSG to evaluate their grasping algorithms.

Each DS is referenced as its related number from this section. In performance evaluations, the grasp is considered successful if:

DSG 1: The object is lifted to a certain distance from its initial position (JOHNS; LEUTENEGGER; DAVISON, 2016; PINTO; GUPTA, 2016);

DSG 2: The object is lifted to the gripper start position (MORRISON; CORKE; LEITNER, 2018);

DSG 3: The angle difference between the predicted rectangle and ground-truth grasp rectangle is less than 30° and the IoU is greater than 25%. This DSG is commonly referred as IoU (RIBEIRO; GRASSI, 2019; MORRISON; CORKE; LEITNER, 2020b);

DSG 4: The object is lifted to a predefined position, shaken, and does not fall (MAHLER et al., 2017).

DSG 5: The object is lifted to a predefined position and held for one second. (LENZ; LEE; SAXENA, 2015).

DSG 6: The object is transported to the bin on the side of the workspace (MAHLER et al., 2019).

The objects used in the experiments are commonly divided into two categories:

1. **Known Objects:** Objects used in training (known by the network);
2. **Unknown Objects:** Objects not used in training (unknown by the network).

2.6.5 Grasping performance

In their study, Lenz, Lee and Saxena (2015) observed a notable performance increase of 9% when incorporating a regularization function during training. Experiments demonstrated that integrating multimodal data, including depth measurements, surface normals, and RGB images of the objects, during training significantly enhanced grasping performance. A success rate of 89% was achieved considering DSG 5. Despite these performance improvements, it’s worth noting that a single grasp took 13.5 s to be generated

In their work, Johns, Leutenegger and Davison (2016) took into account uncertainties when generating grasps. This approach proved to be more effective than the grasp method based on the object’s centroid. This is because the centroid may not always be the optimal point for picking up a complex object, such as a toy gun. The grasp method achieved an impressive 80.0% success rate in a static, non-cluttered environment, specifically in DSG 1.

Pinto and Gupta (2016) performed experiments using objects that have not been seen in the training. 3 *k* grasps attempts were executed in real experiments. They achieved a success rate of 79.3% for unknown objects considering DSG 1. In the experiments, 150 grasp trials were performed.

Gualtieri et al. (2016) performed two different experiments: one experiment was performed with a camera fixed in the environment (passive scenario) and the second with the camera positioned on the robot’s wrist (active scenario). A success rate of 84% was achieved for the passive scenario and 93% for the active scenario. Most of the failures are related to perception errors caused by the grasping detection algorithm. Nonetheless, 90% of the objects were cleared from the bin.

Mahler et al. (2017) presented a new grasping method, denoted GQ-CNN. By using DSG 4, the network performance achieved a success rate of 93% when evaluated with unknown objects. The entire grasping pipeline took 0.8 s to plan a new grasp.

Morrison, Corke and Leitner (2018) presented a new grasping method, denoted GG-CNN, that takes 19 *ms* to generate a new grasp considering the entire grasping pipeline. The network inference takes 2.1 – 6 *ms* to compute a new grasp. Firstly, an experiment was performed in a static environment using individual objects considering DSG 2. The grasps achieved a success rate of 84% for the open-loop case and 81% for the closed-loop case. The adversarial objects from Mahler et al. (2017) (Figure 2.12g) were used in the experiments. Secondly, an experiment was performed in a dynamic environment using individual objects. A grasp success rate of 83% was obtained for the adversarial objects and 88% for the household objects. Finally, an experiment was performed in a cluttered

dynamic environment using multiple objects (Figure 2.13b). A success rate of 81% was obtained for the household objects.

Mahler et al. (2019) evaluated the performance of the GQ-CNN in a new dataset (Dex-Net 4.0). A success rate of 95% was achieved on a physical robot with 300 MPPH in clutter. They achieved a success rate of 97% on a set of household objects. Experimental results showed an improvement when considering ambidextrous grasping. The grasping reliability increased from 80% to 95%. The experiments were evaluated using DSG 6.

Morrison, Corke and Leitner (2019) achieved a success rate of 80% in grasping from clutter considering a multi-view approach. The grasp was evaluated using DSG 6. It was proved that the success rate and time of the grasp depend on the trajectory performed by the robot preceding the grasping action. The method presented achieved a 282 MPPH.

Morrison, Corke and Leitner (2020a) evaluated the performance of the GG-CNN in a new dataset EGAD created specifically for robotic grasping. The performance decreased significantly compared to the results found in Morrison, Corke and Leitner (2018). They accomplished a success rate of 69% for simple objects and 40% for complex objects using DSG 2. This is expected since the test objects from the EGAD dataset provide a higher level of grasping complexity.

Morrison, Corke and Leitner (2020b) presented a new version of the GG-CNN called Generative Grasping Convolutional Neural Network 2 (GG-CNN2). The GG-CNN2 takes only 20 *ms* to generate a new grasp, considering the entire grasping pipeline. The network inference takes only 3 *ms* to compute a new grasp. Regardless of the training set used, the GG-CNN2 outperforms the GG-CNN when evaluated using SGTs. The GG-CNN2 performance was evaluated using only open-loop control through IoU, SGTs and real experiments. The network achieved a success rate of 85% (DSG 2) and 84% (DSG 3). When applying data augmentation to this dataset, the grasping success rate decreased to 79% (DSG 2) and 82% (DSG 3). The experiment performed on a real robot achieved 94% on household objects and 84% on adversarial objects considering DSG 2.

Sun et al. (2023) proposed a new grasp detection based on single-view local point clouds. It introduces a three-stage process consisting of the Candidate Generation Network (CGN), Reliable Adjustment Module (RAM), and Quality Assessment Network (QAN) to predict the graspability and the initial grasp pose of sampled points based on features extracted from local sphere regions, leveraging an improved version of PointNet for feature extraction. The network was trained using real world data acquired from two RGBD cameras. Due to the inference time of 2.58s for 1024 points, the method can only be used in open loop. Experiments performed on real robots reported 88.3% of success rate for single object scenes and 83.0% for cluttered scenes.

Jauhri, Lunawat and Chalvatzaki (2023) introduced a grasping method called Neu-GraspNet. This network was trained in simulation with piled and packed upright objects on a table. The dataset comprises 343 objects split into 303 training and 40 testing objects. The training process employed a total of 1.4 million grasps in 33,313 scenes for pile scenarios and 1.2 million grasps in 33,534 scenes for packed scenarios. The dataset comprises both successful and unsuccessful grasps and utilizes 100,000 occupancy points per scene for training scene reconstruction. In pile scenes, the Grasp Success Rate (GSR) ranged from 73.95% to 86.51%. In packed scenes, the GSR ranged from 78.76% to 97.65%.

Sun et al. (2023) tested the proposed grasping synthesis algorithm using a real hardware. A structured pile of 8 to 10 objects was used to evaluate the performance of the algorithm. The algorithm achieved a success rate of 88.3% of the average success rate for single object scenes and 83.0% for cluttered scenes.

2.6.6 Preliminary Considerations

An overview of the experimental procedures, operating system, and hardware adopted by each related work is presented in Table 2.4.

Table 2.4: Experimental procedures, operating system, and hardware (graphics card, robot, gripper and camera), adopted by each related work.

	Simulated Experiment	Real Experiment	Camera's Position	Operating System	Robot	Gripper	Camera	Graphic Card
Sun et al. (2023)		✓	Gripper		Kinova Gen 3	Robots-2q	Kinect v2	RTX 3080Ti
Morrison, Corke and Leitner (2020a)	✓	✓	Gripper	Ubuntu 16.04	Kinova MICO	Kinova KG-2	Realsense SR300	GTX 1070
Morrison, Corke and Leitner (2019)		✓	Gripper		Franka Emika Panda	Custom	Realsense D435	
Mahler et al. (2019)	✓	✓	Fixed	Ubuntu 16.04	ABB YuMi	Custom	Photoneo PhoXi S industrial 3D	Titan XP
Ribeiro and Grassi (2019)				Ubuntu 16.04				GTX 1050 Ti
Morrison, Corke and Leitner (2018)		✓	Gripper	Ubuntu 16.04	Kinova MICO	Kinova KG-2	Realsense SR300	GTX 1070
Mahler et al. (2017)		✓	Fixed	Ubuntu 14.04	ABB YuMi	Custom	Primesense Carmine 1.08	GTX 1080
Johns, Leutenegger and Davison (2016)	✓	✓	Gripper		Kinova MICO	Kinova KG-2	Primesense Carmine 1.09	
Pinto and Gupta (2016)		✓						
Gualtieri et al. (2016)		✓	Fixed		Baxter Research Robot	Stock Baxter Parallel-jaw	Asus Xtion Pro	GTX 660
Lenz, Lee and Saxena (2015)		✓						

Some approaches limit the maximum grasp width at a specific value and state that this would improve the grasp precision (MORRISON; CORKE; LEITNER, 2020b, 2018). Nevertheless, it can restrict the type and size of objects used in real experiments. Besides that, it can be used as a shortcut to avoid a collision while grasping objects in clutter or placed in a bin. Other grasping techniques dynamically change the grasp width over the path to the grasp pose considering the network inference (MORRISON; CORKE; LEITNER, 2020b, 2018). Therefore, possible collisions with the surrounding objects may be avoided.

When the grasping is performed in closed-loop, some grasping methods stop updating the camera's depth measurements and the robot usually performs a blind grasp from this position (MORRISON; CORKE; LEITNER, 2020b, 2018). Therefore, it can be dangerous if the object continues moving after the depth measurements shutdown. The depth measurement limitation could be solved by a fixed camera position in the environment (MAHLER et al., 2017), but occlusions could easily occur due to the arm position or a pile of objects in clutter. Consequently, the minimum camera depth measurement is an important parameter that must be considered in dynamic grasps.

Testing the grasping performance through real experiments is crucial to determine the efficiency of the grasping method (MORRISON; CORKE; LEITNER, 2020a). To evaluate the grasping performance, different DSGs are employed. DSG 1 is sometimes applied for a quick offline evaluation, whereas other time-consuming DSGs are still used such as DSG 2 (MORRISON; CORKE; LEITNER, 2020b). DSG 3 (IoU) is susceptible

to false negatives because no simulation is performed. In this case, the grasp is considered successful only if a ground-truth label exists in the dataset.

Besides that, only the average of physical success grasps over a set of objects are reported. Nevertheless, it does not allow easy identification of the limitation of the grasping technique proposed. This leads authors to detail the failure modes (MORRISON; CORKE; LEITNER, 2020b).

Small network architectures have proved to be efficient enough to perform grasp using closed-loop control even with an intermediate hardware (MORRISON; CORKE; LEITNER, 2018, 2020b). The benefit of using closed-loop controls is to perform accurate grasps regardless of inaccurate hardware. DSG 3 was commonly employed to test grasping method performances (RIBEIRO; GRASSI, 2019). Nonetheless, without a real or simulated environment, it is difficult to validate the performance reported.

Table 2.5 shows the performance achieved with unknown objects. This table is organized according to the test object used in the experiments such as adversarial objects (MAHLER et al., 2017), household objects, or the objects proposed by (VIERECK et al., 2017). In this table, the unfilled cells mean that the authors did not provide any related information. It also shows if the evaluation was performed in real or simulated environments. Comparisons cannot be made directly since the objects used in these researches differ substantially from each other, so the table is for indicative purposes only.

Table 2.5: Comparative table containing the average success rate, number of parameters, and processing time to generate grasp poses using unknown objects. These objects are divided in Adversarial Objects (A.O.) (MAHLER et al., 2017), Household Objects (H.O.), and objects proposed by Viereck et al. (2017) (V.). The type refers to a real (R) or simulated (S) evaluation environment. The results are divided into open-loop (O.L.), closed-loop (C.L.), and according to the DSGs used in subsection 2.6.4.

				Grasping Performance on Unknown Objects											
				Static Individual		Static Cluttered		Dynamic Individual		Dynamic Cluttered		Param Number	MPPH	Network Inference Time	Entire Grasping Pipeline
	Objects	Type	Metric	C.L.	O.L.	C.L.	O.L.	C.L.	C.L.						
Morrison, Corke and Leitner (2020a)	EGAD	R	2			69%								2.1 – 6ms	19ms
Morrison, Corke and Leitner (2020b)	A.O.	R	1					84%				66k		3ms	20ms
Morrison, Corke and Leitner (2019)	A.O.	R	6					80%				62k	282	2.1 – 6ms	19ms
Morrison, Corke and Leitner (2018)	A.O.	R	2	81%	84%			83%				62k		2.1 – 6ms	19ms
Jauhri et al. (2023)	H.O.	R	2					97.7%						5.5s	
Sun et al. (2023)	H.O.	R	2		88.3%			83%						2.58s	
Morrison, Corke and Leitner (2020b)	H.O.	R	2					94%				66k		3ms	20ms
Mahler et al. (2019)	H.O.	R	6					97%					300		
Morrison, Corke and Leitner (2018)	H.O.	R	2	91%	92%							62k		2.1 – 6ms	19ms
Mahler et al. (2017)	H.O.	R	4					80%				18M			0.8s
Pas et al. (2017)	H.O.	R						93%							0.3 – 6.2s
Levine et al. (2016)	H.O.	R			80%							1M			0.2 – 0.5s
Pinto and Gupta (2016)	H.O.	R	1					79.5%				18M			
Johns, Leutenegger and Davison (2016)	H.O.	R	1		80%							7.3M			
Gualtieri et al. (2016)	H.O.	R						93%						3s	
Lenz, Lee and Saxena (2015)	H.O.	R	5		89%										13.5s
Morrison, Corke and Leitner (2018)	V	R	2	100%	87%					81%		62k		2.1 – 6ms	19ms
Viereck et al. (2017)	V	R		98%	89%					77%					0.2s

Comparisons cannot be made directly since the objects used in these research differ substantially from each other, so the table is for indicative purposes only.

2.7 CONCLUSION

Object features such as the center of mass, friction, or geometry are unknown in unstructured environments. Therefore, real robotic grasping applications that depend on these parameters do not perform well in practice (MAHLER et al., 2018). Analytic methods that require a partial or complete knowledge of known objects are also avoided since they cannot generalize well to unknown objects (MORRISON; CORKE; LEITNER, 2020a; MAHLER et al., 2019).

Note that authors avoid performing experiments only in the simulation since it cannot fairly model the reality. It is easy to observe given that the results significantly differ in simulation and real experiments.

Robotic grasps are commonly performed by fixing the camera on the robot’s wrist. Gualtieri et al. (2016) compared the active scenario (camera on robot’s wrist) and the passive scenario (camera fixed in the environment). They concluded that the active scenario outperforms the passive scenario. The reason is that a better view of the objects can be obtained without occlusion of the robot’s arm and different points of view can be obtained during the grasp.

It is recommended to have the same set of test objects to get a comparable average success rate. Some objects commonly used in real experiments for evaluation consist of simple geometry. Therefore, they do not require a precise grasp. To offer a more challenging robotic grasping environment, Morrison, Corke and Leitner (2020a) proposed a set of test objects called EGAD dataset. It has complex shapes and can be easily 3D printed.

The grasp is usually performed in individual objects rather than cluttered objects. It does not always correspond to the real world since most applications in robotics consider a cluttered environment such as warehouse and order fulfillment. Depending on the target application such as picking objects on a non-stop conveyor belt includes a dynamic environment. In this case, closed-loop control is required, but little research has achieved high performance considering this environment (MORRISON; CORKE; LEITNER, 2018).

Since most of the related work uses deep learning approaches for robotic grasping, it is a common practice to provide the code and the training data for performance comparison (PAS et al., 2017; MAHLER et al., 2019; MORRISON; CORKE; LEITNER, 2018). Table 2.6 summarizes important aspects of each related work.

Table 2.6: Comparative table of the related work.

	Real experiments	Grasp in 6DOF	Camera on gripper	Dataset Objects	Complex Objects	Cluttered Environment	Dynamic Environment	Closed loop	Data available	Multivision	Trajectory Planner
Sun et al. (2023)	✓	✓	✓	✓		✓				✓	
Jauhri, Lunawat and Chalvatzaki (2023)	✓	✓		✓					✓		
Ribeiro, Mendes and Grassi (2021)	✓		✓	✓			✓	✓	✓		
Morrison, Corke and Leitner (2020b)	✓		✓	✓	✓	✓		✓	✓		
Morrison, Corke and Leitner (2020a)	✓		✓	✓	✓	✓	✓	✓	✓		
Morrison, Corke and Leitner (2019)	✓		✓	✓	✓	✓			✓	✓	
Mahler et al. (2019)	✓	✓		✓	✓	✓			✓		
Ribeiro and Grassi (2019)				✓							
Morrison, Corke and Leitner (2018)	✓		✓	✓	✓	✓	✓	✓	✓		
Pas et al. (2017)	✓		✓	✓		✓			✓	✓	TrajOpt (SCHULMAN et al., 2013)
Viereck et al. (2017)	✓		✓			✓	✓	✓			IKFast (DIANKOV; KUFFNER, 2008)
Mahler et al. (2017)	✓				✓				✓		
Pinto and Gupta (2016)	✓					✓					EST (HSU; LATOMBE; MOTWANI, 1997)
Gualtieri et al. (2016)	✓	✓	✓	✓		✓					TrajOpt (SCHULMAN et al., 2013)
Levine et al. (2016)	✓					✓		✓	✓		
Johns, Leutenegger and Davison (2016)	✓		✓								
Lenz, Lee and Saxena (2015)	✓			✓					✓		

Dataset Objects: The object used for testing the grasping method are available for download.

Complex Objects: the objects used for testing are more complex than boxes, cylinders, and bowls and have an adversarial geometry.

Cluttered Environment: The objects used for testing are too close together or on top of each other.

Data available: Code and dataset available for access.

Multivision: Techniques that explore the robot workspace from more than one view to generate a grasp.

Chapter 3

This chapter presents a two-step cascaded grasping system. It is comprised of the grasp generator denoted Generative Grasping Convolutional Neural Network (GG-CNN) and a modified version of the Single Shot Multibox Detector architecture (SSD). The integration of the object detection network to the grasp generator allows the robot to select the object to be grasped.

SELECTIVE PLANAR ROBOTIC GRASPING USING CONVOLUTIONAL NEURAL NETWORKS

3.1 INTRODUCTION

Robotic grasping techniques have achieved the next level in terms of performance in a real environment thanks to the advances in Artificial Intelligence and Computer Vision techniques. Recent researches have proved that letting the system build a grasp function itself instead of developing an analytical model is promising (MORRISON; CORKE; LEITNER, 2019; MAHLER et al., 2018; SATISH; MAHLER; GOLDBERG, 2019).

Morrison, Corke and Leitner (2019) developed an object-agnostic grasping method called GG-CNN well-known for its good performance in open-loop and closed-loop systems. Its not possible to select an specific object to grasp using GG-CNN because it generates grasps based on a quality prediction of an input depth image considering all the objects in the workspace. The object closer to the camera is generally preferred to be grasped by the network inference. This is not desired since the robot may grasp a fixed object such as a bin instead of movable objects, as shown in Figure 3.1.

In order to mitigate the problem shown in Figure 3.1 and also recorded in video¹, the Single Shot Multibox Detector (SSD) was implemented. The SSD is coupled with the GG-CNN to create a grasping system capable of choosing the object to perform the grasp. Furthermore, the GG-CNN input is a fixed area in a depth image of 300×300 . Consequently, if the depth sensor has a higher resolution, a grasp cannot be generated for the objects outside this area (Figure 3.2). Since the actual and target position and orientation of the gripper is known, and no obstacle avoidance is considered in the trajectory to the gripper pose, a quintic polynomial trajectory planner was applied to align the object in the workspace with the GG-CNN area.

¹<https://www.youtube.com/watch?v=teXbXgisPew>

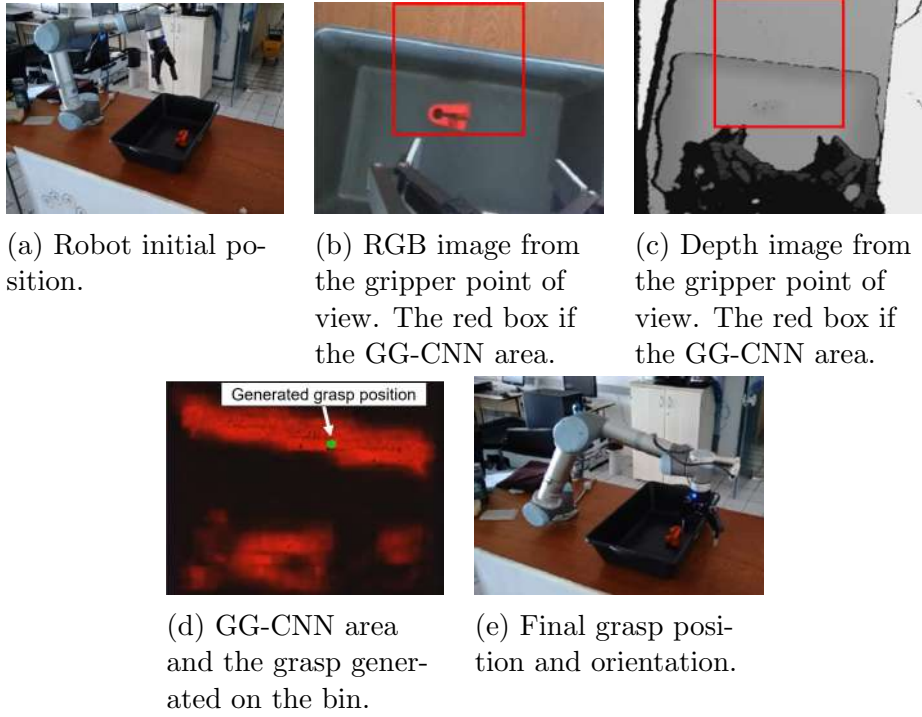


Figure 3.1: Since GG-CNN cannot distinguish between fixed and movable objects, a grasp can be generated on fixed objects such as a bin. To generate a grasp, GG-CNN uses only the area inside the red box, represented in (b) and (c). This experiment was conducted in the Laboratory of Robotics (LaR) at UFBA and is available at youtube.com/watch?v=tebXgisPew.



Figure 3.2: Using the GG-CNN, a grasp is not generated for objects outside the GG-CNN area (red box).

This chapter is organized as follows: Section 3.3 specifies the problem which is tried to be solved and define the variables used in this chapter, Section 3.4 introduces the GG-CNN, Section 3.7 describes the SSD used for object detection, Section 3.8 explains the grasping pipeline, Section 3.9 depicts the hardware and software used in the experiments, Section 3.10 describes the pre-processing and post-processing applied to the input

and output of the GG-CNN, Section 3.11 details the experiments performed using the GG-CNN and the object recognition networks used in this research, and Section 3.12 concludes this chapter.

3.2 CONTRIBUTIONS

This chapter has the following main contributions:

- Development of a 4D grasping pipeline using a Convolutional Neural Network Grasp Generator and an Object Detection Network to selectively grasp objects on a table-top scenario using an RGB+D sensor;
- Validation of the proposed system using an UR5 Robot Arm Manipulator, an RGB+D camera Intel Realsense D435, and the gripper Robotiq 2F-140.

3.3 PROBLEM STATEMENT

In this chapter, a selective 4D grasping pipeline using an RGB+D sensor is proposed. The RGB+D sensor is used because the object is recognized using an RGB image and the grasp is generated using a Depth image. It performs antipodal grasps on objects of interest.

3.3.1 Assumptions

It is assumed that the objects are located on a flat surface, and the grasping is performed by a parallel-jaw gripper and a camera mounted in the robot's wrist. All intrinsic camera parameters are known. The grasp is defined by the position and orientation of the gripper with known geometry, as shown in Figure 3.3. The parameters associated with the grasping, used in that figure and the rest of the work can be found in Subsection 3.3.2.

3.3.2 Definitions

Definition 3.3.1. Let $\mathbf{I}_0 = \mathbb{R}^{H \times W}$ describe an 2.5D depth image in which no object is considered. H and W represent the height and the width of this image, respectively.

Definition 3.3.2. Let $\mathbf{I} = \mathbb{R}^{H \times W}$ be an 2.5D depth image in which every object in the environment is considered. H and W represent the height and the width of this image, respectively, and the pixel value is the depth measurement.

Definition 3.3.3. Let $\mathbf{I}_f = \mathbb{R}^{H \times W}$ denote a filtered 2.5D depth image, in which every object of interest is considered. H and W represent the height and the width of this image, respectively.

Definition 3.3.4. Let $\mathbf{C}$ define the robot base coordinate frame.

Definition 3.3.5. Let $\mathbf{g} = (\mathbf{p}, \phi, \omega, q)$ represents the grasp in $\mathbf{C}$, where $\mathbf{p} = (x, y, z)$ describes the gripper's center point on the object, ϕ denotes the gripper angle around z axis, ω describes the gripper width and q represents the grasp quality obtained from one of the outputs of the GG-CNN.

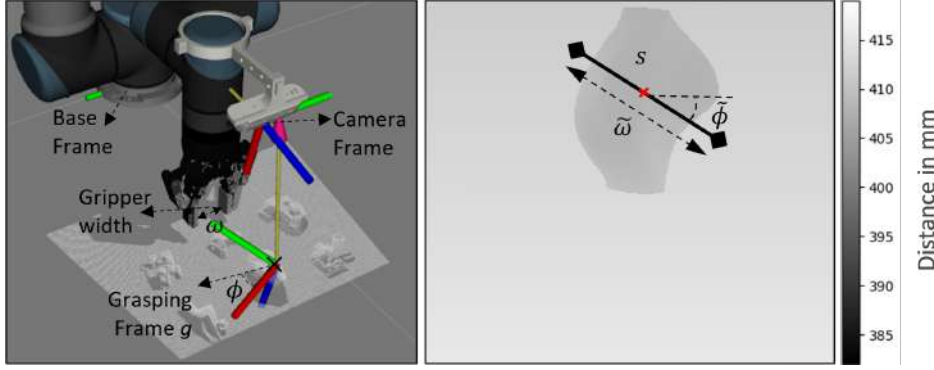


Figure 3.3: (Left) Grasp coordinate frame representation $\mathbf{g} = (\mathbf{p}, \phi, \omega, q)$ and the other frames related to the camera and robot. $\mathbf{p} = (x, y, z)$ describes the gripper's center point on the object, ϕ denotes the gripper angle around z axis, ω describes the gripper width and q represents the grasp quality (Right) Representation of $\tilde{\mathbf{g}} = (\mathbf{s}, \tilde{\phi}, \tilde{\omega}, q)$ on the depth image $\mathbf{I}$, where $\mathbf{s}$ indicates the gripper's center point in pixels, $\tilde{\phi}$ denotes the gripper angle and $\tilde{\omega}$ refers to the gripper width.

Definition 3.3.6. Let $\tilde{\mathbf{g}} = (\mathbf{s}, \tilde{\phi}, \tilde{\omega}, q)$ be the grasp in $\mathbf{I}$, where $\mathbf{s}$ indicates the gripper's center point in pixels, $\tilde{\phi}$ denotes the gripper angle and $\tilde{\omega}$ refers to the gripper width.

Definition 3.3.7. Let $\tilde{\mathbf{G}} = (\tilde{\Phi}, \tilde{\mathbf{W}}, \tilde{\mathbf{Q}}) \in \mathbb{R}^{H \times W \times 3}$ describes a set of grasps $\tilde{\mathbf{g}}$ in $\mathbf{I}$, where $\tilde{\Phi}$, $\tilde{\mathbf{W}}$ and $\tilde{\mathbf{Q}}$ consist of values of $\tilde{\phi}$, $\tilde{\omega}$ and q for each pixel.

Definition 3.3.8. Let $M(\mathbf{I}) = \tilde{\mathbf{G}}$ describes a grasp function that generates a grasp map $\tilde{\mathbf{G}}$ from a depth image $\mathbf{I}$.

Definition 3.3.9. Let $M_\theta(\mathbf{I})$ describes an approximation of the grasp function by means of a CNN, where θ indicates the network weights.

Definition 3.3.10. Let $\tilde{\mathbf{G}}_\theta = (\tilde{\Phi}_\theta, \tilde{\mathbf{W}}_\theta, \tilde{\mathbf{Q}}_\theta) \in \mathbb{R}^{H \times W \times 3}$ describes a set of estimated grasps $\tilde{\mathbf{G}}$ in $\mathbf{I}$, where $\tilde{\Phi}_\theta$, $\tilde{\mathbf{W}}_\theta$ and $\tilde{\mathbf{Q}}_\theta$ consist of the approximated values of $\tilde{\phi}$, $\tilde{\omega}$ and q for each pixel. H and W represent the height and the width of this image, respectively.

Definition 3.3.11. Let $\tilde{\mathbf{G}}_f = (\tilde{\Phi}_f, \tilde{\mathbf{W}}_f, \tilde{\mathbf{Q}}_f) \in \mathbb{R}^{H \times W \times 3}$ denotes an approximated grasp function in $\mathbf{I}_f$, where $\tilde{\Phi}_f$, $\tilde{\mathbf{W}}_f$ and $\tilde{\mathbf{Q}}_f$ consist of the approximated values of $\tilde{\phi}$, $\tilde{\omega}$ and q for each pixel. H and W represent the height and the width of this image, respectively.

Definition 3.3.12. Let $\mathbf{g}_\theta^*$ refers to the best grasp visible in $\mathbf{C}$.

Definition 3.3.13. Let $\tilde{\mathbf{g}}_\theta^*$ be the best visible grasp in $\mathbf{I}$, as a result of $\tilde{\mathbf{G}}_\theta$.

Definition 3.3.14. Let $\tilde{\mathbf{g}}_f^*$ indicates the best visible grasp in $\mathbf{I}_f$, from $\tilde{\mathbf{G}}_f$.

Definition 3.3.15. Let $\tilde{\mathbf{I}}_T$ describes a ground truth depth image from a grasping dataset.

Definition 3.3.16. Let $\tilde{\mathbf{G}}_T$ denotes a ground truth grasp map generated from a ground truth depth image $\tilde{\mathbf{I}}_T$.

Definition 3.3.17. Let $\tilde{Q}_T$ refers to a ground truth quality image.

Definition 3.3.18. Let $\tilde{\Phi}_T$ be a ground truth grasp angle.

Definition 3.3.19. Let $\tilde{W}_T$ denotes a ground truth gripper width.

3.4 GENERATIVE GRASPING CNN

3.4.1 Training dataset and related processes

The Cornell Grasping Dataset (LENZ; LEE; SAXENA, 2015) is used for training the GG-CNN. As mentioned in Section 2.2.1, the Cornell Grasping Dataset has 885 depth images of real objects available and 5110 human-labeled positive and 2909 negative grasps. This dataset has the advantage of having several labeled grasps available per image (Figure 3.4). The Cornell Dataset represents antipodal grasps as rectangles using pixel coordinates, aligned to the position and rotation of a gripper as in Jiang, Moseson and Saxena (2011).

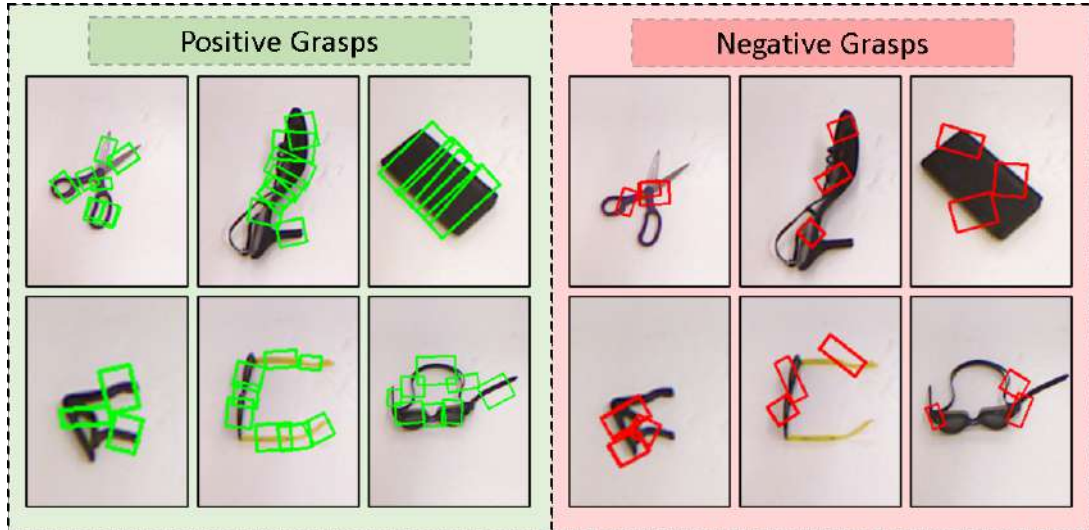


Figure 3.4: Cornell Grasping Dataset (LENZ; LEE; SAXENA, 2015) positive and negative grasps represented as rectangles as in Jiang, Moseson and Saxena (2011).

From each grasp rectangle in Figure 3.4, it is possible to extract the gripper width ω , the tool center point p , and the planar gripper orientation ϕ (Figure 3.9a). Since the antipodal grasp is symmetrical around the y axis, the gripper orientation ϕ is constrained in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ (Figure 3.9b). Rather than considering the gripper orientation as an angle, Hara, Vemulapalli and Chellappa (2017) found that representing the gripper orientation as two components of a unit vector makes the training procedure more effective. This procedure is also adopted by Morrison, Corke and Leitner (2018). However, representing the angle as two components of a unit vector $\sin(\tilde{\Phi}_T)$ and $\cos(\tilde{\Phi}_T)$ (Figure 3.5a) causes a discontinuity around $\pm\frac{\pi}{2}$, as shown in Figure 3.5b, which is particularly challenging when employing AI techniques such as backpropagation that rely on smooth, continuous functions for effective learning and convergence.

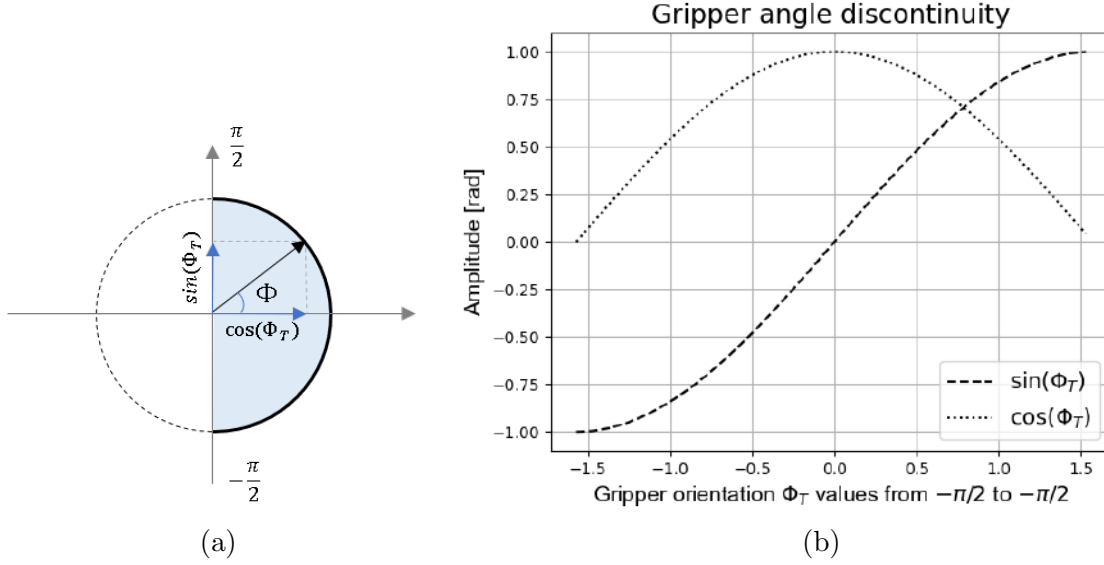


Figure 3.5: (a) Grasp angle representation considering $\cos(\Phi_T) > 0$ (b) grasp angle discontinuity around $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.

In order to avoid this problem, the two components of a unit vector are defined as $\sin(2\tilde{\Phi}_T)$ and $\cos(2\tilde{\Phi}_T)$. It removes the discontinuity around $\pm\frac{\pi}{2}$ and provides a unique representation for the grasp rectangle when the gripper angle is at $-\frac{\pi}{2}$ or $\frac{\pi}{2}$ (Figure 3.6). Morrison, Corke and Leitner (2018) augmented the Cornell Grasping Dataset using random zooms, crops and rotations to generate a set of 8.840 depth images $\tilde{\mathbf{I}}_T$. Each depth image was associated to grasp maps $\tilde{\mathbf{G}}_T$, integrating 51.100 grasp samples in total. Since the Cornell Grasping Dataset provides multiple ground truths per object, a higher estimation of $\tilde{\mathbf{G}}$ is obtained if compared to datasets that only provide a single grasp per image (MAHLER et al., 2017).

3.4.2 Network architecture

The GG-CNN provides an approximation of the complex function $M(\mathbf{I})$. It is proved through experiments that $M_\theta(\mathbf{I}) = \tilde{\mathbf{G}}_\theta = (\tilde{\Phi}_\theta, \tilde{\mathbf{W}}_\theta, \tilde{\mathbf{Q}}_\theta)$ can be learned by means of the input $\tilde{\mathbf{I}}_T$ and the corresponding output $\tilde{\mathbf{G}}_{Tr}$ (MORRISON; CORKE; LEITNER, 2018). It is accomplished by using a set of training data, and applying the cost function L2-norm, represented by $\mathcal{L}$, so that:

$$\theta = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\tilde{\mathbf{G}}_{Tr}, M_\theta(\tilde{\mathbf{I}}_T)) \quad (3.1)$$

Rather than estimate a single grasp from the depth image $\mathbf{I}$, a single grasp is generated on each pixel of $\mathbf{I}$. Therefore, a map of grasps $\tilde{\mathbf{G}}_\theta$ corresponding to the width and height of the image $\mathbf{I}$ is created. GG-CNN comprises a fully CNN architecture since it has proven to be efficient in image segmentation techniques (BADRINARAYANAN; KENDALL; CIPOLLA, 2017; LONG; SHELHAMER; DARRELL, 2015) and contour

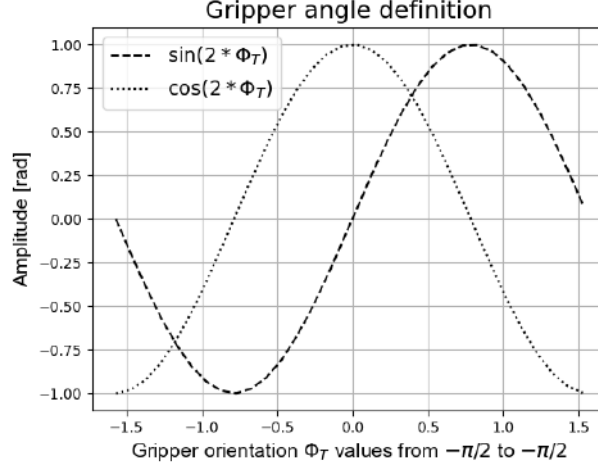


Figure 3.6: Graphical representation of the two components of a unit vector $\sin(2\tilde{\Phi}_T)$ and $\cos(2\tilde{\Phi}_T)$.

generation (YANG et al., 2016). GG-CNN is composed of six convolutional layers as shown in Figure 3.7. Its architecture has only 62,420 parameters, resulting in a low time execution, which is important for closed-loop applications. The authors achieved 52 Hz using the graphics card GTX 1070. 80% of the 8,840 images $\mathbf{I}_T$ were used for training and 20% for validation. No additional noise is added to the depth image $\mathbf{I}_T$, since the Cornell Grasping Dataset already incorporates a noise from a real camera.

The GG-CNN output corresponds to four images with 300×300 pixels (Figure 3.8): one image for the grasp quality $\tilde{Q}_\theta$, one image for each component of a unit vector ($\sin(2\tilde{\Phi}_\theta)$ and $\cos(2\tilde{\Phi}_\theta)$), and one image for the grasp width $\tilde{W}_\theta$. In each GG-CNN output image, the area of the center third part of the grasp rectangle is used to represent the grasp features (grasp quality, width and angle) as shown in Figure 3.8. The values in $\tilde{Q}_\theta$, contained in the range $[0, 1]$, indicate the grasp success rate, with the higher values denoting a higher success rate. The values of $\sin(2\tilde{\Phi}_\theta)$ and $\cos(2\tilde{\Phi}_\theta)$, indicate the antipodal grasp angle, symmetrical around $\pm \frac{\pi}{2} \text{ rad}$. The values of $\tilde{W}_\theta$, contained in the range $[0, 150]$, correspond to the grasp width in pixels. The values of $\tilde{W}_T$ are normalized in the interval $[0, 1]$ during the training.

Each pixel of $\tilde{Q}_\theta$, $\sin(2\tilde{\Phi}_\theta)$, $\cos(2\tilde{\Phi}_\theta)$, and $\tilde{W}_\theta$, corresponds to a grasp representation, summing up to a total of 90,000 grasps samples per image. Note that, by applying this process, only the positive grasps are considered when training the network, contrary to Lenz, Lee and Saxena (2015). Figure 3.9 shows the grasping rectangles as well as the antipodal gripper angle.

3.4.3 Grasp definition

Figure 3.10 shows that the best grasp pose $\tilde{g}_\theta^*$ is defined by using the index M of the maximum value in the quality image $\tilde{Q}_T$. This index is used to obtain the value of each image $\sin(2\tilde{\Phi}_T)$, $\cos(2\tilde{\Phi}_T)$, and $\tilde{W}_\theta$. The corresponding gripper width is calculated in

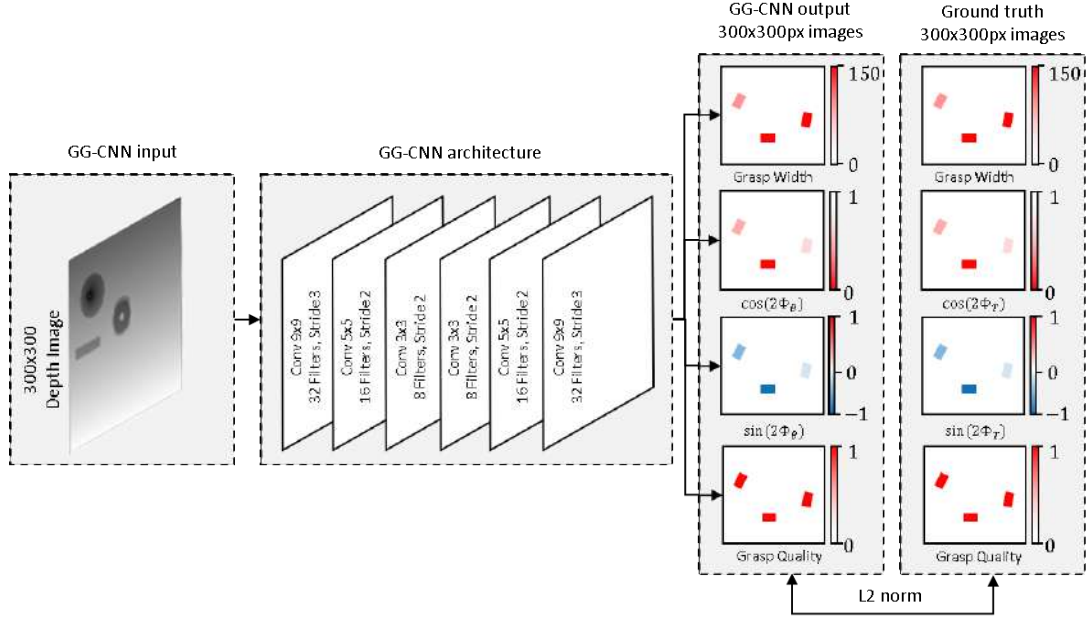


Figure 3.7: GG-CNN Architecture.

pixels in the interval $[0, 150]$. These pixels are converted in meters by using the depth measurement of the grasp pose and intrinsic camera's parameters. The height of the grasp is also obtained by using the depth measurement correspondent to the index M in the depth image $\mathbf{I}_f$ cropped to 300×300 pixels.

The image $\tilde{\Phi}_\theta$, equivalent to the grasp angle, is obtained from the images $\sin(2\tilde{\Phi}_\theta)$ and $\cos(2\tilde{\Phi}_\theta)$ by:

$$\tilde{\Phi}_\theta = \frac{1}{2} \arctg\left(\frac{\sin(2\tilde{\Phi}_\theta)}{\cos(2\tilde{\Phi}_\theta)}\right). \quad (3.2)$$

where $\sin(2\tilde{\Phi}_\theta)$ represents the vector pointing in the y direction and $\cos(2\tilde{\Phi}_\theta)$ represents the vector pointing in the x direction.

Lastly, $\mathbf{g}_\theta^*$ is acquired from a sequence of homogeneous transformations:

$$\mathbf{g}_\theta^* = t_{RC}(t_{CI}(\tilde{\mathbf{g}}_\theta^*)) \quad (3.3)$$

Where t_{CI} indicates the transformation from the image space $\mathbf{I}$ to the camera frame and t_{RC} represents the homogeneous transformation from the camera frame C to the robot's base frame R .

3.5 VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION (VGG)

Simonyan and Zisserman (2014) proposed a network architecture known as VGG. This CNN won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014 in the first and second places in the localization and classification tracks, respectively. VGG is

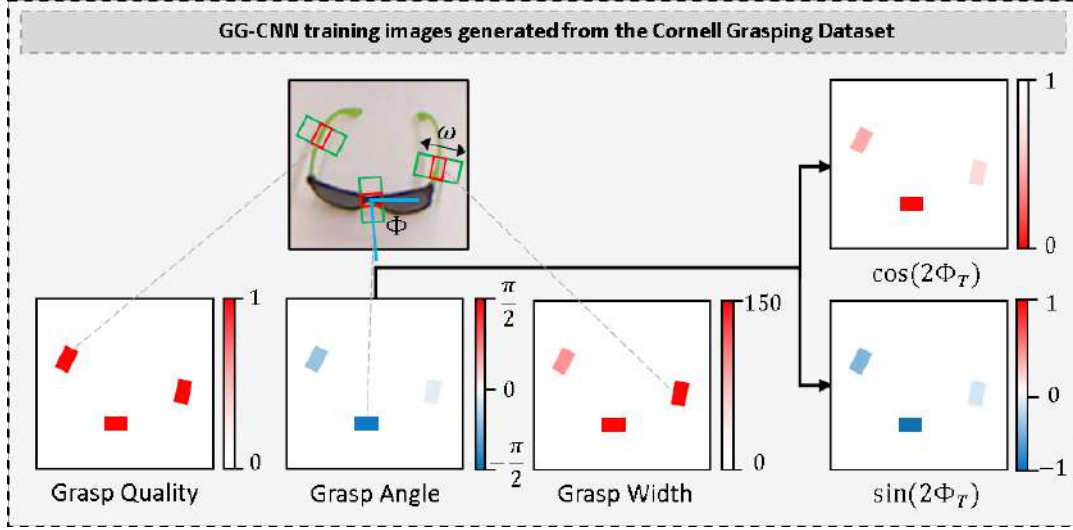


Figure 3.8: The ground truth images $\sin(2\tilde{\Phi}_T)$, $\cos(2\tilde{\Phi}_T)$, $\tilde{Q}_T$ and $\tilde{W}_T$, generated from the Cornell Grasping dataset. The white pixels of the images corresponding to the Grasp Quality, Grasp Angle, Grasp Width and $\sin(2\tilde{\Phi}_T)$ are equal to zero. The white pixels of the image corresponding to $\cos(2\tilde{\Phi}_T)$ are equal to one.

a classification network that increases the depth by applying convolutional layers of size 3×3 . The size of the receptive field (3×3) is chosen to capture the notion of left/right, up/down, and center. A convolution filter of 1×1 is also used to add nonlinearities to the network. The stride used in the convolution is always equal to one. To maintain the spatial resolution, the padding of one is chosen for the convolutional layers. Max-pooling is performed over a 2×2 pixel window, with stride two. Not all the convolutional layers are followed by max-pooling. The input to VGG is RGB images of 224×224 . To obtain the fixed-size 224×224 input images, they were randomly cropped from rescaled training images.

A stack of convolutional layers is followed by two Fully-Connected layers (FC) with 4096 channels each and one FC with 1000 channels. A softmax layer is applied at the end. VGG has six configurations named A, A-LRN, B, C, D and E (Table 3.1). Each configuration varies in depth, going from 11 weight layers in network A to 19 weight layers in network E. The depth of the layers goes from 64 to 512. The configuration D (VGG-16) is used in this thesis due to a better relationship between accuracy and performance. This is because configuration D (VGG-16) has 6 million parameters less than configuration E (VGG-19) and achieved only 0.1% less in the top-5 ILSVRC 2014 validation error.

3.6 DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION

It was a common belief that deeper neural networks could achieve better performance on object recognition. Despite this, He et al. (2016) confirmed that CNNs deeper than VGG-19 (SIMONYAN; ZISSERMAN, 2014) degrades in performance. He et al. (2016)

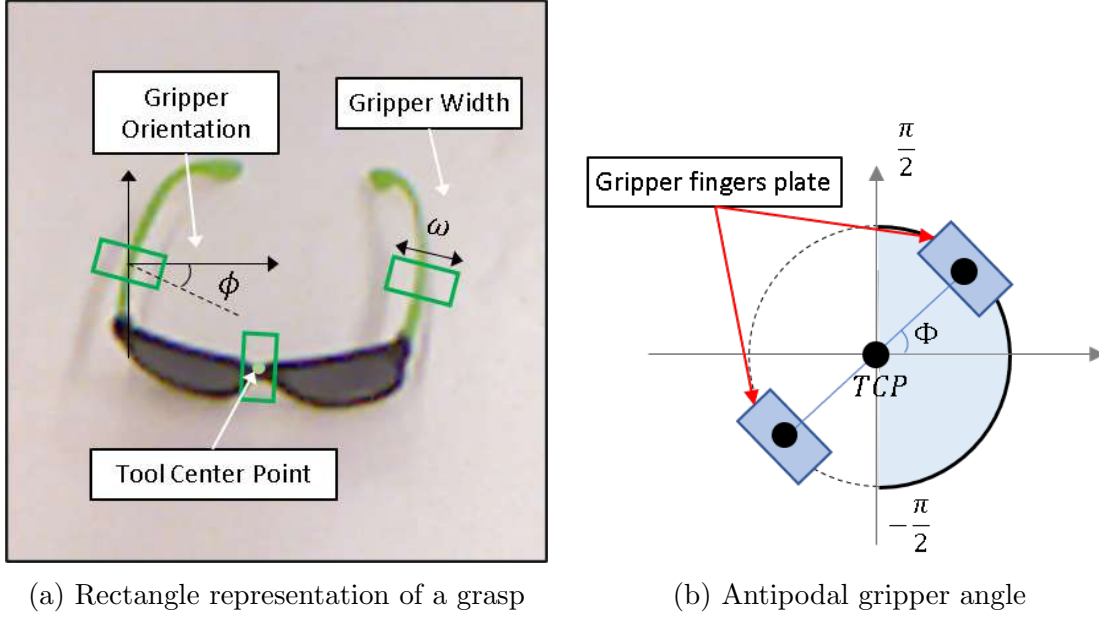


Figure 3.9: (a) Each rectangle comprises the tool center point, planar gripper angle, and gripper width. These attributes were shown in different rectangles to facilitate understanding. (b) Antipodal gripper angle is represented in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

compared plain networks with 20 and 56 layers similar to VGG-16 and proved through experiments that the more the number of layers, the more is the training and testing error. Therefore, adding more layers to the network does not improve its performance. This limitation arises due to the manifestation of vanishing gradients in deeper layers. The authors confirmed that this problem was not caused by overfitting.

He et al. (2016) proposed a CNN called ResNet that is easy to train and deeper than VGG-19 Simonyan and Zisserman (2014). This network won first place on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in 2015. The difference between ResNet and VGG-19 is that ResNet has shortcut connections. These shortcuts are applied after each block of the Table 3.2. Through the shortcuts, the input of the n convolutional layer is directly connected to the input of the next $n + p$ convolutional layer where p represents the number of convolutional layers in the building block (shown in each line of the Table 3.2), also called residual function.

Figure 3.11 shows the residual blocks used in ResNet architectures for $p = 2$ applied in ResNet-18/34 and $p = 3$ used in ResNet-50/101/152.

Between the convolutions 2.x, 3.x, 4.x, and 5.x, a convolution with stride two was applied to reduce the dimension instead of applying a pooling operation. Inside the blocks, the stride and padding operations are set to one to maintain the dimensions. When the dimension of the building blocks reduces, the shortcut connections also need to reduce in order to allow additional operations between layers. When the shortcut dimension is not required to change, an identity shortcut is used. This identify shortcut works by simply bypassing the input volume to the addition operator. It is important not to increase the complexity of the bottleneck architectures. When the width, height,

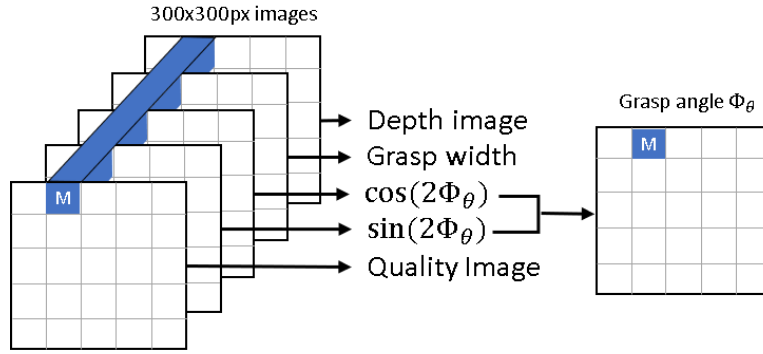


Figure 3.10: The index M of the max value in the quality image is used to acquire the value of the grasp width, angle, and the height of the grasp by using the depth measurement.

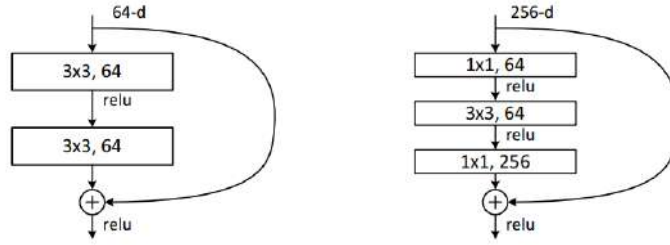


Figure 3.11: Left: A building block (on 56×56 feature maps). Right: a bottleneck building block for ResNet-50/101/152 (HE et al., 2016).

and depth of the layer are required to change, a projection shortcut is applied. This is employed between the convolutions 2.x, 3.x, 4.x, and 5.x.

There are two types of projections shortcuts: padding the input volume or performing 1×1 convolutions. In the case of padding the input volume, a convolution with stride 2, padding 1, and kernels of 3×3 are applied. The number of filters of the convolutional shortcut is chosen to be the same as the output feature map depth of the building block. In the case of 1×1 convolutions, a zero-padding with stride 2 is employed to halve the input size (width and height).

Compared to VGG nets, ResNets have fewer filters and lower complexity. VGG-19 model (SIMONYAN; ZISSERMAN, 2014) has 19.6 billion FLOPs while ResNet-34 (Table 3.2) has 3.6 billion FLOPs. ResNet-50 has 23.521 million parameters compared to 138 million parameters of VGG-16.

3.7 SINGLE SHOT MULTIBOX DETECTOR

SSD is commonly called as a framework by its authors (LIU et al., 2016). It was designed to identify multiple objects in a scene. The SSD has two variants: SSD300 and SSD512, which nomenclature refers to the image resolution of 300×300 pixels and 512×512 pixels, respectively. Liu et al. (2016) achieved a higher accuracy and detection speed (59

Table 3.1: VGG varying in size from configuration A to E (SIMONYAN; ZISSERMAN, 2014).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

FPS with mAP 74.3%) on VOC2007 test if compared to past techniques such as Faster R-CNN (REN et al., 2015) (7 FPS with a mAP 73.2%) or YOLO (REDMON et al., 2016) (45 FPS with mAP 63.4%) by applying the following procedures:

1. Generate a set of fixed-size bounding boxes and scores for each object class, succeeded by a non-maximum suppression to produce the final detection;
2. Add convolutional feature layers that decrease in size progressively to the auxiliary part of the network architecture; and
3. Using multiple features map from deeper layers to perform detection at multiple scales.

By applying the aforementioned process, Liu et al. (2016) achieved a high inference speed using low-resolution images. For each object of interest in an image, the SSD model generates: the coordinates of a bounding box that involve the objects, the class identifier for the object, and a confidence value for each prediction. According to Liu et al. (2016), the SSD outperforms previous works in terms of processed images per second.

Table 3.2: ResNet configurations (HE et al., 2016).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

The first layers of the original SSD model comprise part of a network previously trained for object identification. These first layers are called the base network. Liu et al. (2016) applied the VGG16 Network (SIMONYAN; ZISSERMAN, 2014) as a base network for the SSD, and suggested the possibility to adopt other network architectures as a base network, as shown in Figure 3.12. The remaining part of the network, called the auxiliary part, consists of several convolutional layers that gradually decrease in size. It allows the system to make detection predictions at multiple scales. The SSD300-VGG16 is very sensitive to the bounding box size, and it has worse performance on smaller objects. The SSD512-VGG16 performs better than the SSD300-VGG16 when detecting small objects, as Liu et al. (2016) state. However, both SSD300-VGG16 and SSD512-VGG16 have less localization error and are faster if compared to the R-CNN (GIRSHICK et al., 2014). The reason is that it classifies the object classes and learns to regress the object shape without decoupling steps.

In this work, the performance of the deep learning model networks SSD300, and the SSD512 was evaluated, using ResNet50 (HE et al., 2016) and VGG16 (SIMONYAN; ZISSERMAN, 2014) as a base network. ResNet-50 is chosen from the list of 34, 50, 101, and 152 ResNets as a midterm between the number of parameters and low error rates. VGG-16 is chosen from the list of VGG’s versions (11, 13, 16, and 19) of the original paper (SIMONYAN; ZISSERMAN, 2014) due to a better relationship between accuracy and performance. The COCO Dataset (LIN et al., 2014), and VOC Dataset (EVERINGHAM et al., 2010b) were used in the pre-trained versions of the SSD300 and SSD512. Due to the implementation of the ResNet50 and VGG16 as a base network for the SSD300 and SSD512, they are called SSD300-VGG16, and SSD512-ResNet50 from this subsection. Furthermore, each network appends the name of the dataset used in the training process. The SSD512-ResNet50 is renamed to SSD512-ResNet50-COCO if the COCO dataset is considered in the pre-trained version of the network.

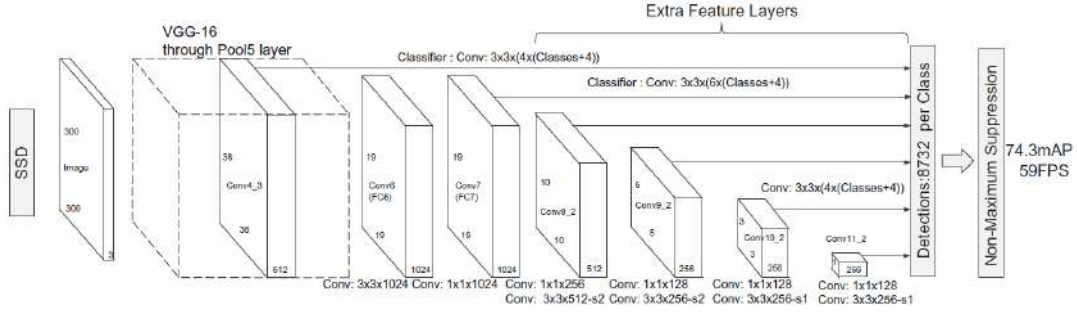


Figure 3.12: Original SSD300 Architecture using the VGG-16 as a base network (LIU et al., 2016).

3.7.1 Test objects

Mahler et al. (2017) proposed a set of objects that provides a moderate degree of complexity in the grasping process. These objects are used in the experiments shown in subsection 3.11.3. They were manufactured using a 3D printer and are shown in Figure 3.13. In order to perform the training of the SSD512 and SSD300 using these objects, a fine-tuning method was applied to a pre-trained model of these networks using both COCO and VOC Dataset.

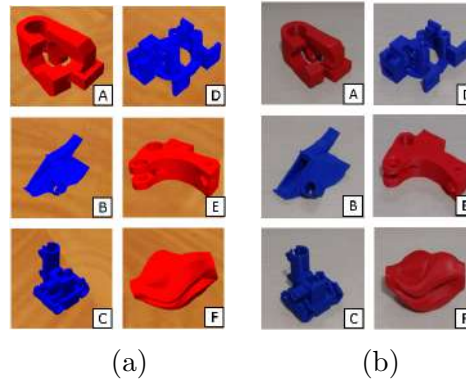


Figure 3.13: Test objects proposed by Mahler et al. (2017). (a) 3D Model (b) 3D-Printed objects. The objects are called: (A) Bar clamp, (B) Nozzle, (C) Part 3, (D) Gear Box, (E) Part 1, and (F) Vase.

3.7.2 SSD Base Network, fine-tuning and dataset

The SSD framework uses the VGG-16 (SIMONYAN; ZISSERMAN, 2014) as a base network with slight modifications. The stride applied to the fifth max-pooling layer is modified to one, the filter size applied is three, and the padding has a size one. In addition, the fc6 and fc7 layers are changed to convolutional layers by subsampling parameters from fc6 and fc7 as also performed in Chen et al. (2014). The *à trous* algorithm (HOLSCHNEIDER et al., 1990) is used to fill empty spaces during the convolution.

The dataset containing the objects of the Figure 3.13 used in training has 614 manually labeled images. 80% was used in training and 20% in validation.

The fine-tuning process was applied to the SSD300 and SSD512, in which both were pretrained in COCO and VOC dataset with VGG16 and ResNet50 as backbone networks. A LR (Learning Rate) of 0.00016 was applied in 80 epochs, using a LRD (Learning Rate Decay) in epoch 30 and 50 of 0.1. In other words, the learning rate decayed 10% in epochs 30 and 50. The Adam optimizer (KINGMA; BA, 2014) was applied with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. A non-maximum suppression algorithm was employed with a IoU (Intersection Over Union) of 50%. The batch size was equal to 16.

3.8 SELECTIVE GRASPING

The GG-CNN generates planar grasps on unknown cluttered objects (MORRISON; CORKE; LEITNER, 2018). However, this technique was not developed to grasp objects of interest. Therefore, the GG-CNN generates the best visible grasp $\tilde{\mathbf{g}}_{\theta}^*$ for any object in the robot workspace without allowing to grasp a specific one. This pipeline proposes an integration of GG-CNN and SSD. This cascade system uses RGB and depth images as input. Therefore, it requires an RGB+D sensor.

3.8.1 Grasping pipeline

To perform robotic grasps on objects of interest on a flat surface, the grasping pipeline (Figure 3.14) is proposed. It comprises the following stages:

Stage 1: The robot moves to a previously defined initial position and all the objects are kept out of the workspace. The robot's position was precisely determined to ensure optimal camera visibility into the 3D printer's environment. In this condition, a depth image $\mathbf{I}_0$ is stored.

Stage 2: All the objects (Figure 3.13) are placed in the robot's workspace. An RGB image comprising all these objects is used as input to the object detection network.

Stage 3: A object of interest is randomly chosen. If the network is capable of recognizing it, a bounding box is generated in the RGB image.

Stage 4: It is verified if the bounding box of the chosen object lies inside the GG-CNN area ($300 \times 300 \text{ pixels}$) just above the gripper. If it is true, this stage is ignored and the pipeline goes to Stage 7. If it is false, trajectory planning is requested to align the bounding box inside the GG-CNN area in Stage 5.

Stage 5: If a trajectory is requested in Stage (4), a quintic polynomial trajectory planner (SPONG et al., 2006) is generated to move the end effector to the location of the detected object. This trajectory facilitates the interpolation between the initial and final robot configurations, guaranteeing a seamless and fluid motion.

Stage 6: A bounding box is generated for the actual location of the object, given the actual camera view.

Stage 7: The indexes of the bounding box acquired in stage (3) are used to generate the bounding boxes in the depth image $\mathbf{I}$. The RGB image and the depth image $\mathbf{I}$ must be aligned. Therefore, the RGB+D camera must be calibrated.

Stage 8: The indexes of the bounding box are used to copy the corresponding area of the depth image $\mathbf{I}$ to the depth image $\mathbf{I}_f$. In this image, only the selected object is considered.

Stage 9: The filtered depth image $\mathbf{I}_f$ is cropped in a square of 300×300 pixels, to fit the GG-CNN input. The image $\mathbf{I}_f$ is inpainted to remove invalid values and a Gaussian filter is applied to increase the grasp robustness, as detailed in Section 3.10.

Stage 10: The best visible grasp $\tilde{\mathbf{g}}_f^*$ in the filtered depth image $\mathbf{I}_f$ is acquired by using the GG-CNN.

Stage 11: The best visible grasp $\mathbf{g}_\theta^*$ in $\mathbf{C}$ is obtained from the grasp $\tilde{\mathbf{g}}_f^*$ (Figure 3.15) as the following:

$$\mathbf{g}_\theta^* = t_{RC}(t_{CI}(\tilde{\mathbf{g}}_f^*)), \quad (3.4)$$

Where t_{CI} indicates the transformation from the image space $\mathbf{I}$ to the camera frame C and t_{RC} represents the homogeneous transformation from the camera frame C to the robot's base frame R .

The entire grasping pipeline is shown in Figure 3.14.

3.9 HARDWARE AND SOFTWARE IMPLEMENTATION

Following the guidelines provided in Mahler et al. (2018), this section describes each aspect of the hardware and software used to implement the grasping pipeline presented.

3.9.1 UR5 Robot

The UR5 Robot (ROBOTS, 2019) is a flexible, lightweight and user-friendliness industrial robotic arm designed to perform repetitive manual tasks weighing up to 5 kg (Figure 3.16a). It is commonly applied in light tasks such as packing, assembly, bin picking, or testing. This robot is commonly used in industry due to the fast playback time offered, being operational in less than a day due to the simple 3D visualization programming provided in teach pendant. The UR5 robot has a maximum reach of 850 mm . The joint limit rotation is $\pm 360^\circ$ and the maximum speed is $180^\circ/\text{s}$. This robot has 6 rotating joints and weights 18.4 kg . Its repeatability is $\pm 0.1\text{ mm}$.

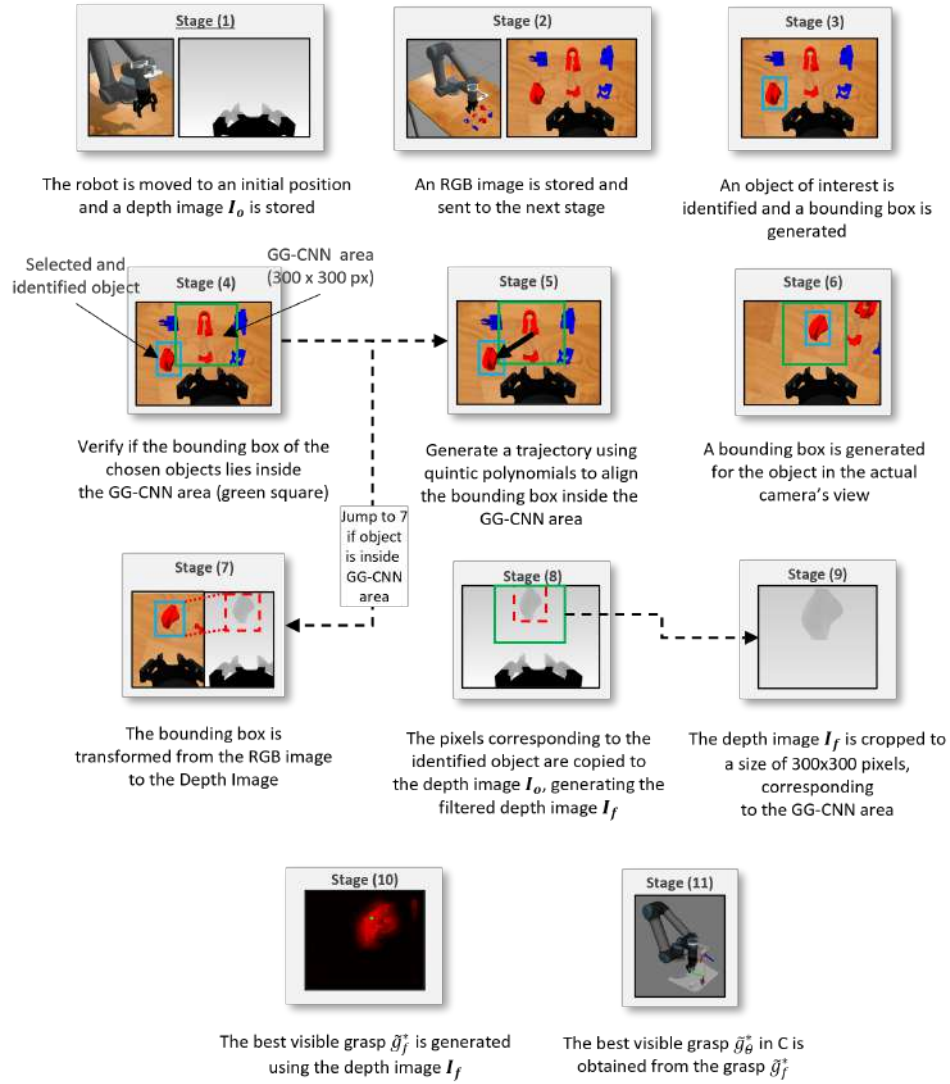


Figure 3.14: Grasping pipeline.

3.9.2 Robotiq Gripper 2F-140

The Robotiq Gripper 2F-140 (ROBOTIQ, 2020) is a plug and play end effector designed for collaborative robots. Its payload is 50% lower than the Robotiq Gripper 2F-85 or up to 2.5 kg , and a grip force from 10 N to 125 N . It has a maximum stroke of 140 mm (64, 7% higher than the Robotiq Gripper 2F-85). Its maximum closing speed is configurable between 30 to 250 mm/s . Since the gripper weight is 1 kg and the UR5 robot has a payload of 5 kg , only objects up to 4 kg can be grasped.

3.9.3 Intel Realsense D435

The Intel Realsense D435 (INTEL, 2019) is a stereo RGB-D camera with a range up to 10 m . It is composed of left and right HD image sensors, one IR projector, and one 1080p

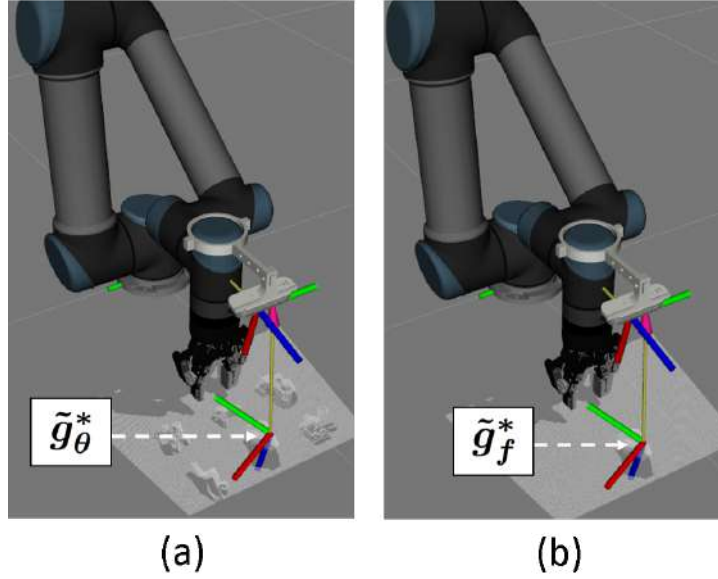


Figure 3.15: Depth cloud acquired from the simulated Intel Realsense D435 camera in RViz (GOSSOW et al., 2020) (a) Depth cloud representation of the depth image I and the best visible grasp $\tilde{g}_\theta^*$ (b) Depth cloud representation of the filtered depth image I_f acquired in stage 9 and the best visible grasp $\tilde{g}_f^*$.

RGB image sensor. The depth output resolution is up to 1280×720 at maximum 90 Frames Per Second (FPS). The minimum depth distance is 0.105 m . The minimum depth distance is highly important in grasping applications since the camera cannot update the grasp pose when it is close to the object. The depth field of view is $85^\circ \times 57^\circ (\pm 3^\circ)$.

The RGB sensor provides the maximum resolution of 1920×1080 at 30 FPS. The RGB field of view is $69.4^\circ \times 42.5^\circ (\pm 3^\circ)$. The connector is a USB-C 3.1 Generation 1, providing a 5 Gbps of data rate, approximately 10 times faster than USB 2.0 high-speed. Custom camera support was designed and manufactured on a 3D printer to fix the camera on the UR5 robot's wrist. The distance between the camera and the Robotiq Gripper 2F-140' tool center point is 23 cm along the line of sight. The custom 3D-printed camera support was rotated 14° in the gripper's direction, as shown in Figure 3.16b.

3.9.4 Robotic Operating System Implementation

The Robot Operating System (ROS) (QUIGLEY et al., 2009) is an open-source robot framework developed for dealing with the growing development and complexity of advanced robotic systems. This framework was designed to follow a series of criteria such as:

1. The possibility to connect multiple hosts at a run time through a peer-to-peer topology;
2. Support cross-language development. It is possible due to a language-neutral interface definition language (IDL) to describe the messages sent between modules

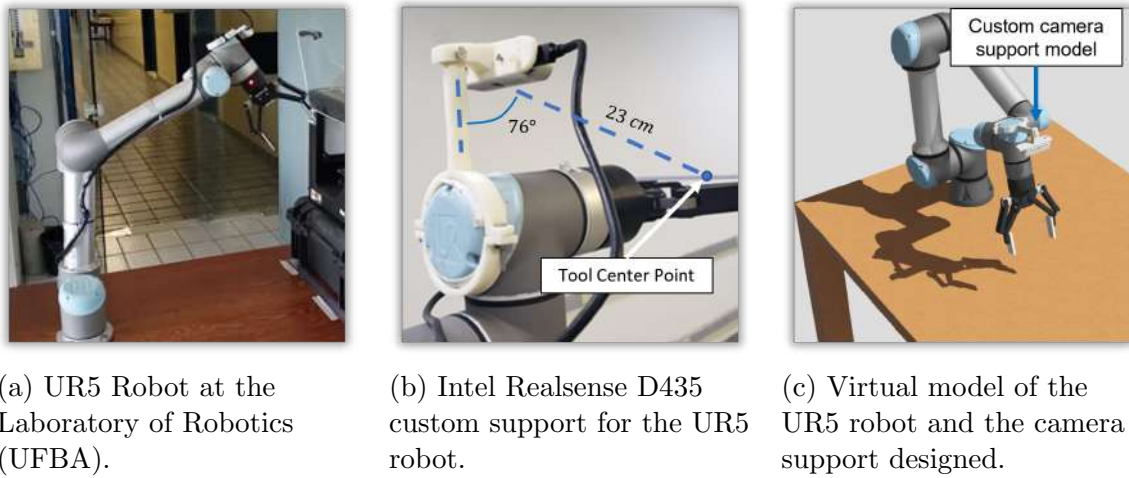


Figure 3.16: (a) UR5 Robot at the Laboratory of Robotics (LaR) at UFBA (b) the Intel Realsense D435 custom support developed (c) virtual model of the UR5 robot and the custom camera support developed.

(nodes or services);

3. Stay lean to ensure effortless sharing of implementations;
4. Allow the re-use of code from numerous open-source libraries such as OpenCV (BRADSKI, 2000b) and OpenRAVE (DIANKOV; KUFFNER, 2008); and
5. Offer unrestricted access for both non-commercial and commercial projects by adopting the BSD license, ensuring freedom and openness in usage.

Within the framework of Robotics Operating System (ROS), facilitating data exchange among modules occurs through inter-process communication. This communication architecture comprises essential elements: nodes, messages, topics, services. Nodes, functioning as autonomous processes, execute computations facilitated by messages—a structured data format supporting fundamental types (boolean, integer, floating-point, etc.). Messages are transmitted via topics, where a publishing node disseminates information to a specific topic, allowing subscribing nodes to receive this data. Additionally, ROS employs services, enabling nodes to request specific tasks or information from other nodes, creating a synchronous, request-response interaction.

The UR5 Robot², Intel Realsense D435 camera³, and the Robotiq 2F-140⁴ are compatible with ROS since there are solid ROS packages already implemented for this purpose. The SSD and the GG-CNN were also implemented using different ROS nodes that receive data from the Master Node, to perform the object recognition and generate a grasp pose. The Webots simulator was used to evaluate the performance of the grasping pipeline

²https://github.com/ros-industrial/universal_robot

³<https://github.com/IntelRealSense/realsense-ros>

⁴<https://github.com/ros-industrial/robotiq>

in simulation, and the RViz⁵ (GOSSOW et al., 2020) was used for visualizing the data being published by the nodes such as the point cloud and the coordinate frames. Webots simulator was chosen over Gazebo⁶ due to the better performance related to physics interaction between the gripper and objects.

Figure 3.17 shows the architecture implemented using ROS, RViz, and Webots. The virtual model of the Intel Realsense D435 camera, UR5 robot, and Robotiq Gripper 2F are used in Webots. The main node receives the simulated depth data from the Intel Realsense D435, the joint angles from the UR5 robot, and the gripper angles from the Robotiq Gripper. By following the grasping pipeline detailed in Section 3.8.1, the main node generates a new grasp g_{θ}^* in the robot’s base coordinate frame $\mathcal{C}$. To move the gripper to the grasp pose, a quintic polynomial trajectory algorithm is applied considering the initial and final joint angles. The TF package⁷ is used to track the model coordinate frames over time.

3.10 PRE-PROCESSING AND POST-PROCESSING

3.10.1 Pre-processing

The depth images acquired from a real sensor often provide invalid depth measurements. These invalid values can highly affect the GG-CNN performance by causing the grasp to converge to regions of poor grasp quality. Therefore, to avoid this problem, a method called inpainting provided by OpenCV (BRADSKI, 2000a) is applied to the depth image before the network processing. This method removes the invalid depth measurements by replacing those values with its neighboring pixels so that it looks like the neighborhood.

3.10.2 Post-processing

Johns, Leutenegger and Davison (2016) found that filtering the quality image $\tilde{Q}_{\theta}$ using a Gaussian kernel removed the maxima that were close to areas of low quality grasps as well as created a more diverse set of grasps that are more consistent between the successive grasps when performing the grasp in closed-loop. This filter is used in the quality image $\tilde{Q}_{\theta}$ of the GG-CNN.

3.11 EXPERIMENTS

3.11.1 Hardware and software

The computations were performed on a desktop running Ubuntu 20.04 with a 2.9 GHz Intel Core i5-10400F, 32 GB RAM, and NVIDIA GeForce RTX 3060 graphic card with 12 GB VRAM. The experiments were performed in Webots simulator (KOENIG; HOWARD, 2004) integrated to the ROS (Robot Operating System) Noetic version (QUIGLEY et al., 2009). The virtual model of the UR5 robot (Figure 3.16c) was employed in the simulation.

⁵<https://github.com/ros-visualization/rviz>

⁶<http://gazebo.org/>

⁷<http://wiki.ros.org/tf>

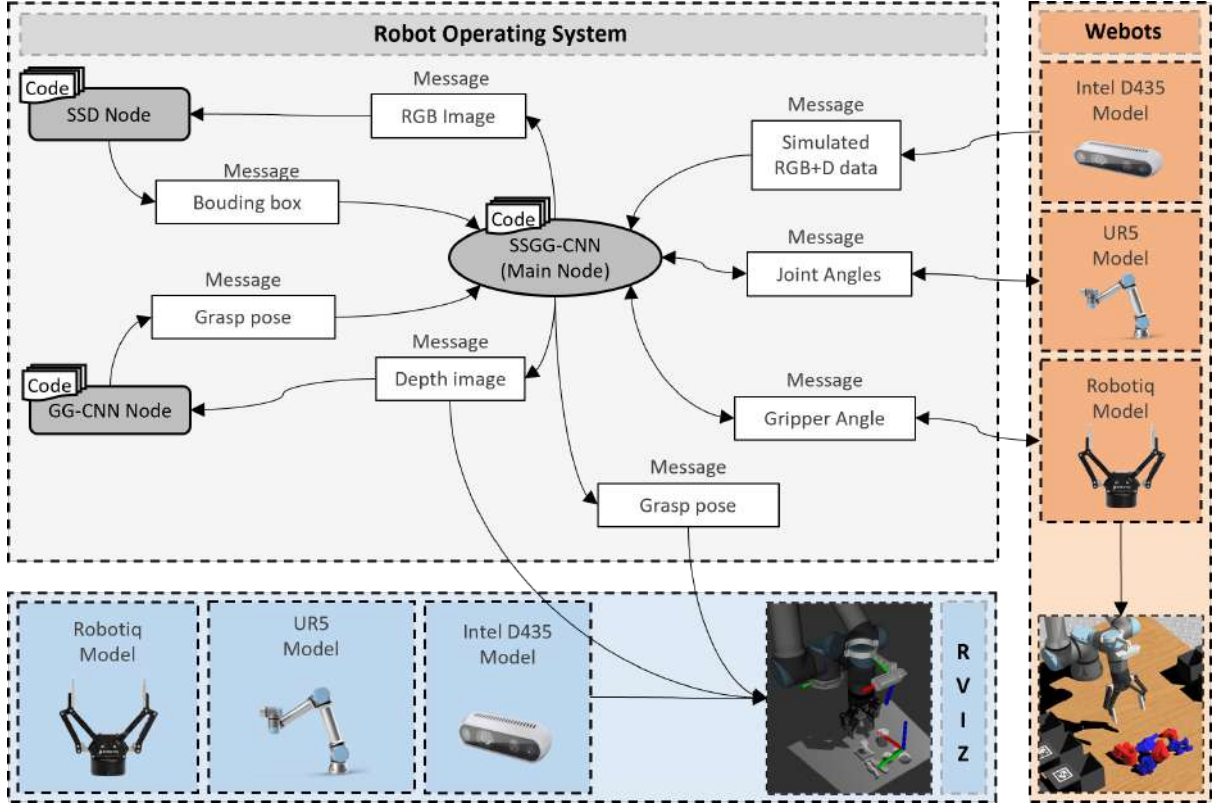


Figure 3.17: The condensed version of the selective grasping ROS architecture employed. The SSD and GG-CNN nodes are used to perform the object recognition and generate the grasp pose. The pivotal SSGG-CNN Main node orchestrates interactions between the user, the Webots simulator, and the nodes linked to SSD and GG-CNN. Its role is to manage the sequence of detection and grasp generation requests, ensuring their orderly execution for correct robotic grasping and object detection.

3.11.2 Performance evaluation of the object's recognition system

This subsection shows a performance evaluation of the SSD300-VGG16-VOC, SSD512-ResNet50-COCO, SSD512-VGG16-VOC, and the SSD512-VGG16-COCO in detecting the 3D-printed objects shown in Figure 3.13 using the IoU metric with the threshold value set to 0.5, 0.75 and 0.5 : 0.95 (an average of the IoU's between 0.5 and 0.95).

The SSD300-VGG16-VOC network had the worst performance since the small image resolution jeopardizes the object identification, as stated by Liu et al. (2016). As a result, the mAP is lower with the SSD300 network, regardless of the base network used. The SSD512 has proved to be better in recognizing the objects than the SSD300 in simulation, although it has more computational cost compared to the SSD300. In addition, the base network ResNet50 has shown small improvements if compared with the VGG16 network. The higher sensitivity of the ResNet50 to edges compared to the VGG16 may contribute to this result (MIAO et al., 2019). It was possible to achieve a higher mAP using the COCO dataset if compared to the VOC dataset since the objects in COCO tend to be

smaller than VOC.

The best performance related to the object recognition system is given to the SSD512-ResNet50-COCO. Since recognition is part of the grasping system, it highly influences grasping performance.

Table 3.3: Evaluation of the average precision by class and mean average precision considering the IoU metric with threshold of 0.5, 0.75, and 0.5 : 0.95.

Network	mAP@IoU			Average Precision by class					
	0.5	0.75	0.5:0.95	Bar clamp	Gear-box	Vase	Part 1	Part 3	Nozzle
SSD300-VGG16-VOC	93	86.4	67.7	99.62	99.9	90.91	81.64	90.91	99.65
SSD512-ResNet50-COCO	97.7	92.2	73.5	100	90.91	100	90.91	99.86	100
SSD512-ResNet50-VOC	96.5	87.5	69	100	99.9	99.93	90.91	90.91	100
SSD512-VGG16-VOC	94.1	87	69.5	99.62	90.91	99.48	81.82	90.91	100
SSD512-VGG16-COCO	95.4	89.6	70.7	100	99.71	90.76	100	90.83	90.91

By using the hardware mentioned in subsection 3.11.1, the GG-CNN takes the average of 100 *ms* to compute a new grasp. In order to recognize an object, the SSD512 takes 230 *ms*, and the SSD300 takes 200 *ms* with the code written in Python.

3.11.3 Performance evaluation of the robotic grasping system

In the experiment, only static grasps are performed. The objects of interest are randomly placed in the workspace before the grasps are performed. After picking the object, it is moved to a bin close to the robot. The object's classes are indicated in Figure 3.13. A grasp is considered successful if an object of interest is recognized and moved to the bin, following the Metric 6 referenced in Section 2.6. The success rate is considered as a fraction of the total number of grasp attempts. The object is replaced in the workspace if it is successful.

In simulation, the objects positions were kept through all the experiments for a fair comparison between the object recognition methods. In order to acquire the joint angles of the robot given the position and the orientation of the grasp, the inverse kinematics problem was solved and implemented for the UR5 Robot arm.

When only the GG-CNN is used, the grasps are performed considering the depth image I where all the objects are included. Therefore, for a fair comparison, the GG-CNN is only evaluated with the object recognition networks and not individually, as it cannot identify the objects in the workspace. Five experiments were conducted considering the GG-CNN for grasp generations and each one of the five networks mentioned in Table 3.3. The objects shown in Figure 3.19a and b were employed. The grasping pipeline used in this experiment is explained in Section 3.8.1. April tags (OLSON, 2011) were used to identify each one of the six bins (Figure 3.19c) located in the workspace to place the object after the grasp action. A successful grasp is considered when the object is placed in the correct bin. If the grasp generated is not stable enough, during the trajectory to the bin, the object falls from the gripper, and the grasp is considered a failure.

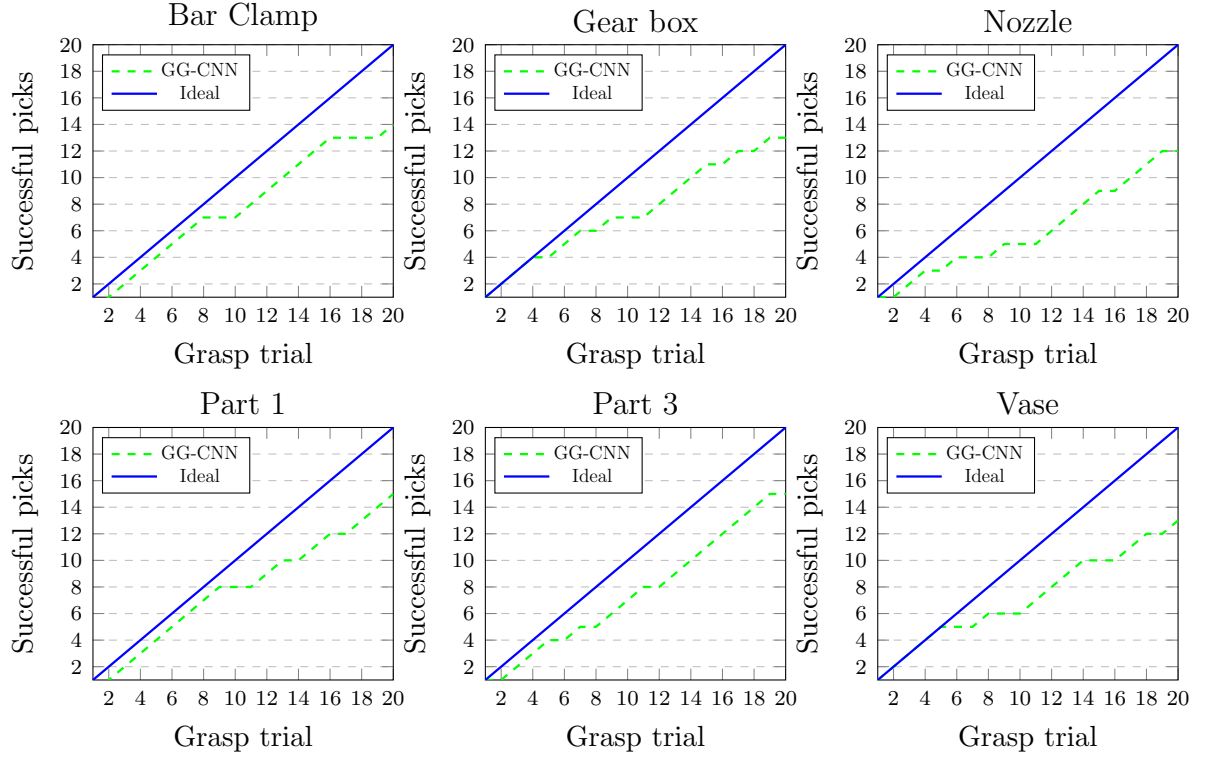


Figure 3.18: Grasps executed by the proposed grasping pipeline (Figure 3.14). The Grasp trial axis represents the number of grasps executed by the robot. The dashed green line represents the number of successful grasps executed by the robot using the GG-CNN model.

Fig 3.18 shows the successful and failed grasps for each one of the six objects used in the experiments. In the experiments, 20 grasp attempts were performed per object. An average of 68% of the grasps was successful in this environment. Table 3.4 shows the successful and fail grasps per object. Results show that some objects such as part 1 and part 3 are easier to grasp compared to the other objects. The Nozzle is the most challenging object to grasp since it has a small contact area. The Gear Box recognition failed 20%, decreasing the grasp success for this object.

According to Mahler et al. (2018), industrial practitioners characterize picking in terms of the rate, reliability, and range (class of objects). One metric for comparison is MPPH, which is formalized as:

$$MPPH = \mathbf{v} \cdot \Phi, \quad (3.5)$$

computed as the mean over T grasp attempts, where $\mathbf{v}$ is the mean grasp rate, or an average number of attempts per hour, and Φ is the mean grasp reliability (or success rate). The mean grasp rate can be formalized as:

$$\mathbf{v} = \frac{1}{t_s + t_c + t_r}, \quad (3.6)$$

Table 3.4: Grasp and recognition fail per object considering the SSD512-ResNet50-COCO as the object detector. 20 grasps were performed per object.

	Grasp Success	Grasp Fails	Recognition Fail
Bar Clamp	70%	30%	0%
Part 1	75%	10%	15%
Part 3	75%	25%	0%
Nozzle	60%	25%	15%
Vase	65%	25%	10%
Gear box	65%	15%	20%

where $t_s + t_c + t_r$ are the average times for sensing, computation and robot motion, respectively, in fractions of an hour.

The mean grasp reliability Φ is defined as:

$$\Phi = \left[\frac{1}{T} \sum_{t=0}^{T-1} R_t \right], \quad (3.7)$$

where T is the total number of attempts, and $R_t = 1$ if the grasp is considered successful.

Table 3.5 shows the grasp reliability of each object recognition network used in this research. The performance of the object recognition network is essential to generate a new grasp using the GG-CNN, since the object recognition precedes the grasp generation, as stated in the grasping pipeline. As expected, the SSD512-ResNet50-COCO had the best MPPH performance, although it was not noticed a big difference between the MPPH of the object recognition networks in the simulated experiments. The ideal MPPH depends on the application. For example, in a warehouse, the MPPH should be high to increase the productivity. Besides that, the MPPH is also related to the robot's speed. The faster the robot, the higher the MPPH.

Table 3.5: Evaluation of the grasping reliability using the Man Picks Per Hour (MPPH) metric as suggested in Mahler et al. (2018).

	Attempts	Fails	MPPH
SSD300-VGG16-VOC	80	16	169
SSD512-ResNet50-COCO	80	13	177
SSD512-ResNet50-VOC	80	14	175
SSD512-VGG16-VOC	80	15	172
SSD512-VGG16-COCO	80	14	175

3.11.4 Failure mode

The failure modes can be classified into the following categories:

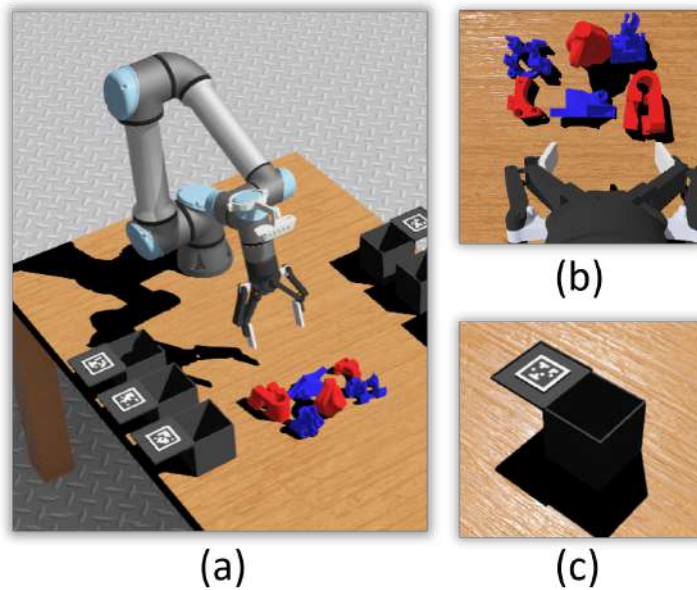


Figure 3.19: (a) Simulated environment in Webots (b) Camera's view from the support shown in Figure 3.16 (c) Bin where the objects are placed after the grasp action.

- The grasp generated is planar. Consequently, it may not be possible to grasp the objects depending on their position and orientation in the workspace;
- If the camera is located in the robot's wrist, the gripper fingers may be visible. In consequence, the object recognition system may generate false positives in this region;
- When a quintic polynomial trajectory is generated to align the bounding box into the GG-CNN area, the object recognition system needs to acquire a new RGB image to generate a new bounding box from the new pose. Therefore, it may not be possible to recognize the object considering the new camera angle;
- The GG-CNN only generates a single grasp that is acquired by evaluating the max quality value from the network outputs. This can lead to a poor grasp generation since the generated grasp may not be kinematically feasible or may be in collision with other objects; and
- Since the bounding box of the RGB image must be transformed to the depth image, it is required to align the RGB and Depth image precisely. The Intel Realsense D435 camera has a calibration tool that can be used to calibrate the RGB and Depth images. Nevertheless, alignment inaccuracies are difficult to completely avoid.

3.12 CONCLUSION

In this chapter, a two-stage cascade system to perform selective planar grasps is proposed. This system performs robotic grasps on objects of interest located in the robot's

workspace. The proposed method has demonstrated improvements compared to the GG-CNN (MORRISON; CORKE; LEITNER, 2018), since it allows the grasping of specific objects in the robot workspace. The grasp generator and the object recognition system are different algorithms. Nevertheless, experiments performed in Webots have shown that the performance of the object recognition network affects the grasping success rate since it is employed in series before the grasp generation.

The grasping pipeline proposed in this research is slower compared to the GG-CNN. The reason is that SSD has more parameters and it is integrated in series with the GG-CNN. The SSD considered in this research takes 50 ms on average to recognize an object using an RGB image and the GG-CNN takes 20 ms to generate a new grasp position and orientation. The execution time is fixed and depends on the GPU used. The number of parameters of the SSD regardless of the base network is considerably higher than the GG-CNN. Despite this disadvantage, the technique proposed demonstrated a good performance since it is applied in a static environment.

It is important to note that GG-CNN is a planar grasp generator. In other words, the gripper can only grasp an object orthogonal to the surface. It limits applications such as grasping manufactured objects in a 3D printer since the gripper cannot be orthogonal to the 3d printer bed. In that case, a 6D grasp generator is recommended since the robot can grasp the object from all directions outside the printer.

The GG-CNN was not trained using SGT. Therefore, it required a high number of ground truths grasps manually labeled. In practice, it would not be feasible to manually generate thousands of grasps to improve the CNN. Recent research is employing SGTs using parallel processing to generate millions of annotated grasps. Nevertheless, it requires powerful GPUs to deal with the high processing of the dataset generation (MOUSAVIAN; EPPNER; FOX, 2019). Besides that, the dataset used to train the GG-CNN only considers grasps of the object from a few points of view. It is a limitation of this technique since the same object can be grasped from any angle and point of view and not just one or two.

Chapter 4

This chapter presents a selective grasping pipeline to generate 6D grasps using an RGB+D sensor avoiding collisions between the robot’s gripper and the environment. The performance analysis of the proposed system is validated using real hardware and environment.

SELECTIVE 6D ROBOTIC GRASPING USING CONVOLUTIONAL NEURAL NETWORKS

4.1 INTRODUCTION

Grasps can be single (MORRISON; CORKE; LEITNER, 2018; JOHNS; LEUTENEGGER; DAVISON, 2016) or multiple (MOUSAVIAN; EPPNER; FOX, 2019; GUALTIERI et al., 2016). Multiple grasps refer to the conception of more than one feasible grasp per object and single grasp techniques converge to a single solution per image. Ongoing research continues to focus on multiple 6D grasp techniques, driven by the necessity to navigate the expansive six-dimensional space involving the object. This exploration is essential despite the considerably higher computational demands compared to single grasp techniques.

In the context of additive manufacturing systems (ARRAIS et al., 2019; COSTA et al., 2020), it is necessary to apply a grasping technique capable of yielding a diverse set of 6D grasps. This is necessary as some grasps may not be kinematically possible or collides with objects in the robot’s volumetric space. Deep learning-based grasps techniques provided a great tool to improve the performance of grasping unknown objects. However, grasps are usually performed in environment that offer a low risk of collision with objects. Techniques to avoid collisions between the robot’s gripper and the environment are still an open area of research.

Some grasping techniques have shown a convincing grasping performance (VIERECK et al., 2017; MAHLER et al., 2017; LEVINE et al., 2016). Despite that, they consider only planar grasps and have high computational cost. The solution of the computational efficiency problem was proposed by Morrison, Corke and Leitner (2018). Nevertheless, it considers only planar and single grasps. Mousavian, Eppner and Fox (2019) presented a grasp technique that generates multiple grasps per object in 6D. Therefore, it is possible to choose a grasp in which the inverse kinematic solution for the robot is feasible and avoids possible collisions with other objects in the environment.

Regarding the fourth industrial revolution, there is a high demand for flexible and versatile robotic manipulators applicable to new environments and products (COSTA et al., 2020). In this regard, the additive manufacturing industry has shown a growing interest in these systems, which regularly deal with an increasing number of new parts (ARRAIS et al., 2019). In view of the above, a grasping method capable of generating multiple 6D grasps is required.

In this chapter, a selective grasping pipeline is introduced for generating 6D grasps by leveraging an RGB+D sensor. This approach prioritizes collision avoidance between

the robot's gripper and the surrounding environment (Figure 4.1). To avoid collisions with nearby obstacles, a new collision detection system and a heuristic method to filter grasps were developed. The collision avoidance algorithm is exclusively applied to the gripper, as, in the particular test case, there is no danger of the robotic arm colliding with the 3D printer. Although the method has been tested in an Additive Manufacturing Unit to pick objects from a 3D printer bed, it is not limited to this application and can be adapted for other environments, such as bin picking. An extended analysis of the grasping performance is given with real and simulated experiments.

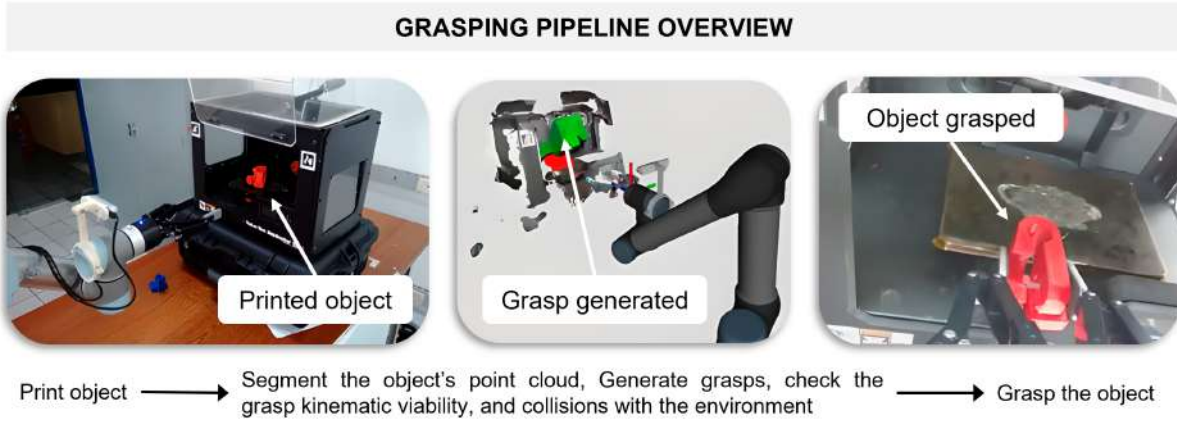


Figure 4.1: Overview of the proposed grasping pipeline.

4.2 CONTRIBUTIONS

This chapter has the following main contributions:

- Development of a selective grasping pipeline based on Variational Autoencoders and an Object Segmentation Network to generate 6D grasps using an RGB+D sensor, avoiding collisions between the robot's gripper and the environment;
- Development and validation of a low computational complexity collision avoidance system to discard grasps in collision with the environment and a heuristic method to filter the best grasps;
- Integration of an object recognition and instance segmentation method, a 6D grasping generator, and a new collision detection system;
- Development of an automatic object recognition dataset generation pipeline using a simulator; and
- Validation of the proposed system using an UR5 Robot Arm Manipulator, an RGB+D camera Intel Realsense D435, and the gripper Robotiq 2F-140.

4.3 PROBLEM DEFINITION

It has only been in the last few years that convincing experimental data have proven, in practice, the efficiency of grasping methods. Nevertheless, the grasping techniques are often applied for picking objects on planar surfaces such as a table (MORRISON; CORKE; LEITNER, 2018) or in a bin (MAHLER et al., 2019; MORRISON; CORKE; LEITNER, 2019). This workspace (table and bin) offers relatively simple test benches to evaluate the grasping performances if compared with constrained spaces such as inside 3D printers. Therefore, 4D grasping methods (also called planar grasps) are enough to generate feasible grasps for planar surfaces but not are suitable for constrained spaces such as inside 3D printers.

To perform grasp in constrained spaces, it is required to avoid collisions with nearby obstacles such as the printer bed. Therefore, it is necessary to generate a set of feasible grasps with different positions and orientations for the same object since some grasps are in collision with obstacles in the workspace or kinematically infeasible. Following the description of symbols utilized within this thesis:

RGB image. C_i expresses a raw 8-bit RGB image.

Depth image. Let I be an 8-bit 2.5D depth image in which every object in the environment is considered. H and W represent the height and the width of this image, respectively.

Segmentation Mask. M_r represents the object segmentation mask.

Object point cloud. N_r evidence the detected object point cloud.

Filtered object point cloud. N_f represents the filtered point cloud of the detected object.

Printer point cloud. K_r represents the raw point cloud of the 3D printer.

Downsampled printer point cloud. K_d denotes the downsampled point cloud of the 3D printer.

Grasp set. $\tilde{G}_g = (\tilde{P}, \tilde{O})$ denotes a 6D grasps set, in which $\tilde{P}$ and $\tilde{O}$ denotes the position and orientation angles, relative to the camera frame.

Filtered grasps. $G_{gf} = (P_{gf}, O_{gf})$ represents the position P_{gf} and orientation O_{gf} of the filtered grasps by applying the heuristics described in section 4.7, relative to the camera frame.

Collision-free grasps. $G_o = (P_o, O_o)$ denote the position P_o and orientation O_o of the collision-free grasps, relative to the robot base coordinate frame.

Grasp on the robot base coordinate frame. $G_{fb} = (P_{fb}, O_{fb})$ represents the position P_{fb} and orientation O_{fb} of the collision-free grasps, relative to the robot base coordinate frame.

Current gripper pose. $G_a = (P_a, O_a)$ describes the actual gripper position P_a and orientation O_a , relative to the robot base coordinate frame.

4.4 OBJECT SEGMENTATION

4.4.1 Mask R-CNN

Mask R-CNN (HE et al., 2017) is a deep learning algorithm used for object instance segmentation. It extends *Faster R-CNN* (REN et al., 2015) by adding a branch for predicting segmentation masks on each Region of Interest (RoI). This branch is set up in parallel with the existing branch for classification and bounding box regression. The mask branch downgrades the performance of the object detection but is still able to reach better performance and accuracy than the COCO instance segmentation task winner in 2016. Mask R-CNN runs at five fps on an Nvidia Tesla M40 GPU.

The layer applied in Faster R-CNN after the Region Proposal Network was used to extract features to classify and apply box regression. It was not designed for pixel-to-pixel alignment between network inputs and outputs. To solve this problem, He et al. (2017) applied a quantization-free layer, called *RoIAlign*, that maintains spatial locations. The classes' mask is inferred independently and depends on the RoIAlign to classify and predict categories as is also done on Faster R-CNN.

Mask R-CNN receives an RGB image as input and returns three outputs: a bounding box, class, and mask for each object. Similar to *Faster R-CNN*, the Mask R-CNN also employs a backbone architecture. Results show that using a better feature extractor network such as ResNeXt-101-FPN (XIE et al., 2017) instead of ResNet-50-FPN (LIN et al., 2017), improves the performance of the Mask R-CNN.

4.4.2 Dataset and Training

In order to detect and segment the object's image, the Mask R-CNN was trained using a fine-tuning process. The test objects, as proposed by Mahler et al. (2017), depicted in Figure 4.2, were specifically chosen to evaluate the grasping method due to their difficulty to grasp.

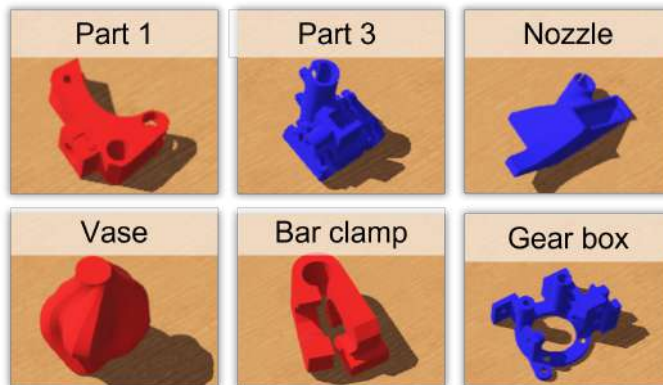


Figure 4.2: Objects (MAHLER et al., 2017) used to test the robotic grasping pipeline proposed in this work.

A synthetic dataset was generated using the Webots simulator by applying the following pipeline (Figure 4.3):

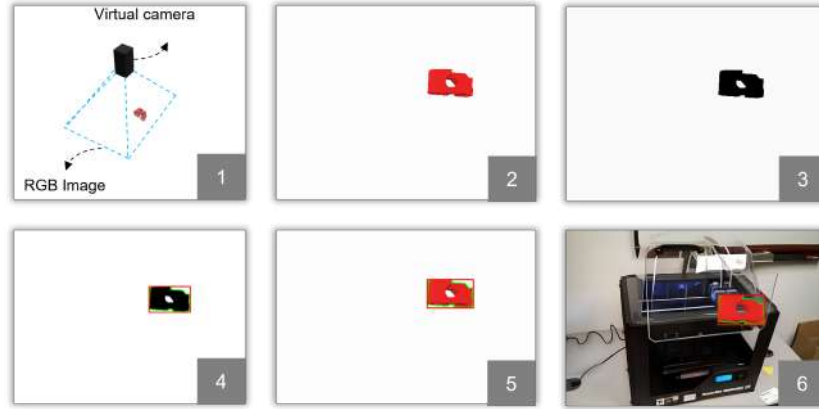


Figure 4.3: Dataset generation pipeline using Webots.

1. The object is randomly positioned in a predefined area with a white background and a camera is used to capture an RGB image of the object.
2. The RGB image is generated in simulation.
3. It was experimentally verified that the object is better segmented in the simulation if the RGB image is turned into a gray image.
4. The object's contour is generated using the chain approximation method (BRADSKI, 2000b). A bounding box is also automatically generated using the edges of the contour.
5. The contour is shown in the RGB image of stage 2.
6. The object pixels in the synthetic RGB image are copied to real images of 3D printers as shown in Fig 4.4. 3D printers were used to increase the detections' accuracy since the network will learn to differentiate the printer and the objects.

In total, 900 images were generated using this pipeline, 150 for each object in Figure 4.2. The training and validation set was divided into 80% and 20% respectively. The Average Precision (AP), considering the average of IoU thresholds of 0.5:0.05:0.95, was 87.9% for the segmentation task. This average precision with averaging IoUs is used to determine the winner of the COCO challenge dataset. Figure 4.5 shows detection examples using real object images. Figure 4.4 shows some images of the dataset automatically generate to fine-tune the Mask R-CNN.

The Mask R-CNN was pretrained on the COCO dataset and fine-tuned in 26 epochs. The learning rate was set to 0.0025 with a decay of 10% in epochs 17 and 23. The Stochastic Gradient Descent was used with a weight decay of 0.0001 and momentum of 0.9. The batch size was set to two.

4.5 GRASPNET

This section briefly describes how *GraspNet* works. For more details refer to Mousavian, Eppner and Fox (2019).



Figure 4.4: Some figures part of the dataset automatically generated in Webots using real images as background. This dataset was used to train the Mask R-CNN.

GraspNet is a 6D grasping CNN based on Variational Autoencoders (VAE) (KINGMA; WELLING, 2013). VAEs are deep generative models developed based on classic Autoencoders. This network consists of an encoder and decoder that map the input data to a latent space of reduced dimensionality. The encoder and decoder use the *PointNet++* (QI et al., 2017) network architecture to extract spatial characteristics from each point of the object’s point cloud and the robot’s end effector for each grasp generated, whether successful or not.

This network has two modules: generator and evaluator. The generator module relies on different samples of a latent space and the object and gripper’s partial point cloud to produce several grasps. The evaluator module accepts or rejects the grasps based on their probability of success. GraspNet was trained from grasps obtained in the software FleX (VICENT et al., 2016). The dataset comprises 2 million successful grasps generated from 10.8 million grasps sampled using SGTs. The grasp reconstruction cost function is given by

$$\mathcal{L}(g, \hat{g}) = \frac{1}{n} \sum \|\mathcal{T}(g; p) - \mathcal{T}(\hat{g}; p)\|_1, \quad (4.1)$$

where $\mathcal{T}(\cdot; p)$ represents the transformation of a set of predefined points p on the robot gripper, g is the ground-truth grasp position and orientation, $\hat{g}$ is the grasp generated by the decoder, and n is the number of grasps generated.

Considering the latent space z , the point cloud of the object N_f , and the generated

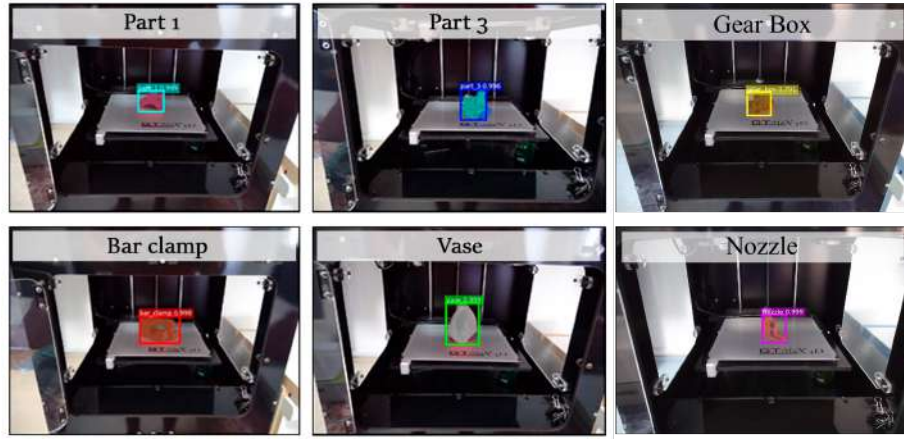


Figure 4.5: Detection examples performed using real object images.

grasp $\hat{g}$, the encoder is responsible for mapping each pair $(N_f, \hat{g})$ to a latent space z and the decoder reconstructs the grasp $\hat{g}$ through z . To guarantee a normal distribution of the latent space, the Kullback-Leibler divergence defined by $\mathcal{D}_{KL}$ is used between the output of the encoder $Q(\cdot|\cdot)$ and a normal distribution $\mathcal{N}(0, I)$, such that the cost function is given by

$$\mathcal{L}_{\text{vae}} = \sum_{z \sim Q, g \sim \tilde{G}} \mathcal{L}(\hat{g}, g) - \alpha \mathcal{D}_{KL}[Q(z | N_f, g), \mathcal{N}(0, I)]. \quad (4.2)$$

The Kullback-Leibler divergence, denoted $D_{KL}(P||Q)$, is a statistical distance, and measure how the probability distribution P is different from the probability distribution Q .

The *GraspNet* evaluator module associates a probability of success to each grasp, observing the partial point cloud of the object N_f such that $P(S | g, N_f)$. It was trained using successful and failed grasps. The cost function applied in the training of this module is given by

$$\mathcal{L}_{\text{evaluator}} = -(y \log(S) + (1 - y) \log(1 - S)), \quad (4.3)$$

where S is the success probability of the inferred grasp and y is the binary label representing the *ground-truth*.

In addition, there is a grasp refinement process, which applies a grasp shift, given by Δg , to increase the chances of successful grasps, so that

$$P(s = 1 | g + \Delta g) > P(s = 1 | g). \quad (4.4)$$

During grasp inference, the encoder Q is removed, and the latent values z are sampled from $\mathcal{N}(0, I)$.

4.6 COLLISION CHECK

It is important to note that the 6D grasping generator was trained using Simulated Grasp Trials (SGTs), considering objects of simple geometry such as bowls, mugs, and boxes.

Nevertheless, the objects employed for testing this grasp generator in this work were more complex, although it is still possible to achieve good grasping results as it generalizes well for new objects. Therefore, the 6D grasping generator applied in the grasping pipeline is not optimized to generate grasps for small objects. Consequently, it may take seconds to find a grasp for small objects.

The 6D grasp generator applied is not capable of analyzing the workspace around the selected object to avoid collisions. To mitigate this problem, a new collision detection system was developed to discard grasps in collision with the environment by using the point cloud of the objects in the workspace. A simplified mesh of the Robotiq 2F-140 (Figure 4.7b) was created to verify collisions with the workspace. Simplified meshes are preferred to reduce the computational cost of the collision check. For every new grasp generated by GraspNet it is verified if this mesh collides with the point cloud of any object in the workspace. To check for collisions, it is calculated the signed distance between each point of the workspace's point cloud and the boundaries of Robotiq 2F-140 collision mesh in the metric space, such that

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega \\ -d(x, \partial\Omega) & \text{if } x \in \Omega^c \end{cases}, \quad (4.5)$$

where $\partial\Omega$ denotes the boundary of Ω for any $x \in X$. The function $f(x)$ takes on different values depending on whether x is inside Ω or in its complement Ω^c . The distance is defined as the infimum

$$d(x, \partial\Omega) := \inf_{y \in \partial\Omega} d(x, y). \quad (4.6)$$

where $d(x, y)$ defines the distance $d(x, \partial\Omega)$ from a point x to the boundary $\partial\Omega$ of a set Ω as the infimum $\inf_{y \in \partial\Omega}$ of the distances $d(x, y)$ between x and all points y in the boundary $\partial\Omega$. This formula essentially calculates the shortest distance from x to any point on the boundary of the set Ω .

Figure 4.6 exemplifies this statement. Figure 4.7 shows the workspace used for testing (Figure 4.7a), a collision-free grasp (Figure 4.7b), and a identified collision between the gripper collision mesh and the workspace point cloud (Figure 4.7c).

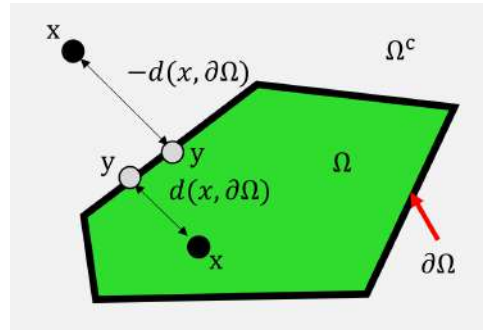


Figure 4.6: Signed distance used to verify collisions between the point cloud and the Robotiq 2F-140 collision mesh.

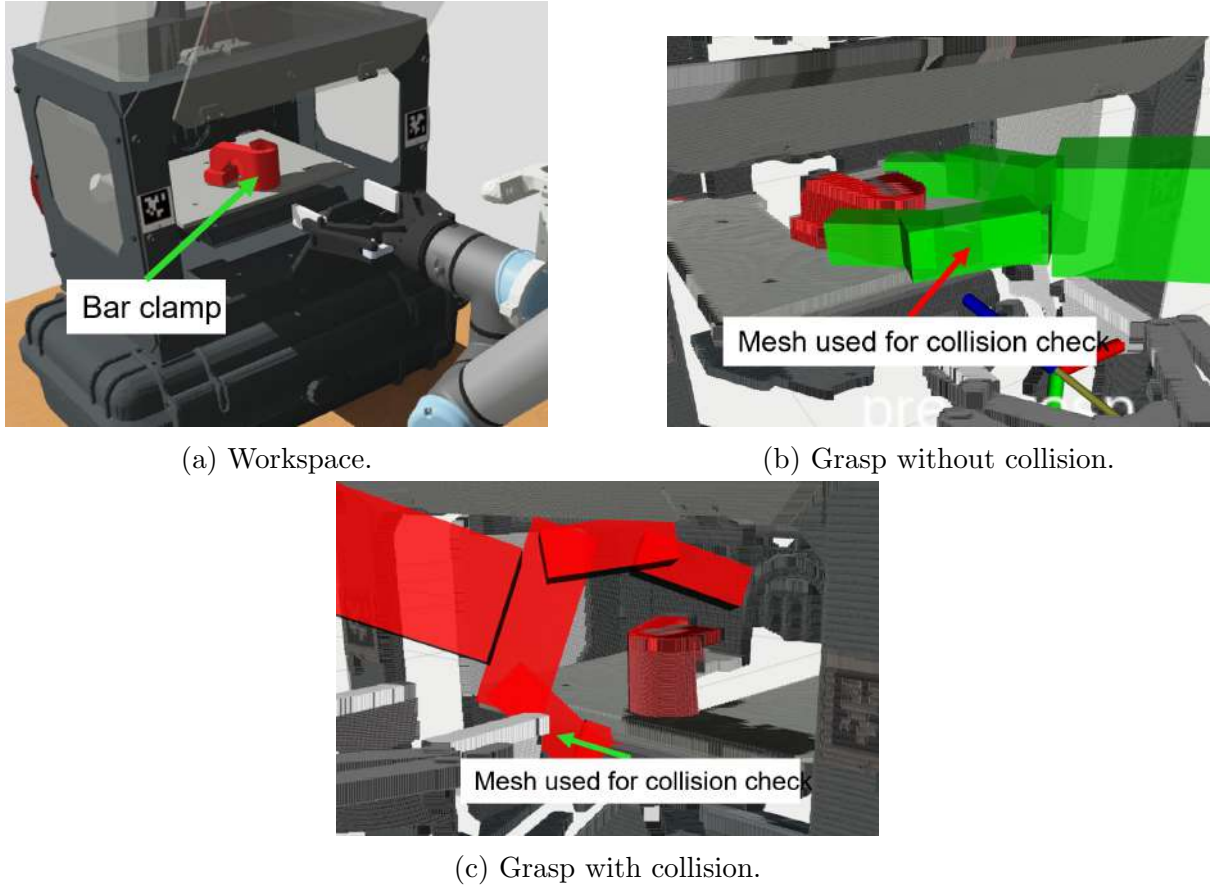


Figure 4.7: Collision check system in an additive manufacturing system. (a) Workspace configuration used for testing (b) There are no points inside the gripper collision mesh (c) There are points of the 3D printer or the object point cloud inside the collision mesh of the gripper.

The point cloud of the workspace, considering the robot pose in front of the printer, has in average 10.000 points in simulation. It is required 0.8 s to calculate the signed distance for this number of points. Figure 4.8 shows the time required to calculate the signed distance for different number of points. For each grasp generated, it is necessary to verify the collision with the environment. Therefore, the collision check system can take considerable time to execute if the point cloud is not downsampled.

4.7 GRASPING PIPELINE

The proposed grasping pipeline comprises each one of the following 13 stages (Figure 4.9):

1. The initial state of the robot $\mathbf{G}_a$ is stored.
2. The image $\mathbf{C}_i$ is obtained by positioning the gripper in the front of the 3D printer.
3. The Mask R-CNN receives an image $\mathbf{C}_i$ as input and generate a mask $\mathbf{M}_r$.

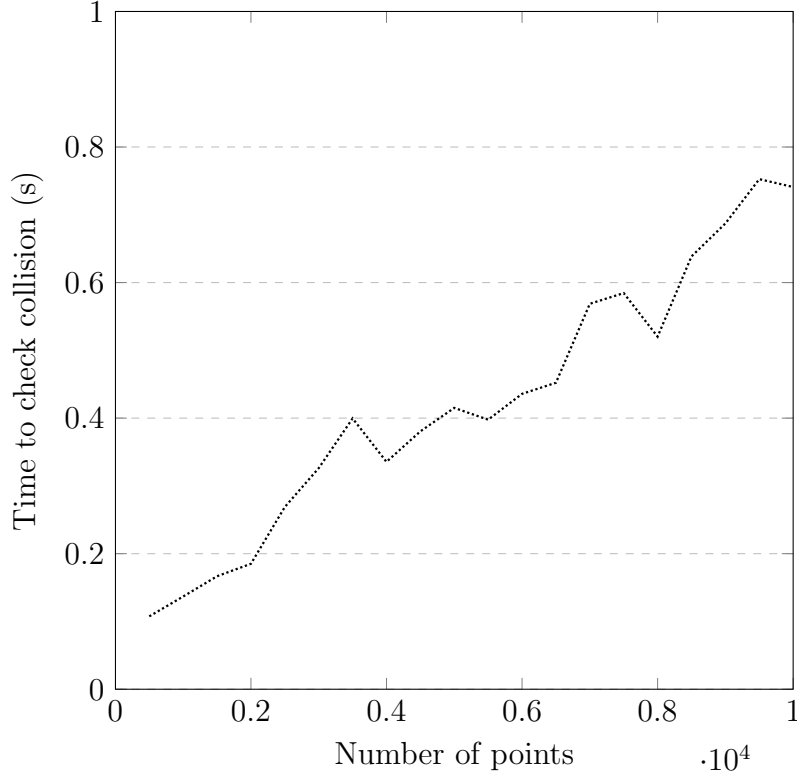


Figure 4.8: Relationship between the number of points of the workspace point cloud and the time required to verify the collision. It is required 0.8s to calculate the signed distance for $1 \cdot 10^4$ points.

4. The mask $\mathbf{M}_r$ is copied to the depth image $\mathbf{I}$.
5. The pixels of the image $\mathbf{I}$ are selected by using the mask $\mathbf{M}_r$. The point cloud $\mathbf{N}_r$ is generated by using a backprojection algorithm in the selected pixels of the image $\mathbf{I}$.
6. A statistical outlier removal filter (ZHOU; PARK; KOLTUN, 2018) is applied in the point cloud $\mathbf{N}_r$ to generate a new point cloud $\mathbf{N}_f$.
7. The printer point cloud $\mathbf{K}_r$ is acquired.
8. $\mathbf{K}_r$ is downsampled to generate a new point cloud $\mathbf{K}_d$.
9. $\mathbf{N}_f$ is used as input to GraspNet to generate a set of 6D grasps $\tilde{\mathbf{G}}_g$.
10. $\mathbf{G}_{gf}$ is selected from $\tilde{\mathbf{G}}_g$ considering the grasps with a score greater than 80% and $\mathbf{O}_{gf}$ closer to $\mathbf{O}_a$, so:

$$\mathbf{G}_{gf} = \mathbf{O}_a - \tilde{\mathbf{O}} < \mathbf{O}_m \quad (4.7)$$

in which $\mathbf{G}_{gf}$ is a set of grasps next to $\mathbf{G}_a$ taking into account a predetermined interval $\mathbf{O}_m$.

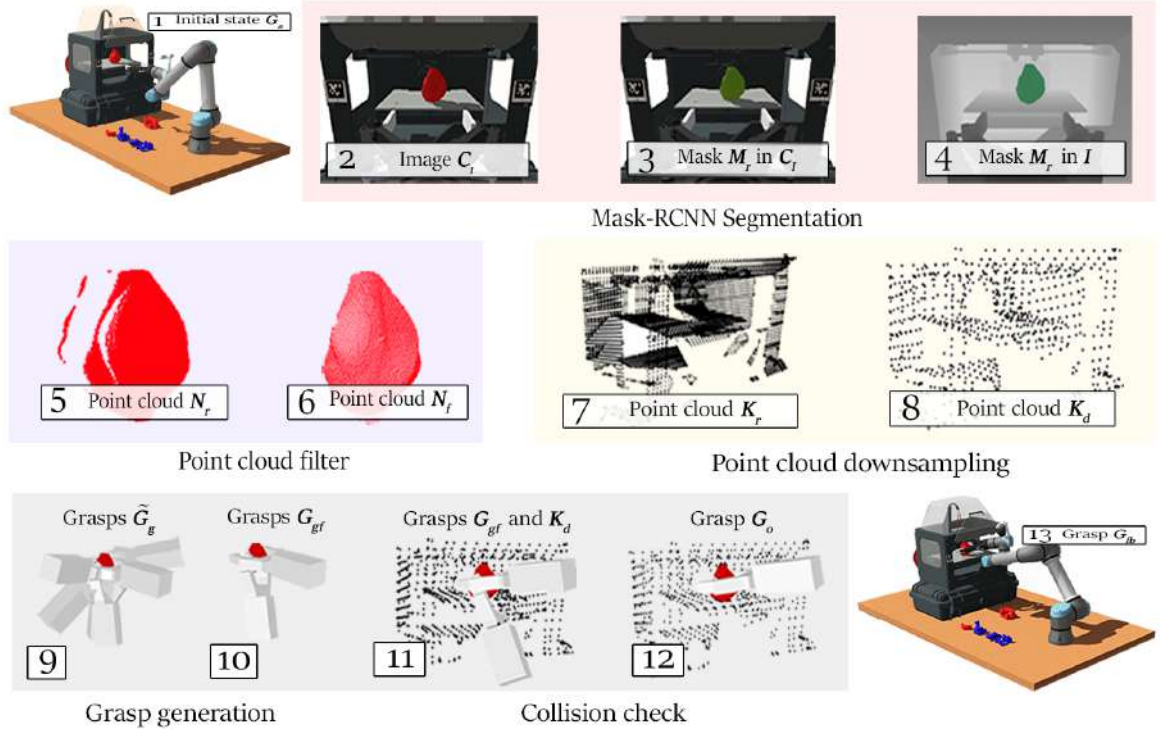


Figure 4.9: Grasping pipeline using GraspNet (MOUSAVIAN; EPPNER; FOX, 2019), Mask R-CNN (HE et al., 2017), and a collision check system based on point clouds and collision meshes. The simulation is performed in Webots (WEBOTS, 2021) using the virtual model of the UR5 robot manipulator, Robotiq 2F-140 and Intel Realsense D435.

11. Each grasp of G_{gf} is rejected if any point of the point cloud K_d lies inside the Robotiq 2F-140 gripper mesh.
12. The grasp without collision G_o is obtained.
13. The final grasp G_{fb} is reached by using a quintic polynomial trajectory planning.

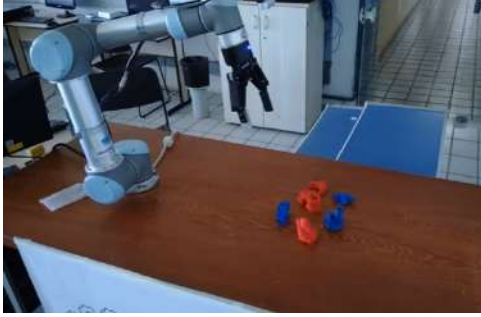
Summarizing, after a detection of an object of interest, a pixel-wise segmentation algorithm is applied to create a mask of the objects using RGB images. This mask is used to segment the object in the depth image and then generate a point cloud by using a back-projection algorithm. The point cloud of the object is used to generate a 6D grasp, and the point cloud of the environment is used to check if the generated grasp collides with obstacles. Besides this, each grasp is discarded if it is not kinematic viable. Finally, the final grasp is reached by using a quintic polynomial trajectory planning.

4.8 EXPERIMENTAL SETUP

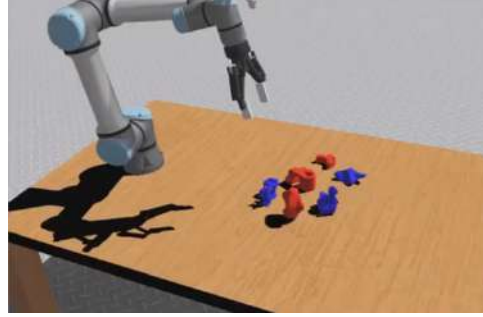
In this work, real and simulated experiments were performed to evaluate the proposed system. The simulated experiments were conducted in Webots (WEBOTS, 2021) inte-

grated to ROS (Robot Operating System) (QUIGLEY et al., 2009) Noetic version. The UR5 robot from Universal Robots (ROBOTS, 2019) fitted with a Robotiq Gripper 2F-140 was used to perform the grasping trials. The Intel Realsense D435 camera was mounted to the wrist of the robot, approximately 23 cm above the fingertips, and inclined 14° towards the gripper. The computations were performed on a desktop running Ubuntu 20.04 with a 2.9 GHz Intel Core i5-10400F, 32 GB RAM, and NVIDIA GeForce RTX 3060 graphic card with 12 GB VRAM.

The object classes used to test the grasping method are shown in Figure 4.2. A grasp is considered successful if the object is moved out of the printer without slipping through the gripper. To obtain the joint angles, given the grasp pose, the TRAC-IK (BEESON; AMES, 2015) was applied. Fig 4.10 shows the real UR5 at the Laboratory of Robotics (UFBA) and the simulated UR5 in Webots.



(a) UR5 robot arm manipulator.



(b) Simulated UR5 robot arm.

Figure 4.10: Real UR5 at the Laboratory of Robotics (UFBA) and simulated UR5 in Webots.

4.9 EXPERIMENTAL RESULTS AND ABLATION STUDY

Some experiments were conducted to better understand the benefits of the integration of the proposed heuristics and the point cloud collision check into the 6D grasping generator. The objects of Figure 4.2 were used in these experiments, and only one object was randomly placed on the 3D printer bed per grasp. In the ablation study, the 6D grasp generator and instance segmentation network are employed in each one of the following cases:

1. Using only stages 1 to 6 of the grasping pipeline of Figure 4.9. The highest score grasp is chosen in this case;
2. Considering stages 1 to 10 of the grasping pipeline of Figure 4.9. A heuristic to filter the generated grasps is applied. The highest score grasp is chosen between the filtered grasps;
3. Employing the complete grasping pipeline of the Figure 4.9.

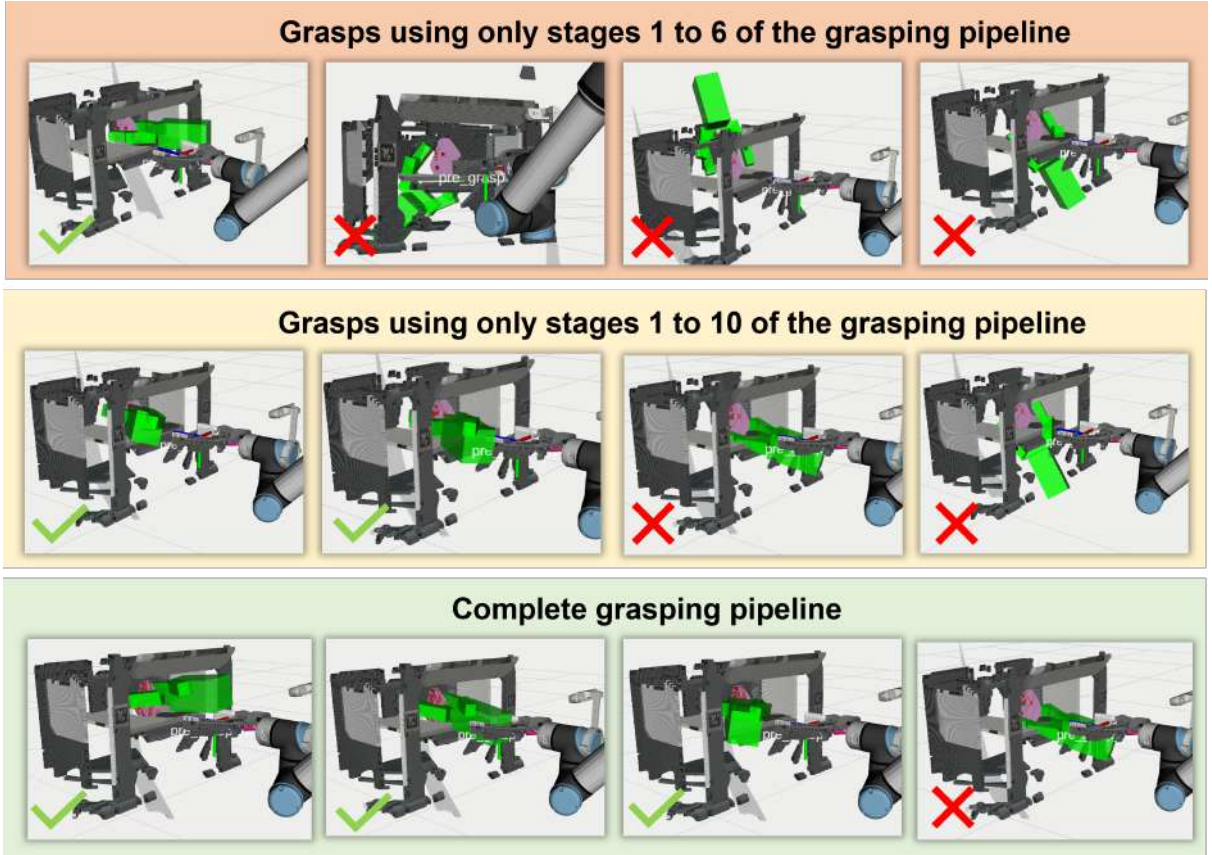


Figure 4.11: Successful and failed grasps generated in each step of the ablation study using the simulated environment.

Figure 4.11 shows examples of successful and failed grasps. Since GraspNet only takes the object point cloud as input and does not see the environment around, it generates grasps that often collide with the printer. When an orientation constraint is added to GraspNet (stage 10 of the grasping pipeline), the grasp feasibility is improved but it does not guarantee that the robot will not collide with the environment, as clearly seen in Figure 4.11. Nevertheless, grasps that are closer to the current end-effector orientation are considered, and grasps far away are ignored. For that, a heuristic already explained in Section 4.7 is applied. When a collision check algorithm using the point cloud of the environment is applied, the successful grasps are considerably improved as seen in Figure 4.12.

To analyze the performance of the proposed method, 20 pick attempts were performed per object. Figure 4.12 shows each grasp trial in simulation, considering the referred ablation study. From the experiments, it can be inferred that GraspNet does not effectively generate a grasp for small objects such as part 1, nozzle, and gear box. Larger objects such as vase, part 3, and bar clamp lead to more stable grasps.

In the ablation study, 20 grasp attempts were performed per object. Table 4.1 shows the performance obtained in simulation for each case of the ablation study. Only 8 (or 7%) grasps were successful when employing the grasping pipeline from stage one to six.

When applying the grasping pipeline from stage one to 10, 18 (or 15 %) grasps were successful. Considering the entire grasping pipeline, 74 (or 62 %) grasps were successful.

Table 4.1: Successful and failed grasping comparison of the ablation study using the simulated environment.

	Grasp Success	Grasp Fails
Entire grasping pipeline	62%	38%
Stage 1 to 10	15%	85%
Stage 1 to 6	7%	93%

It can be noted through Figure 4.13 that GraspNet and Mask R-CNN are fast enough to generate a grasp from 2.6 to 3.4 seconds using the hardware mentioned in Section 4.8. Despite this lower grasping planning time, the grasping success rate is low. When the grasp collision check using the point cloud is employed, the grasp planning time is increased as well as the success rate as described in Figure 4.12.

Despite the success rate of the entire grasping pipeline (62 %), the time consumed to generate a grasp is considerably high, as shown in Figure 4.13. Besides that, the time required to generate a grasp highly depends on the object’s geometry. Small objects such as part 1 demand a significant time to find a feasible grasp, and for bigger objects such as the vase, a grasp is generated faster. The reason is that GraspNet was not trained with small objects as seen in Mousavian, Eppner and Fox (2019). Despite this, Figure 4.14 shows that if we set a time threshold to generate a grasp we would still get a high success rate for some objects. The lower the time threshold is, the lower the success rate because the grasp planner has less time to explore the 6D space.

4.10 COMPARATIVE STUDY

The grasping pipeline presented in this chapter is compared to the grasping pipeline proposed in Chapter 3. In order to perform a fair comparison, the grasping pipeline using GraspNet was modified to integrate the same object recognition CNN used with the GG-CNN in Chapter 3. Therefore, the SSD512-ResNet50-COCO is integrated in series with GraspNet as well. The consequent grasping pipelines for both GraspNet and GG-CNN are shown in Figure 4.15.

Although the GG-CNN and GraspNet differ in terms of grasp orientation restrictions, the environment in which these techniques are contrasted provides fair comparability. The reason is that the test environment does not have space constraints, such as inside a 3D printer. It gives a fair comparison since GG-CNN can only generate grasp in planar surfaces, considering the gripper orthogonal to the surface. The adapted grasping pipeline shown in Figure 4.15 comprises the following stages:

1. The image $\mathbf{C}_i$ is obtained from the Intel Realsense D435 virtual camera in Webots.
2. It executes an attempt to detect the selected object using the image $\mathbf{C}_i$ as an input to the SSD512-ResNet50-COCO.

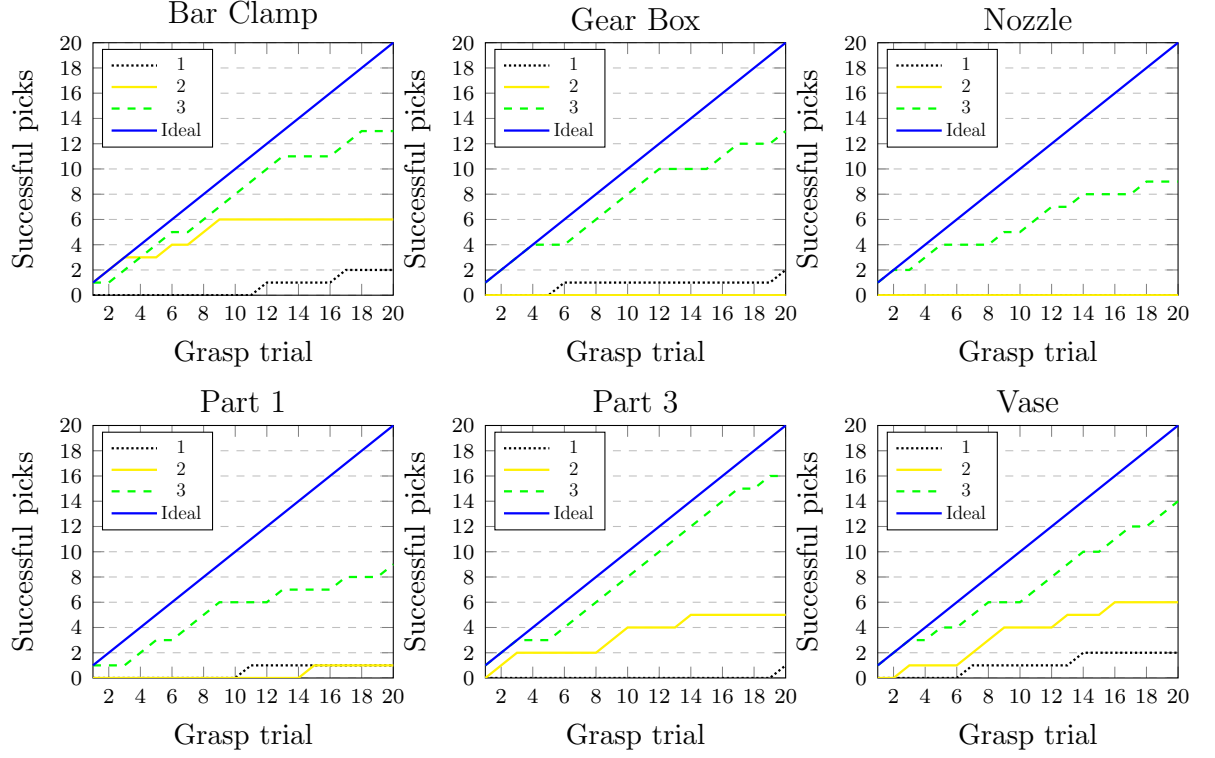


Figure 4.12: Grasps performed in simulation by employing the ablation study referred in Section 4.9. (1) Grasping pipeline from stages 1 to 6, (2) Grasping pipeline from stages 1 to 10, and (3) Complete grasping pipeline. The objects employed in the experiments are shown in section 4.5. The curve 1 for the part Nozzle is under the curve 2. In other words, there were no successful grasps in the 20 attempts.

3. If the object is detected, the bounding box generated by SSD512-ResNet50-COCO is obtained.
4. The bounding box obtained in stage 3 is copied to the depth image I .

From stage 4, the grasping pipeline is divided into two parts. One part is related to the grasping pipeline presented in this chapter and the other is the grasping pipeline presented in Chapter 3. The grasping pipeline 1 (using GraspNet - from stages A1 to F1) comprises the following stages:

A1: The region inside the bounding box in the depth image I is transformed to point cloud using a back projection algorithm.

B1: The point cloud N_r of the detected object is obtained.

C1: The point cloud N_r obtained from stage B1 is used as input to the GraspNet.

D1: A set of grasps $\tilde{G}_g$ is obtained from stage C1.

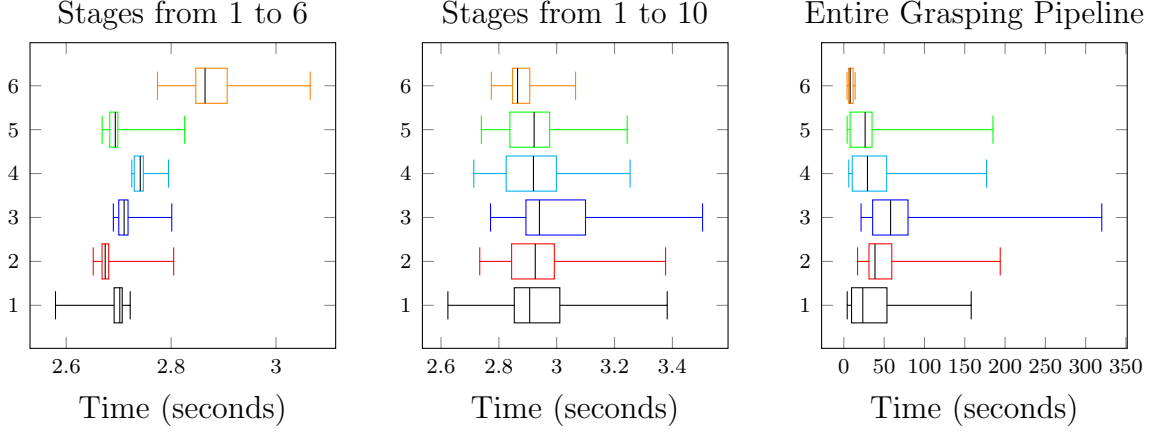


Figure 4.13: Time to generate a grasp for each object used in the experiments. (1) Bar clamp, (2) Gear Box, (3) Nozzle, (4) Part 1, (5) Part 3, (6) Vase. The processing times were based on the ablation study referred in Section 4.9. It is noticed that the entire grasping pipeline is time consuming due to the collision check with the point cloud. However, grasp success is considerably increased as shown in Figure 4.12. Note that the x axis of the graphs have different scales.

E1: From $\tilde{\mathbf{G}}_g$, it is possible to select $\mathbf{G}_{fb}$ that has the highest score generated by GraspNet. Nevertheless, it was noticed that not always a successful grasp has a high score. Therefore, it was developed an heuristic to select $\mathbf{G}_{fb}$, considering the grasp set $\tilde{\mathbf{G}}_g$ that has a score higher than 70% and that $\mathbf{O}_f$ is close to $\mathbf{O}_a$.

$$\tilde{\Phi} = |\mathbf{O}_a - \tilde{\mathbf{O}}_f|, \quad (4.8)$$

where $\tilde{\Phi}$ is the vector differences between the actual gripper orientation $\mathbf{O}_a$ and the target gripper orientation $\tilde{\mathbf{O}}_f$.

The grasp $\mathbf{G}_{fb}$ is obtained as

$$i_d = \max(i(1 - \text{sign}(\tilde{\Phi}_i - \min(\tilde{\Phi})))); \quad i = 0 \dots p, \quad (4.9)$$

where i_d is index of the lowest vector difference Φ and p is the number of grasps generated. Therefore,

$$\mathbf{G}_f = \tilde{\mathbf{G}}(i_d). \quad (4.10)$$

F1: It is verified if any point of the point cloud $\mathbf{K}_d$ lies inside the collision mesh of the Robotiq 2F-140 gripper, considering each grasp of $\mathbf{G}_{gf}$.

The grasping pipeline 2 (using GG-CNN - stages A2 and B2) is applied using the same steps mentioned in Chapter 3 excluding the trajectory planning step to align the object in the GG-CNN area if it is outside this area. The difference between both methods in terms of grasp orientation constraint can be clearly observed in stages D1 and B2. In stage D1, there are multiple 6D grasps generated by the GraspNet. In stage B2, there is

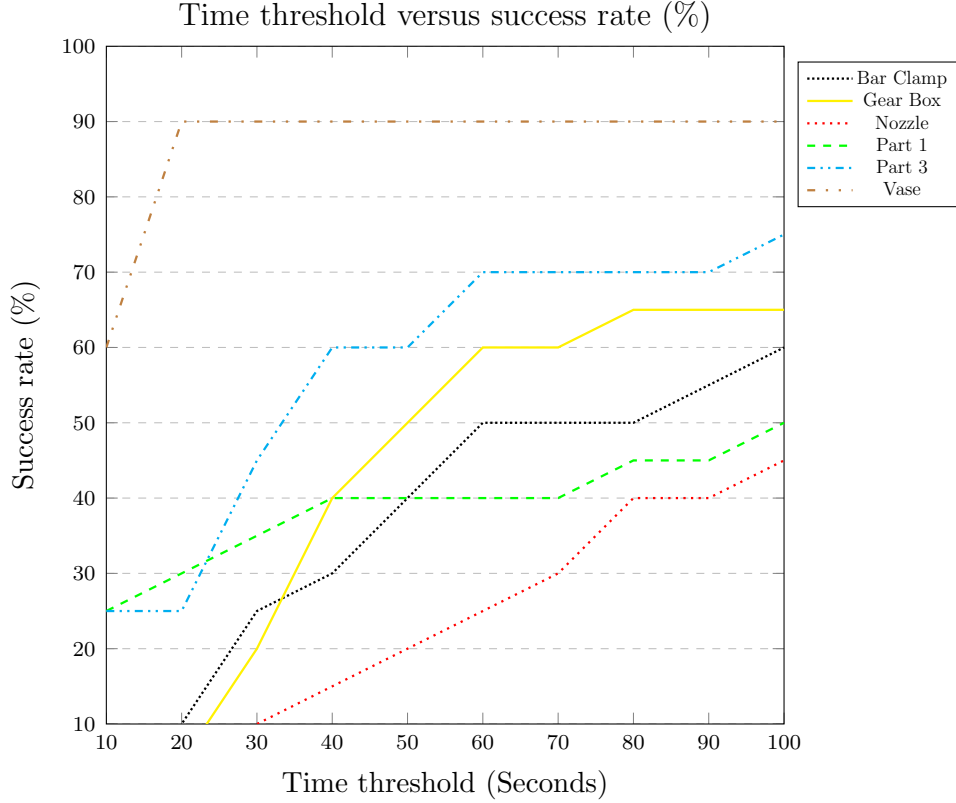


Figure 4.14: Relationship between time threshold and success rate. The lower the time threshold is, the lower is the success rate due to the time consumed by the entire grasping pipeline presented in Section 4.9.

only one planar grasp generated by GG-CNN. Finally, in stage 5, $\mathbf{G}_{fb}$ is acquired through a sequence of homogeneous transformations such as

$$\mathbf{G}_{fb} = t_{RC}(\mathbf{G}_f), \quad (4.11)$$

where t_{RC} represents the homogeneous transformation from the camera's coordinate system and the robot's coordinate system.

It was performed 20 grasp attempts for each object. After each attempt, the objects were randomly placed in the environment. Figure 4.16 reveals some grasps generated by Graspnet and GG-CNN using the pipeline presented in Figure 4.15.

It is verified that GraspNet fails when trying to grasp a small object such as part 1, nozzle, and gear-box. This fact can be clearly observed in Figure 4.12 since the failed grasps for part 1, nozzle, and gear-box are higher. GraspNet achieved a higher success when considering the bar clamp. The GG-CNN had lower performance when considering this object. For part 3, both networks had an equivalent performance.

As shown in Table 4.2, from the 120 grasps performed, GraspNet achieves a grasp success rate of 64%. The SSD512-ResNet50 failed in 9% of these grasp attempts. From the executed grasps, 68% were successful when considering the GG-CNN as a grasp

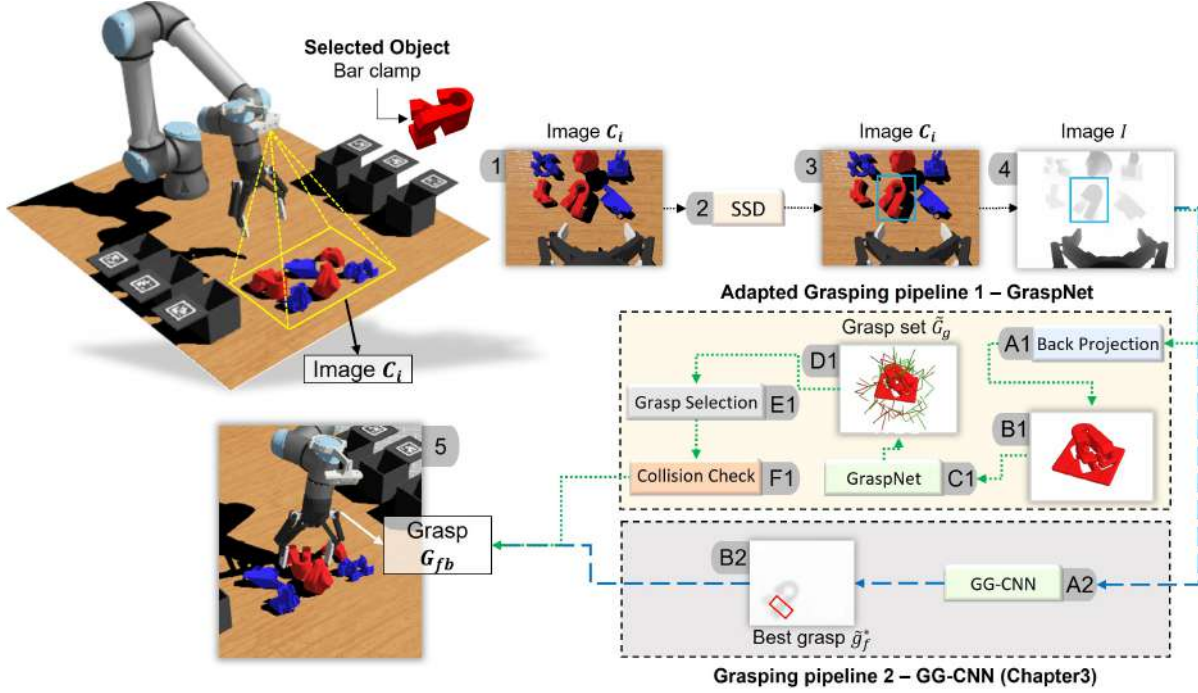


Figure 4.15: Comparison between GraspNet and GG-CNN using two different grasping pipelines. The SSD512-ResNet50-COCO was employed to detect the object of interest in both pipelines. The images depict a simulation performed in Webots.

generator. 11% of the grasps failed due to the SSD512-ResNet50. Regarding processing time, GG-CNN is faster than GraspNet, as shown in Figure 4.17. Nevertheless, GG-CNN only generates a single planar grasp, but GraspNet explores the 6D space and generates a set of grasps instead of a single one. Therefore, GraspNet requires more time to generate a grasp. This is because Graspnet has more network parameters and its performance is directly related to this (MORRISON; CORKE; LEITNER, 2018). Since GG-CNN performs only planar grasps, it cannot be applied in constrained spaces such as the interior of a 3D printer. GraspNet takes advantage of this limitation because it processes the object's point cloud and generates 6D grasps. Figure 4.18 shows the grasp success rate for each object.

4.10.1 Real hardware implementation

The experiments were also performed using the real UR5 robot arm as shown in Figure 4.19. The UR5 robot arm was equipped with the camera Intel Realsense D435 and robotiq gripper 2F-140. As in the simulated experiments, 20 picks were performed per object.

To investigate the grasp repeatability, two poses were determined empirically for each object, see Figure 4.20, and 15 grasp attempts were performed for each pose for each object, see Figure 4.21. It is important to point out that this repeatability test does not have a statistical value, but an empirical analysis. We can infer that the pose of the object's point cloud N_f in relation to the visual sensor generates different levels of

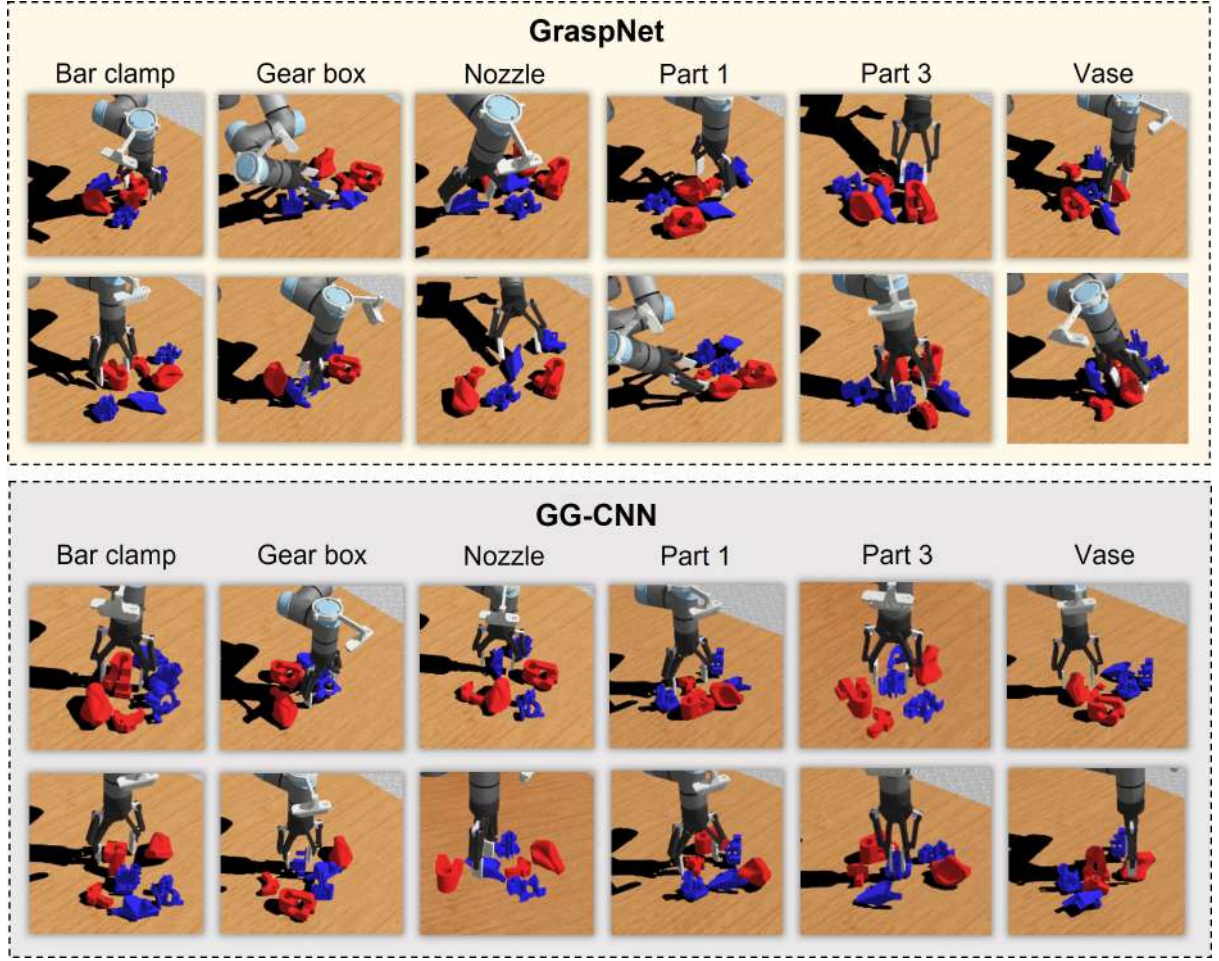


Figure 4.16: Grasps executed by the GraspNet and GG-CNN, employing SSD512-ResNet50 to identify the objects. It was applied the grasping pipeline detailed in Figure 4.15.

Table 4.2: GraspNet and GG-CNN comparison

	GraspNet	GG-CNN
Grasp type	6D	Planar
Grasp width inference	Does not infer the grasp width	Infer the grasp width
Network Input	Point Cloud	Depth Image
Grasp Success Rate	64%	68%
Working Environments	Any Environment	Planar Surfaces

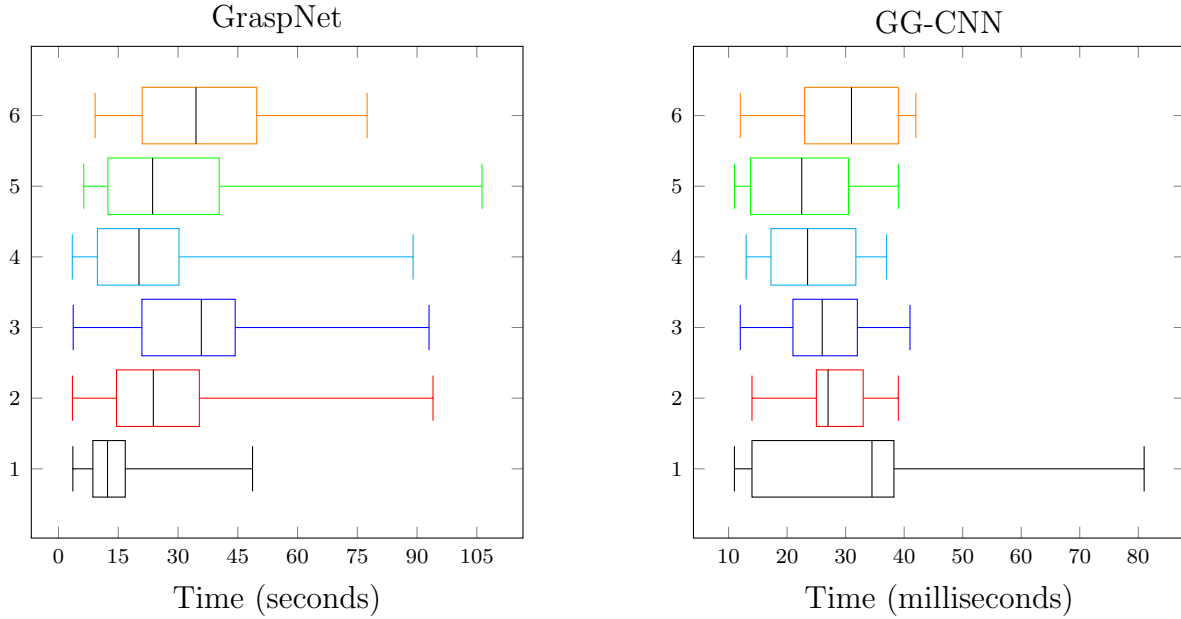


Figure 4.17: Time to generate a grasp for each object used in the experiments. It is important to note the different time scales in the x-axis for the GraspNet (seconds) and GG-CNN (milliseconds). In the graph, the objects are represented as (1) Bar clamp, (2) Gear Box, (3) Nozzle, (4) Part 1, (5) Part 3, (6) Vase. It is important to note that GraspNet requires more time because it generates a set of grasps rather than a single one like GG-CNN does. Besides that, GraspNet has more parameters than GG-CNN and explore the entire 6D space to generate non-planar grasps.

graspability, and consequently the grasping generator produces a set of different grasps, except in cases of objects with a high degree of symmetry (e.g. balls, cubes, rectangles, cylinders, etc).

4.10.2 Failure mode

The failure modes can be classified into the following categories:

- The grasp algorithm (GraspNet) is not optimized to generate grasps for small objects.
- Since the bounding box of the RGB image is transformed to the depth image, it is required to align the RGB and Depth image precisely. Misalignments lead to wrong grasping generation.
- If the environment has low-light conditions, the object cannot be recognized well by the object detector algorithm since it uses RGB images as input. The grasp generator does not suffer from low-light conditions since it uses the object point cloud. Despite that, the grasping algorithm depends on the object recognition network to generate a grasp. Therefore, low-light conditions highly affect the system.

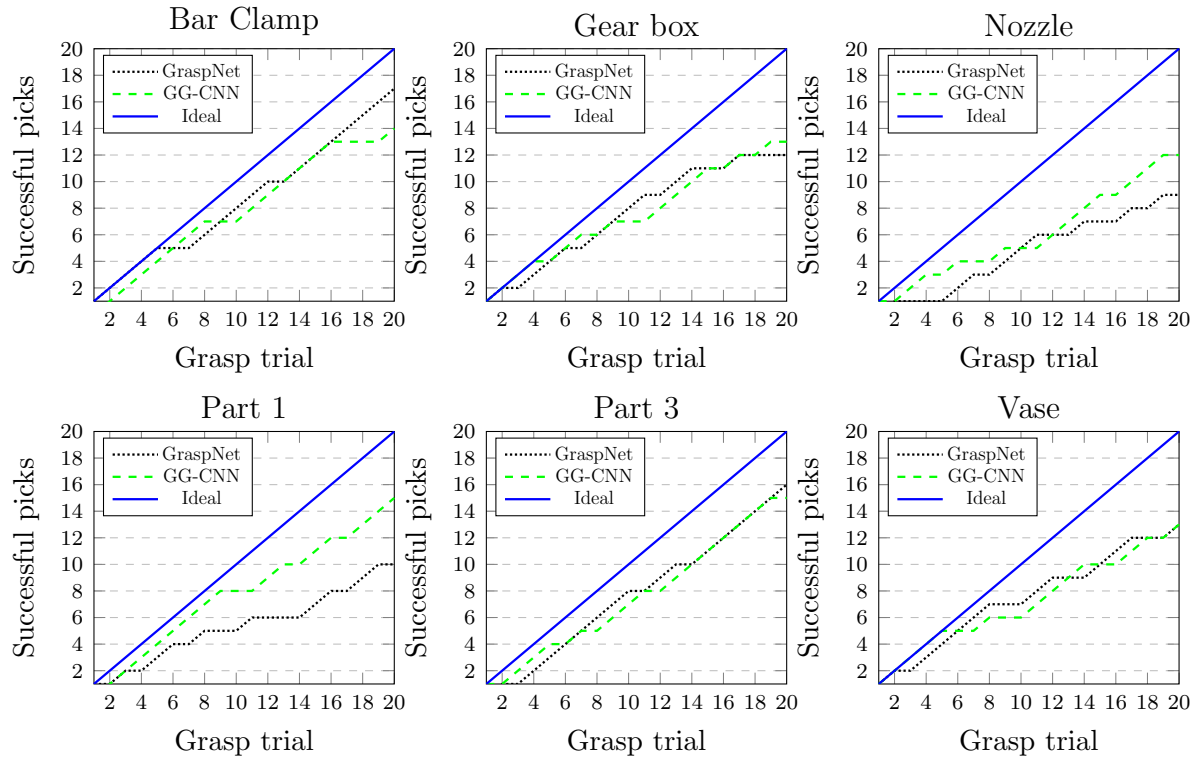


Figure 4.18: Comparison between Graspnet and GG-CNN using the grasping pipeline of the Figure 4.15.

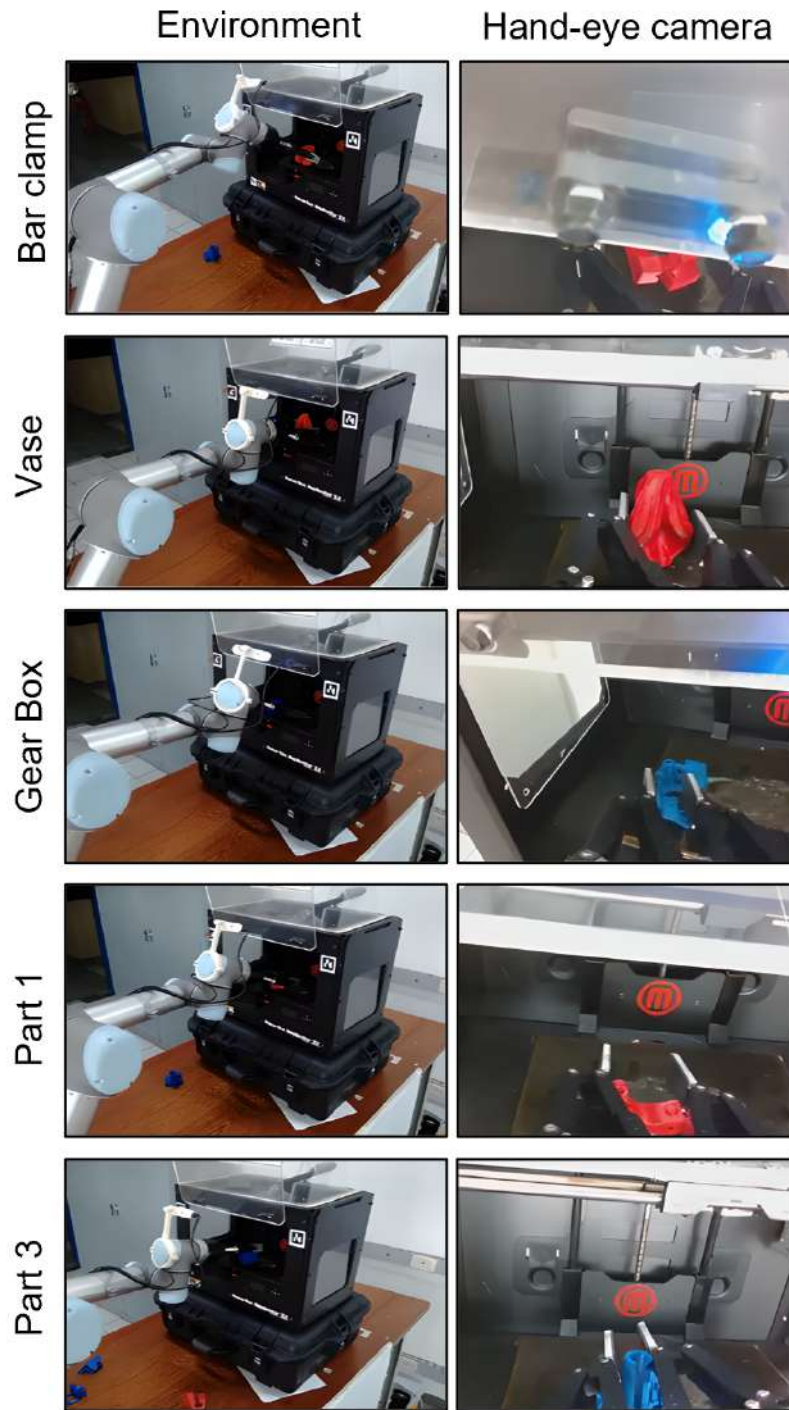


Figure 4.19: Grasps performed in printed objects inside a 3D printer. Video link: [youtube.com/watch?v=APXHeSNuZYU](https://www.youtube.com/watch?v=APXHeSNuZYU)

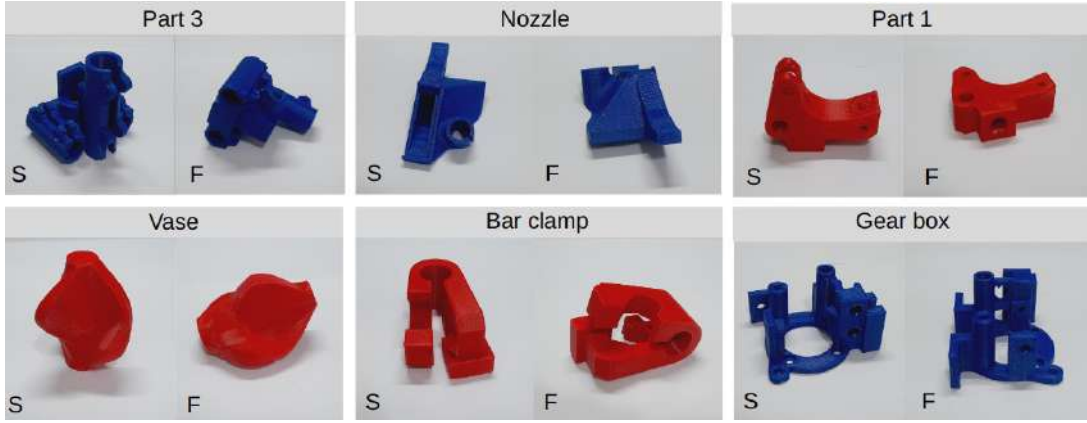


Figure 4.20: Poses S and F for each object.

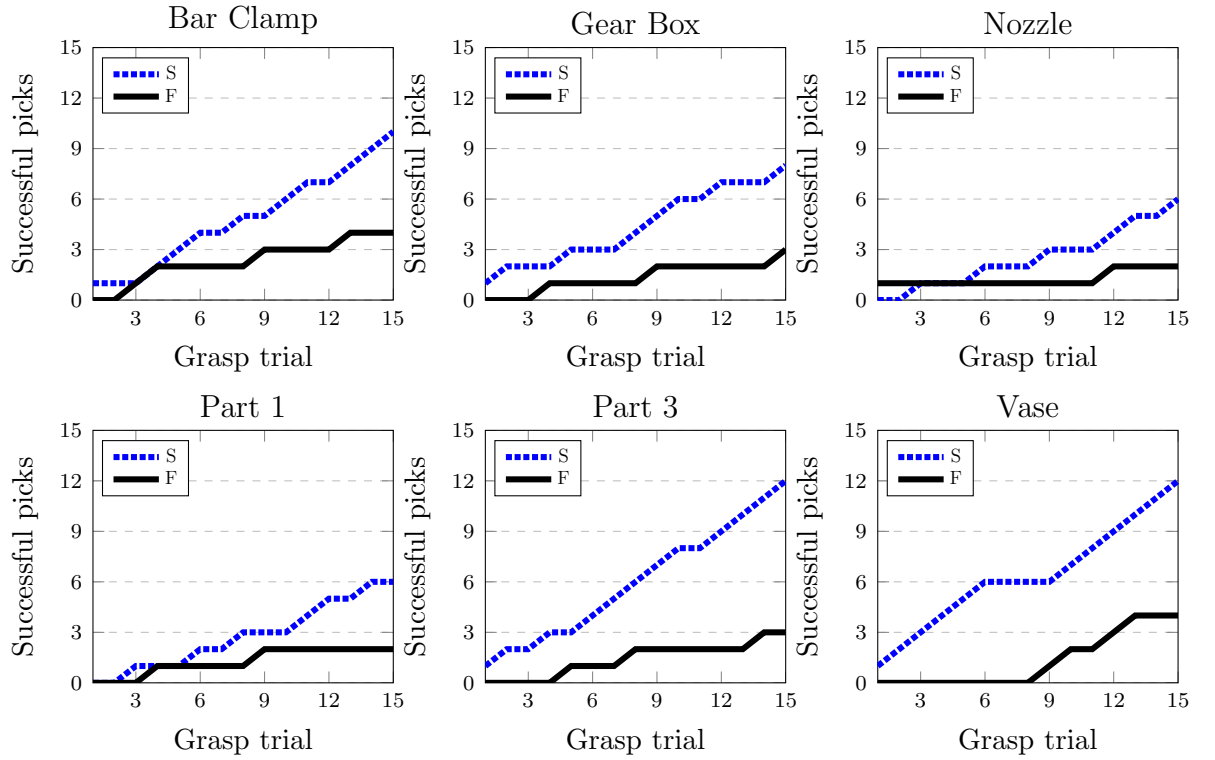


Figure 4.21: Grasps performed by GraspNet using Mask R-CNN as the object detector. To analyze the repeatability, the objects position were kept the same. S means the object pose that is easier to grasp and F is the object pose harder to grasp. These poses were randomly determined.

4.11 CONCLUSION

This thesis has proposed a selective grasping pipeline to generate 6D grasps. It was accomplished by the integration of an object recognition and instance segmentation method, a 6D grasping generator, and a collision detection system based on point clouds. An extensive analysis of experimental results is provided, involving an ablation study, computational cost for collision detection, and repeatability, applied to additive manufactured objects in a complex environment.

The proposed grasping pipeline generates multiple feasible grasps per object. The variety of grasps produced makes it possible to analyze several viable kinematic solutions, and eliminate those that are in collision in the robot's volumetric space.

The main advantage of this solution comes from the integration of important functionalities for grasping systems: (i) selective grasp, the system can grasp and identify the target objects; (ii) segmentation and statistical outlier removal filter to generate object's point cloud in complex environments; (iii) generation of ranked collision-free grasps. The system with such functionality can be easily adapted to other applications, such as selective pick and place in unstructured environment, selective bin picking, among others.

In future work we consider a detailed investigation of the grasping efficiency in small printed objects and the object recognition training process improvement. The 6D grasp generator had lower performance when considering small objects. This is even more noticeable when considering a constrained space such as inside the 3D printer. In view of the application mentioned in this chapter, we do not perceive a significant issue with allocating additional time to compute the optimal grasp, given that we possess the 3D shape of all printed objects as well as the 3D printers and the grasp computation can occur concurrently with the parts' printing. Nevertheless, incorporating a pre-processing step to optimize the voxel representation prior to inputting it into the grasping network is an interesting prospect.

Chapter 5

CONCLUSION

This thesis is dedicated to the review of the state-of-the-art grasping methods and the development of new grasping pipeline strategies using Deep Learning. Recent research shows the scalability of these methods in the real world with a good confidence level. Despite that, there are still many issues to be addressed until it can be reliably deployed in real scenarios.

Chapter 2 presents a deep review of robotic grasping methods by analyzing each aspect of the distinct methods since it can highly affect grasping performance. For each aspect, a comparison is made, and further considerations are given. This comprehensive review serves as the foundation for understanding the state-of-the-art in robotic grasping, providing valuable insights into the strengths and weaknesses of different approaches. By dissecting the key components of these methods, readers can gain a clear understanding of the factors that influence the success of robotic grasping systems.

Chapter 3 presents a two-stage cascade system using the GG-CNN (MORRISON; CORKE; LEITNER, 2018), and modified versions of the SSD (LIU et al., 2016). The proposed system is comprised of both GG-CNN and SSD integrated in series. It is assumed that the objects are located on a flat surface, and the grasping is performed by a parallel-jaw gripper mounted with an RGB+D camera. The proposed grasping pipeline is able to perform grasp on objects of interest. Simulation results have proven the high scalability of this method in real-world scenarios. Despite the results, this method is not able to perform grasps in six dimensions, and only generates a single grasp per object.

Chapter 4 introduces a novel selective grasping pipeline for generating six-dimensional (6D) grasps through the integration of object recognition, instance segmentation, 6D grasping generation, and collision detection based on point clouds. Extensive experimental analyzes, including ablation studies, computational cost assessments for collision detection, and repeatability tests, are conducted on additive manufactured objects within complex environments. The pipeline produces multiple feasible grasps per object, allowing the analysis of various kinematic solutions while ensuring collision avoidance. Its key strengths lie in selective grasping, object segmentation, statistical outlier removal, and ranked collision-free grasp generation, making it adaptable to applications as selective pick-and-place operations in unstructured settings and bin picking. Real experiments were performed using an UR5 Robot Arm Manipulator, an RGB+D camera Intel RealSense D435, and the gripper Robotiq 2F-140. A comparison between the pipeline presented in Chapter 3 and the proposed pipeline in Chapter 4 is also presented, highlighting the advantages and disadvantages of each method, using the same set of objects.

5.1 MAIN CONTRIBUTIONS

- Development of a selective grasping pipeline designed to generate four-dimensional (4D) grasps by integrating object recognition and 4D grasping generation based on

depth images;

- Development of a selective grasping pipeline designed to generate six-dimensional (6D) grasps by integrating object recognition, instance segmentation, 6D grasping generation, and collision detection based on point clouds;
- Extensive experimental analysis, including ablation studies, evaluation of computational cost for collision detection, and repeatability testing, applied to additive manufactured objects within a complex environment;
- Potential adaptability to various applications, such as selective pick-and-place operations in unstructured environments and selective bin picking; and
- Detailed comparison between 4D and 6D grasp generators using real hardware and simulation environments.

5.2 FUTURE WORKS

The development of this thesis and further research in the robotic grasping field indicated that it is worthwhile to extend the research in the following aspects:

- Conducting a more in-depth exploration of grasping efficiency, especially for small printed objects. This includes refining the training processes for object recognition, with a specific emphasis on optimizing point cloud representations to enhance performance. Special attention should be given to scenarios involving constrained spaces like those found in 3D printing environments;
- Integrating gripper geometry as an input parameter for the grasping generator network to facilitate precise collision avoidance with the environment. This consideration can significantly improve the safety and accuracy of grasping actions;
- Investigating techniques for dimensionality reduction within the 6 degree-of-freedom (6DOF) space to simplify network training;
- Exploring the application of shape completion methods to enhance grasping performance, particularly when dealing with objects that are partially or fully occluded;
- Investigating the feasibility of direct point cloud segmentation, as opposed to relying exclusively on masks generated from RGB or Depth images, without compromising overall performance;
- Implementing orientation and reach constraints within the grasping generation process, tailored to the specific requirements of the robot in use. This customization can ensure that the grasping actions are aligned with the capabilities and limitations of the robotic system.

BIBLIOGRAPHY

- ARRAIS, R. et al. Application of the open scalable production system to machine tending of additive manufacturing operations by a mobile manipulator. In: SPRINGER. *EPIA Conference on Artificial Intelligence*. [S.l.], 2019. p. 345–356.
- BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 39, n. 12, p. 2481–2495, 2017.
- BEESON, P.; AMES, B. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In: IEEE. *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. [S.l.], 2015. p. 928–935.
- BEKIROGLU, Y. et al. Benchmarking protocol for grasp planning algorithms. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 315–322, 2019.
- BOHG, J. et al. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, IEEE, v. 30, n. 2, p. 289–309, 2013.
- BOTTAREL, F. et al. Grasp 1.0: Grasp is a robot arm grasping performance benchmark. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 836–843, 2020.
- BRADSKI, G. The opencv library. *Dr Dobb’s J. Software Tools*, v. 25, p. 120–125, 2000.
- BRADSKI, G. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- BREYER, M. et al. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. *arXiv preprint arXiv:2101.01132*, 2021.
- CALLI, B. et al. Guest editorial: Introduction to the special issue on benchmarking protocols for robotic manipulation. *IEEE Robotics and Automation Letters*, IEEE, v. 6, n. 4, p. 8678–8680, 2021.
- CALLI, B. et al. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 36, n. 3, p. 261–268, 2017.
- CALLI, B. et al. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, IEEE, v. 22, n. 3, p. 36–52, 2015.

- CHANG, A. X. et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- CHEN, L.-C. et al. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.
- CHONG, L.-Y.; TEOH, A. B. J.; ONG, T.-S. Range image derivatives for grcm on 2.5 d face recognition. In: *Information Science and Applications (ICISA) 2016*. [S.l.]: Springer, 2016. p. 753–763.
- CIOCARLIE, M. et al. Towards reliable grasping and manipulation in household environments. In: SPRINGER. *Experimental Robotics*. [S.l.], 2014. p. 241–252.
- COLLINS, J. et al. Benchmarking simulated robotic manipulation through a real world dataset. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 1, p. 250–257, 2019.
- CORRELL, N. et al. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 15, n. 1, p. 172–188, 2016.
- COSTA, F. S. et al. Fasten iiot: An open real-time platform for vertical, horizontal and end-to-end integration. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 19, p. 5499, 2020.
- DEPIERRE, A.; DELLANDRÉA, E.; CHEN, L. Jacquard: A large scale dataset for robotic grasp detection. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2018. p. 3511–3516.
- DIANKOV, R.; KUFFNER, J. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, v. 79, 2008.
- DOWNS, L. et al. Google scanned objects: A high-quality dataset of 3d scanned household items. In: IEEE. *2022 International Conference on Robotics and Automation (ICRA)*. [S.l.], 2022. p. 2553–2560.
- EPPNER, C.; MOUSAVIAN, A.; FOX, D. Acronym: A large-scale grasp dataset based on simulation. In: IEEE. *2021 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2021. p. 6222–6227.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International journal of computer vision*, Springer, v. 88, n. 2, p. 303–338, 2010.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, jun. 2010.
- FANG, H. et al. Transcg: A large-scale real-world dataset for transparent object depth completion and a grasping baseline. *IEEE Robotics and Automation Letters*, IEEE, v. 7, n. 3, p. 7383–7390, 2022.

- FANG, H.-S. et al. Graspnet-1billion: A large-scale benchmark for general object grasping. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 11444–11453.
- FUJITA, M. et al. What are the important technologies for bin picking? technology analysis of robots in competitions based on a set of performance metrics. *Advanced Robotics*, Taylor & Francis, v. 34, n. 7-8, p. 560–574, 2020.
- GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587.
- GOSSOW, D. et al. *RViz: ROS 3D Robot Visualizer*. 2020. 3D visualizer for the Robot Operating System (ROS) framework. Disponível em: <https://github.com/ros-visualization/rviz>.
- GUALTIERI, M. et al. High precision grasp pose detection in dense clutter. In: IEEE. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2016. p. 598–605.
- HARA, K.; VEMULAPALLI, R.; CHELLAPPA, R. Designing deep convolutional neural networks for continuous object orientation estimation. *arXiv preprint arXiv:1702.01499*, 2017.
- HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969.
- HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.
- HERNANDEZ, C. et al. Team delft’s robot winner of the amazon picking challenge 2016. In: SPRINGER. *Robot World Cup*. [S.l.], 2016. p. 613–624.
- HOLSCHNEIDER, M. et al. A real-time algorithm for signal analysis with the help of the wavelet transform. In: *Wavelets*. [S.l.]: Springer, 1990. p. 286–297.
- HSU, D.; LATOMBE, J.-C.; MOTWANI, R. Path planning in expansive configuration spaces. In: IEEE. *Proceedings of International Conference on Robotics and Automation*. [S.l.], 1997. v. 3, p. 2719–2726.
- INTEL. *Intel Robotics Open Source Project*. 2019. Accessed: 2020-06-16. Disponível em: <http://wiki.ros.org/IntelROSProject>.
- JAUHRI, S.; LUNAWAT, I.; CHALVATZAKI, G. Learning any-view 6dof robotic grasping in cluttered scenes via neural surface rendering. *arXiv preprint arXiv:2306.07392*, 2023.
- JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

- JIANG, Y.; MOSESON, S.; SAXENA, A. Efficient grasping from rgbd images: Learning using a new rectangle representation. In: IEEE. *2011 IEEE International Conference on Robotics and Automation*. [S.l.], 2011. p. 3304–3311.
- JIANG, Z. et al. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *arXiv preprint arXiv:2104.01542*, 2021.
- JOHNS, E.; LEUTENEGGER, S.; DAVISON, A. J. Deep learning a grasp function for grasping under gripper pose uncertainty. In: IEEE. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2016. p. 4461–4468.
- KAPPLER, D.; BOHG, J.; SCHAAL, S. Leveraging big data for grasp planning. In: IEEE. *2015 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2015. p. 4304–4311.
- KASPER, A.; XUE, Z.; DILLMANN, R. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 31, n. 8, p. 927–934, 2012.
- KATZ, D.; KENNEY, J.; BROCK, O. How can robots succeed in unstructured environments. In: CITESEER. *In Workshop on Robot Manipulation: Intelligence in Human Environments at Robotics: Science and Systems*. [S.l.], 2008.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- KOBER, J.; PETERS, J. Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine*, IEEE, v. 17, n. 2, p. 55–62, 2010.
- KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. [S.l.], 2004. v. 3, p. 2149–2154.
- KONIDARIS, G. et al. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 31, n. 3, p. 360–375, 2012.
- KRAGIC, D.; CHRISTENSEN, H. I. Robust visual servoing. *The international journal of robotics research*, Sage Publications Sage CA: Thousand Oaks, CA, v. 22, n. 10-11, p. 923–939, 2003.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105.

-
- KROEMER, O. B. *Machine Learning for Robot Grasping and Manipulation*. Tese (Doutorado) — Technische Universität, 2015.
- KUMRA, S.; KANAN, C. Robotic grasp detection using deep convolutional neural networks. In: IEEE. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2017. p. 769–776.
- LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998.
- LEE, J. et al. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, The Open Journal, v. 3, n. 22, p. 500, Feb 2018. Disponível em: <https://doi.org/10.21105/joss.00500>.
- LEITNER, J. et al. The acrv picking benchmark: A robotic shelf picking benchmark to foster reproducible research. In: IEEE. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2017. p. 4705–4712.
- LENZ, I.; LEE, H.; SAXENA, A. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 34, n. 4-5, p. 705–724, 2015.
- LEVINE, S. et al. Learning hand-eye coordination for robotic grasping with large-scale data collection. In: SPRINGER. *International Symposium on Experimental Robotics*. [S.l.], 2016. p. 173–184.
- LIN, T.-Y. et al. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 2117–2125.
- LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755.
- LIN, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. 2015.
- LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37.
- LIU, W.; HE, J.; CHANG, S.-F. Large graph construction for scalable semi-supervised learning. 2010.
- LIU, Z. et al. Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation. *arXiv preprint arXiv:2104.11446*, 2021.
- LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3431–3440.

- LUNDELL, J. et al. Constrained generative sampling of 6-dof grasps. *arXiv preprint arXiv:2302.10745*, 2023.
- MAHLER, J. et al. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- MAHLER, J. et al. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 1–8.
- MAHLER, J. et al. Learning ambidextrous robot grasping policies. *Science Robotics*, Science Robotics, v. 4, n. 26, p. eaau4984, 2019.
- MAHLER, J. et al. Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 15, n. 4, p. 1440–1442, 2018.
- MAHLER, J. et al. 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: IEEE. *2016 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2016. p. 1957–1964.
- MAITIN-SHEPARD, J. et al. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In: IEEE. *2010 IEEE International Conference on Robotics and Automation*. [S.l.], 2010. p. 2308–2315.
- MIAO, Z. et al. Insights and approaches using deep learning to classify wildlife. *Scientific reports*, Nature Publishing Group, v. 9, n. 1, p. 1–9, 2019.
- MISHRA, B.; SCHWARTZ, J. T.; SHARIR, M. On the existence and synthesis of multifinger positive grips. *Algorithmica*, Springer, v. 2, n. 1, p. 541–558, 1987.
- MNYUSIWALLA, H. et al. A bin-picking benchmark for systematic evaluation of robotic pick-and-place systems. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 1389–1396, 2020.
- MORGAN, A. S. et al. Benchmarking cluttered robot pick-and-place manipulation with the box and blocks test. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 454–461, 2019.
- MORRISON, D.; CORKE, P.; LEITNER, J. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*, 2018.
- MORRISON, D.; CORKE, P.; LEITNER, J. Multi-view picking: Next-best-view reaching for improved grasping in clutter. In: IEEE. *2019 International Conference on Robotics and Automation (ICRA)*. [S.l.], 2019. p. 8762–8768.
- MORRISON, D.; CORKE, P.; LEITNER, J. Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*, IEEE, 2020.

MORRISON, D.; CORKE, P.; LEITNER, J. Learning robust, real-time, reactive robotic grasping. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 39, n. 2-3, p. 183–201, 2020.

MOUSAVIAN, A.; EPPNER, C.; FOX, D. 6-dof graspnet: Variational grasp generation for object manipulation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 2901–2910.

MURALI, A. et al. 6-dof grasping for target-driven object manipulation in clutter. In: IEEE. *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2020. p. 6232–6238.

NATARAJAN, S.; BROWN, G.; CALLI, B. Aiding grasp synthesis for novel objects using heuristic-based and data-driven active vision methods. *Frontiers in Robotics and AI*, Frontiers Media SA, v. 8, 2021.

NEWBURY, R. et al. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, IEEE, 2023.

NI, P. et al. Pointnet++ grasping: learning an end-to-end spatial grasp generation algorithm from sparse point clouds. In: IEEE. *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2020. p. 3619–3625.

OLSON, E. Apriltag: A robust and flexible visual fiducial system. In: IEEE. *2011 IEEE International Conference on Robotics and Automation*. [S.l.], 2011. p. 3400–3407.

PAS, A. ten et al. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 36, n. 13-14, p. 1455–1473, 2017.

PINTO, L.; GUPTA, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In: IEEE. *2016 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2016. p. 3406–3413.

PRATTICIZZO, D.; TRINKLE, J. C. Grasping. *Springer handbook of robotics*, Springer, p. 671–700, 2008.

QI, C. R. et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. *ICRA workshop on open source software*. [S.l.], 2009. p. 5.

REDMON, J.; ANGELOVA, A. Real-time grasp detection using convolutional neural networks. In: IEEE. *2015 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2015. p. 1316–1322.

- REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788.
- REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 91–99.
- RIBEIRO, E. G.; GRASSI, V. Fast convolutional neural network for real-time robotic grasp detection. In: *International Conference on Advanced Robotics (ICAR)*. [S.l.: s.n.], 2019.
- RIBEIRO, E. G.; MENDES, R. de Q.; GRASSI, V. Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation. *Robotics and Autonomous Systems*, Elsevier, v. 139, p. 103757, 2021.
- ROBOTIQ. *Robotiq Gripper*. 2020. Accessed: 2020-06-16. Disponível em: <https://robotiq.com/products/2f85-140-adaptive-robot-gripper>.
- ROBOTS, U. *UR5 collaborative robot arm — flexible and lightweight robot arm*. 2019. Accessed: 2019-10-15. Disponível em: <https://www.universal-robots.com/products/ur5-robot/>.
- SATISH, V.; MAHLER, J.; GOLDBERG, K. On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters*, IEEE, v. 4, n. 2, p. 1357–1364, 2019.
- SCHULMAN, J. et al. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 33, n. 9, p. 1251–1270, 2014.
- SCHULMAN, J. et al. Finding locally optimal, collision-free trajectories with sequential convex optimization. In: CITESEER. *Robotics: science and systems*. [S.l.], 2013. v. 9, n. 1, p. 1–10.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- SINGH, A. et al. Bigbird: A large-scale 3d database of object instances. In: IEEE. *2014 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2014. p. 509–516.
- SPONG, M. W. et al. *Robot modeling and control*. [S.l.: s.n.], 2006.
- SUN, J. et al. A model-free 6-dof grasp detection method based on point clouds of local sphere area. *Advanced Robotics*, Taylor & Francis, p. 1–12, 2023.
- SUN, Y. et al. Robotic grasping and manipulation competition: Task pool. In: SPRINGER. *Robotic Grasping and Manipulation Challenge*. [S.l.], 2016. p. 1–18.

-
- SUN, Y. et al. Research challenges and progress in robotic grasping and manipulation competitions. *IEEE robotics and automation letters*, IEEE, v. 7, n. 2, p. 874–881, 2021.
- SUNDERMEYER, M. et al. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In: IEEE. *2021 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2021. p. 13438–13444.
- TOBIN, J. et al. Domain randomization for transferring deep neural networks from simulation to the real world. In: IEEE. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2017. p. 23–30.
- VICENT, J. et al. Flex end-to-end mission performance simulator. *IEEE Transactions on Geoscience and Remote Sensing*, IEEE, v. 54, n. 7, p. 4215–4223, 2016.
- VIERECK, U. et al. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017.
- WANG, Z. et al. Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering*, SAGE Publications Sage UK: London, England, v. 8, n. 9, p. 1687814016668077, 2016.
- WEBOTS. <http://www.cyberbotics.com>. 2021. Open-source Mobile Robot Simulation Software. Disponível em: [⟨http://www.cyberbotics.com⟩](http://www.cyberbotics.com).
- WENG, T. et al. Neural grasp distance fields for robot manipulation. In: IEEE. *2023 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2023. p. 1814–1821.
- WOHLKINGER, W. et al. 3dnet: Large-scale object class recognition from cad models. In: IEEE. *2012 IEEE international conference on robotics and automation*. [S.l.], 2012. p. 5384–5391.
- WU, Z. et al. 3d shapenets: A deep representation for volumetric shapes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1912–1920.
- XIE, S. et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017.
- YANG, J. et al. Object contour detection with a fully convolutional encoder-decoder network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 193–202.
- YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- ZHANG, H. et al. Regrad: A large-scale relational grasp dataset for safe and object-specific robotic grasping in clutter. *IEEE Robotics and Automation Letters*, IEEE, v. 7, n. 2, p. 2929–2936, 2022.

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.