UNIVERSITY OF SÃO PAULO
SÃO CARLOS SCHOOL OF ENGINEERING

Larissa Cassador Casteluci

Artificial data generation pipeline for visual grasping deep learning

São Carlos

2023

Larissa Cassador Casteluci

Artificial data generation pipeline for visual grasping deep learning

Thesis presented to Sao Carlos School of Engineering, for obtaining the Academic Title of Master in Sciences - Graduate Program in Mechanical Engineering.

Research Area: Dynamic and Mecatronics

Supervisor: Prof. Dr. Daniel Varela Magalhães

CORRECTED VERSION

São Carlos

2023

Catalog card prepared by Patron Service at "Prof. Dr. Sergio Rodrigues Fontes" Library at EESC/USP

C349a

Casteluci, Larissa Cassador
Artificial data generation pipeline for visual grasping deep learning / Larissa Cassador Casteluci; Thesis directed by Daniel Varela Magalhães. -- São Carlos, 2023.

Master (Thesis) - Graduate Program in Mechanical Engineering and Research Area in Dynamic and Mechatronics -- São Carlos School of Engineering of the University of São Paulo, 2023.

1. Visual grasp detection. 2. Deep learning. 3. Synthetic Data. 4. Computer vision. I. Title.

Elena Luzia Palloni Gonçalves – CRB 8/4464

FOLHA DE JULGAMENTO

Candidata: Engenheira **LARISSA CASSADOR CASTELUCI.**

Título da dissertação: "Pipeline para geração de dados artificiais para treinamento de redes de preensão robótica"

Data da defesa: 17/04/2023

Comissão Julgadora	**Resultado**
Prof. Associado **Daniel Varela Magalhães** **(Orientador)** (Escola de Engenharia de São Carlos – EESC/USP)	APROVADA
Prof. Associado **Valdir Grassi Junior** (Escola de Engenharia de São Carlos – EESC/USP)	APROVADA
Prof. Dr. **Lucio André de Castro Jorge** (Empresa Brasileira de Pesquisa Agropecuária/EMBRAPA)	APROVADA

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Adriano Almeida Gonçalves Siqueira**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Carlos De Marqui Junior**

ABSTRACT

CASTELUCI, L. C. **Artificial data generation pipeline for visual grasping deep learning**. 2023. 81p. Master Thesis - São Carlos School of Engineering, University of São Paulo, São Carlos, 2023.

The rise of deep learning algorithms in academia has changed the area of robotic grasping. Before, methods involving analytical analysis and grasping modelling were the most common strategies. However, deep learning strategies have become recently more prevalent. They have presented incredible results in the last decade. However, they present disadvantages of their own. A major drawback is that they require large amounts of representative data to be trained on. For specific applications, a specific dataset with custom targets is required. But generating data for robotic grasping is not an easy task. It is more challenging than creating datasets for classification or object detection problems, since it requires lab experiments. Manual acquisition of this data can be time-consuming. In this context, the generation of synthetic data using rendering and simulation tools can be a viable solution. This strategy, on the other hand, also has its own set of problems. The most relevant is the reality gap, i.e. the intrinsic difference between reality and simulated data. There are a few techniques developed to mitigate this problem, such as domain randomization and photorealistic data. We provide a tool that allows the creation of datasets for robotic grasping for a configurable set of targets. We compare in a real life scenario a neural network trained on this custom dataset and compare results with the same network trained on a state-of-the-art dataset and show that our tool creates viable datasets that neural networks can be trained on and produce suitable results.

Keywords: Visual Grasp Detection, Deep Learning, Synthetic Data, Computer Vision.

RESUMO

CASTELUCI, L. C. **Pipeline para geração de dados artificiais para treinamento de Redes de Preensão Robótica**. 2023. 81p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

O advento de algoritmos de aprendizagem profunda mudou o panorama da área de preensão robótica. Se antes a área se focava em métodos analíticos e modelagem de preensão para planejar e analisar a qualidade de preensão de objetos, hoje esse encargo recaí sobre os algoritmos de inteligência artificial. Embora esses algoritmos tenham apresentado resultados surpreendentes na última década, eles também possuem desvantagens se comparados a técnicas de modelagem de preensão. A principal desvantagem é a necessidade de treinamento da rede em um conjunto de dados amplo e representativo do problema. Para aplicações especializadas, pode ser necessário um *dataset* customizado com objetos específicos. Mas a criação desses *datasets* não é uma tarefa fácil. Para a área de preensão robótica, a geração desses dados é mais complexa que a geração de conjuntos de dados para classificação e detecção de objetos, uma vez que requer experimentos em laboratórios. A obtenção desses dados de forma manual pode ser demorada e ser suscetível a erros. Nesse contexto, a geração de dados de forma artificial, por renderização de dados e simulação, se torna uma alternativa viável para geração de dados para treinamento de redes. Por sua vez, essa estratégia também apresenta os seus problemas. A principal entre elas é denominada de *reality gap*. Ou seja, é a diferença que existe em dados simulados e dados obtidos na realidade. Tentando mitigar esse efeito, foram elaboradas técnicas para compensar essa diferença. Abordagens relevantes nesse sentido são a geração de dados foto realísticos e a *domain randomization*. Nesse trabalho, é disponibilizada uma ferramenta para a criação de *datasets* de preensão robótica com a configuração de objetos. Foram realizados experimentos que comparam os resultados de uma rede neural treinada nesse *dataset* customizado com a mesma rede treinada em um *dataset* estado da arte em ambiente de laboratório. Os resultados demostram que a ferramenta é capaz de gerar *datasets* viáveis para o treinamento de redes neurais, e que elas produzem resultados viáveis.

Palavras-chave: Detecção de Preensão com Visão. Aprendizagem Profunda. Dados Sintéticos. Visão Computacional.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
BOP	Benchmark for 6D Object Pose Estimation
CNN	Convolutional Neural Network
DOF	Degrees of Freedom
GAN	Generative Adversarial Network
GG-CNN	Generative Grasping Convolutional Neural Network
EGAD	Evolved Grasping Analysis Dataset
IoU	Intersection over Union
OpenCV	Open-Source Computer Vision
PnP	Perspective-n-Point
PPF	Point-Pair Features
PTP	Point To Point
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
ROS	Robot Operating System
TCP	Transmission Control Protocol
SDK	Software Development Kit
URDF	Unified Robotics Description Format
V-HACD	Volumetric Hierarchical Approximate Convex Decomposition
VSD	Visual Surface Discrepancy
YCB	Yale-CMU-Berkeley

LIST OF SYMBOLS

Pinhole Camera Model

f_x	Focal distance in axis x
f_y	Focal distance in axis y
c_x	Camera's center position in axis x
c_y	Camera's center position in axis y
γ	Inclination between camera's axes
R	Camera's orientation matrix
t	Camera's translation matrix
w	scale coefficient

GG-CNN Theory

g	grasp
q	quality measure
$\mathbf{p}$	gripper's center position
ϕ	gripper's rotation around z-axis
w	gripper's width
$\tilde{g}$	grasp in image coordinates
$\tilde{\mathbf{s}}$	gripper's center position in image coordinates
ϕ	gripper's rotation in the camera's reference frame
t_{RC}	conversion between the world frame and camera frame
t_{CI}	conversion from Image Space to the Cartesian Space
$\tilde{\mathbf{G}}$	Grasp Map
$\tilde{\mathbf{Q}}$	Quality Measure Map
$\tilde{\Phi}$	Gripper's rotation Map
$\tilde{\mathbf{W}}$	gripper's width map

$\tilde{g}*$	optimal grasp in image coordinates
M	Function that is the optimal solution
I	Image
M_Θ	Approximated Function
$\tilde{G}_\Theta$	Approximated Grasp Map
Θ	Weights
	LINEMOD Precision Metric
m	Precision metric
M	Set of points belonging to the model
$\bar{R}$	Real Rotation
$\bar{T}$	Real Translation
$\hat{R}$	Estimated Rotation
$\hat{T}$	Estimated Translation
k_m	Arbitrary Coefficient
d	Model's Diameter
	BOP Challenge Precision Metric
$\hat{P}$	Estimated Pose
$\bar{P}$	Real Pose
M	Set of points belonging to the model
$\hat{S}$	Estimated Pose Distance Map
$\bar{S}$	Real Pose Distance Map
$\hat{V}$	Estimated Pose visibility mask
$\bar{V}$	Real Pose visibility mask
ρ	Pixel
x_ρ	Point in the world coordinates system that projects ρ
τ	Threshold
e_{VSD}	Calculated Error VSD

CONTENTS

1 INTRODUCTION

1.1 Motivation

Robotics has always been dependent on human explicit trajectory programming. A representative example is the car industry, where repetitive and high-precision jobs are performed by robotic manipulators, mounted along an assembly line. Every move needs to be programmed, and the robots do not have any sort of intelligence on their own. Even with decades of continuous development in this area (HVILSHøJ, 2012), the way robotics is used outside of research has not changed much. This can be seen on the security standards that are commonplace for using this technology in industry (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2018). Which means that robust and reliable algorithms are necessary to change this landscape.

The ability of comprehension and to act according to the environment, specially when humans are involved, is one of the greatest challenges for current robotics (SÜNDERHAUF et al., 2018). To do this reliably, in systems where robots will not cause accidents due to an algorithm or a malfunction, is a vital milestone that has been in the spotlight of robotic's research for a long time. For this to be possible, a robotic system needs the ability to obtain data and extract information about its environment as quickly as possible, so it can adjust accordingly in a reliable manner.

RGB and RGB-D cameras are among the most important sensors in this aspect, obtaining a large amount of data about the environment quickly. However, this technology cannot be used to its full potential if we do not have algorithms that extract the useful information at a fast pace. The area that addresses these questions is called Machine Vision.

1.1.1 Machine Vision

Humans can easily distinguish and recognize objects through vision, with such a precision in object detail and depth information that we can easily stretch our arms and grasp an object with just vision alone. Although this may seem a simple process, it is, in fact, one that developed through millions of years through evolution (RENSINK, 2000). The only way computers can hope to achieve this is to process this data through algorithms that can extract useful information (or features) from this data and use them to make conclusions about what the image is representing. In the last decade, deep learning algorithms have become the most prominent type of algorithm in visual data processing (Zhao et al., 2019).

RGB cameras are perhaps the most common sensor in the world currently. Almost

every laptop and cellphone has one, and development on how to improve the data's quality and extract information has been going strong in the last decade. Data processed from a single camera, however, does not provide information about depth. To circumvent this, more cameras can be employed in the same system, allowing for rich depth information. These systems are called stereo cameras.

Given these sensors, several problems in robotics can be addressed, i.e. the mapping of an environment (DURRANT-WHYTE; BAILEY, 2006; BAILEY; DURRANT-WHYTE, 2006), obstacle avoidance (MATTHIES et al., 2014), and, the area that is relevant for this project, object grasping.

1.1.2 Artificial Data

As data-driven models become more common in different areas, strategies to facilitate training and improve results start to become commonplace. For the creation of the COCO dataset (LIN et al., 2014), even if the dataset was immense, it was still feasible to hand-label the data, since humans can easily locate objects in images and label them. In other areas, labelling things is not as simple. For example, the task to determine a pose of a given object. To create a representative label, a human observer should be able to provide this information with such a precision that an algorithm can learn how to guess poses for this given object. The task of creating such datasets is challenging. In this case, an alternative solution would be to use a 3D renderer software, that has all the data about an object (in this case, a model) at all times. That is, use artificial data to train a deep learning algorithm.

This solution, however, also has its downsides, such as the reality gap. That is, a network that has only been trained in artificial data does not perform well in real-world scenarios. Several works have tried to address this problem. Among them are the use of photorealistic data (MCCORMAC et al., 2016; GAIDON et al., 2016), and Domain Randomization (TOBIN et al., 2017; TREMBLAY et al., 2018a)

1.1.3 Robotic Grasping

Traditionally, the area of robotic grasping was dominated by analytical methods. They focus on modelling the contact behavior between the object and the gripper, usually defining a number of key contact points, where forces and moments are applied. A grasp can be modelled in a number of ways (i.e. frictionless with normal force, frictionless with tangential forces, not frictionless, etc.). For a grasp to be successful, it must in equilibrium, that is, the sum of the forces must be zero. A number of strategies were developed to solve this constraint problem (SAHBANI; EL-KHOURY; BIDAUD, 2012). However, these methods tend not to perform well in real-world scenarios (WEISZ; ALLEN, 2012; MORRISON, 2021). Some of the problems that affect the performance of analytical methods

are: assumption that there is perfect knowledge about the objects and the manipulators, simplifications to models that introduce errors and computationally expensive calculations for real-world applications.

Furthermore, if we want robotic systems that can extract information from the world and act accordingly, we cannot expect that we have precise knowledge about models and contact points, so analytical methods fall short in the context of generalization and unstructured environments. The alternative that has risen strongly in the last decade are data-driven methods.

Data-driven models have become the norm in many research areas. From speech recognition, object recognition, data quality enhancement, and many other areas. The rise of deep learning algorithms has changed the landscape in research. A drawback is that these models require massive amounts of data to converge. In Reinforcement Learning approaches, this means many hours in real-world setups or simulations so that the algorithm can converge to a specific behavior. In Supervised Learning algorithms, this will imply in large amounts of labelled data and many hours to train the network. Hand-labelling this data, or creating a lab setup to train a neural network, can be a very time-consuming endeavor. Besides, it is not a scalable strategy. A solution for this problem that has been gaining traction is the use of artificial data to simulate real-world conditions or try to replicate data (and create labels) where neural networks can be trained on.

Robotics is, due to its very nature, a complex part of engineering. By adding black-box models such as deep learning algorithms to the mix, it becomes increasingly more significant to have reliable methods to train and evaluate the performance of these systems, in a reproducible manner, but still customizable for different applications.

1.2 Goals

1.2.1 Main Goals

Provide an open and customizable pipeline for rendering artificial data for neural networks focused on top-down grasps of a set of objects.

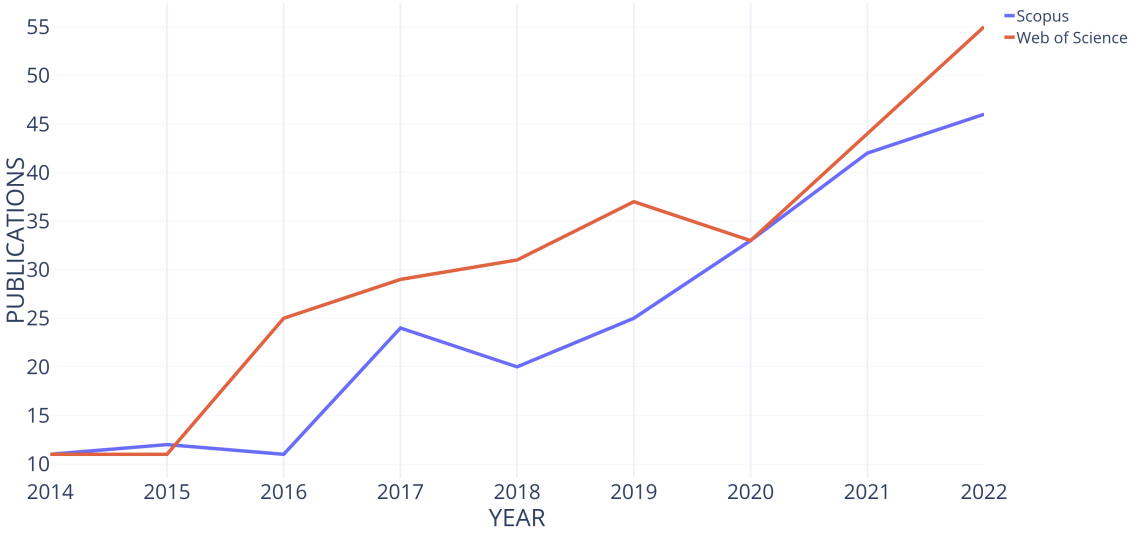
2 LITERATURE REVIEW

2.1 Bibliometrics

We first analyze how relevant the area of robotic grasping has been by measuring its growth in the last years. For this search, we use the databases from Scopus and Web of Science.

We search for the keywords "Visual Grasp Detection" in the fields title, keywords and abstract. All three words must be present to be relevant to this search. The results can be seen in figure 1.

Figure 1 – Results with the keywords "Visual Grasp Detection"



Source: Author

It is noticeable the increase in the number of publications related to this area. A possible reason for this expansion is the use of deep learning techniques that have improved the results involving vision-related problems.

2.2 Camera Modelling

In this section, we provide a brief explanation of both RGB camera and depth (stereo) cameras.

2.2.1 Camera Models

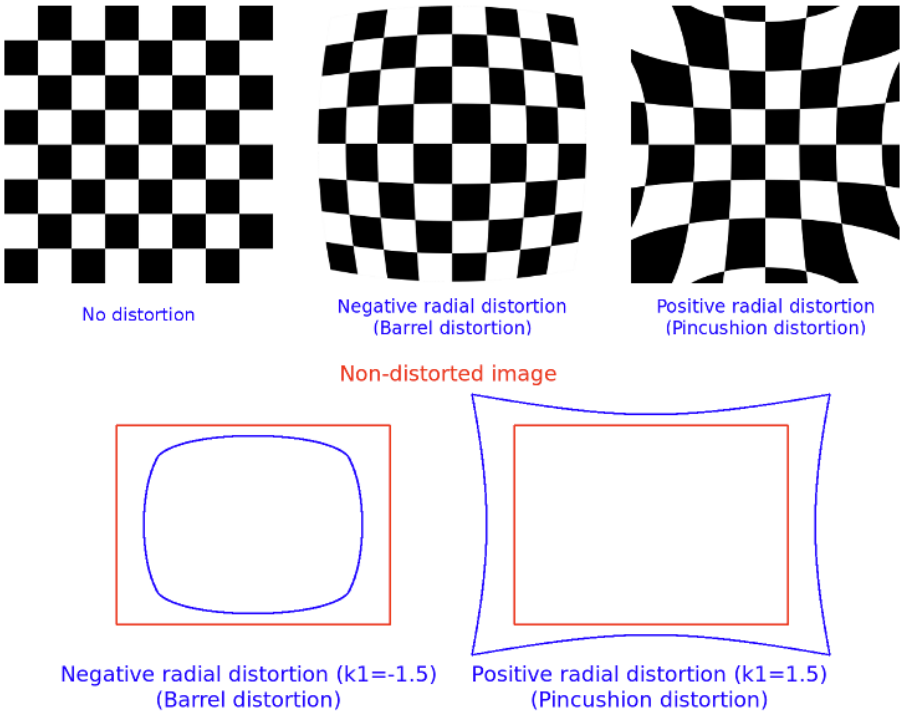
The **Pinhole Camera Model** is the simplest and most specialized among the finite camera models (HARTLEY; ZISSERMAN, 2003). This model assumes that images are formed based on the projection of 3D points in the images' plane. This model assumes a

simple projection from the object to the image's plane. This model takes into consideration intrinsic camera properties (focal distances $< f_x, f_y >$, camera's center position $< c_x, c_y >$ and the inclination between axis γ) as well as external properties, such as translation (matrix t), orientation (matrix R) and the scale w. This is shown in Equation 2.1, where X, Y, Z represent the real coordinates of an object, and x, y, z represent the projection in the image's plane. It is relevant to highlight that this model does not take into consideration the camera's distortions (positive radial, negative radial and tangential distortions).

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R_{3x3}|t_{3x1}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

A more faithful representation of a real captured image can be represented by the **Brown-Conrady Camera Model** (DUANE, 1971). It is the model that OpenCV (OPENCV, 2020) accepts as the default. It consists of a polynomial model to define the radial distortions. In this work, we follow the same convention as in OpenCV (2023), that it is based on Louhichi et al. (2007). Figure 2 shows the kind of distortion that this camera model takes into consideration.

Figure 2 – Distortions in an image obtained by a camera



Source: OpenCV (2023)

2.2.2 Depth Stereo Camera

Depth Estimation using a single RGB camera is not feasible. Stereo Vision is a technique used to estimate the depth of an image by using more than one camera. This is achieved by using epipolar geometry. There are two types of epipolar geometry for stereo vision: the general case and parallel case. The latter being is a specific case of the first. The information is then calculated by triangulation (that is, the depth is calculated given the disparity of the image). A thorough explanation of how the model works and how it is capable of obtaining depth data is clarified in Loop e Zhang (1999).

2.3 Convolutional Neural Networks

2.3.1 Neural Networks

The concept of neural networks dates back to the 50s. The idea was to use the same mathematical model to what was believed to be the behavior of human neurons. The idea, of course, was performing complex computations for the time. Concepts such as the Perceptron (ROSENBLATT, 1957) and Adaline (WIDROW, 1960) were developed in this time-frame.

A resurgence of neural networks happened after a few decades, where the idea of backpropagation was developed (RUMELHART; HINTON; WILLIAMS, 1985). This allowed a huge improvement in the performance of neural networks, allowing for even recognition of handwritten digits (LECUN et al., 1989). But the computational power did not allow for more complex tasks, and that became a handicap for further developments.

The current age of neural network's development starts at the beginning of the 2010s, with works such as Krizhevsky, Sutskever e Hinton (2012) and Simonyan e Zisserman (2014). It starts attached to the development of convolution neural networks and image classification, showing much better performance than other algorithms at the time. Since then, several types of new deep learning algorithms in image recognition, reinforcement learning, unsupervised classification, data generation and many other areas have been published and have changed on how we work with data.

2.3.2 Neural Networks Key Concepts

In this section, we enumerate the main steps that all neural networks have in common.

Forward Step

The forward step consists of computing the output given an input, doing so according to internal weights and biases. After the network is trained, it is expected that the output of the forward step is the correct answer for the problem the network

was trained for (At this point, the forward step will be called inference). It is critical to choose a correct activation function for this, since it will add non-linearity to the system (GOODFELLOW; BENGIO; COURVILLE, 2016).

Error Estimation and Loss Function

Given the output that the network gives after the Forward Step, it is necessary to compute its error compared to the Ground-Truth or some other kind of criteria, and compute the error between from what was expected to what the network has answered. Common Loss functions to compute the error (or loss) of neural networks are L1 distance, L2 distance and Softmax.

Error Propagation

The error propagation step happens after the error estimation. Given the computed Loss function, we propagate the error through the network. The backpropagation (RUMELHART; HINTON; WILLIAMS, 1985) algorithm is widely used, but there are several different ways to propagate errors. In Recurrent Neural Networks (HOCHREITER; SCHMIDHUBER, 1997), a recursive strategy is implemented, and in very large neural networks (HE et al., 2016), an error propagation strategy is necessary to avoid gradient vanishing.

Optimization Algorithm

To update the weights and biases of the neural network, given the computed error for a specific neuron, an optimization function is employed. This function dictates how the computed error will be used to update the parameter's values. Common choices can be the Stochastic Gradient Descent, Stochastic Gradient Descent with momentum, Adam (KINGMA; BA, 2014), and many others.

2.3.3 Specializations of the Convolutional Neural Networks

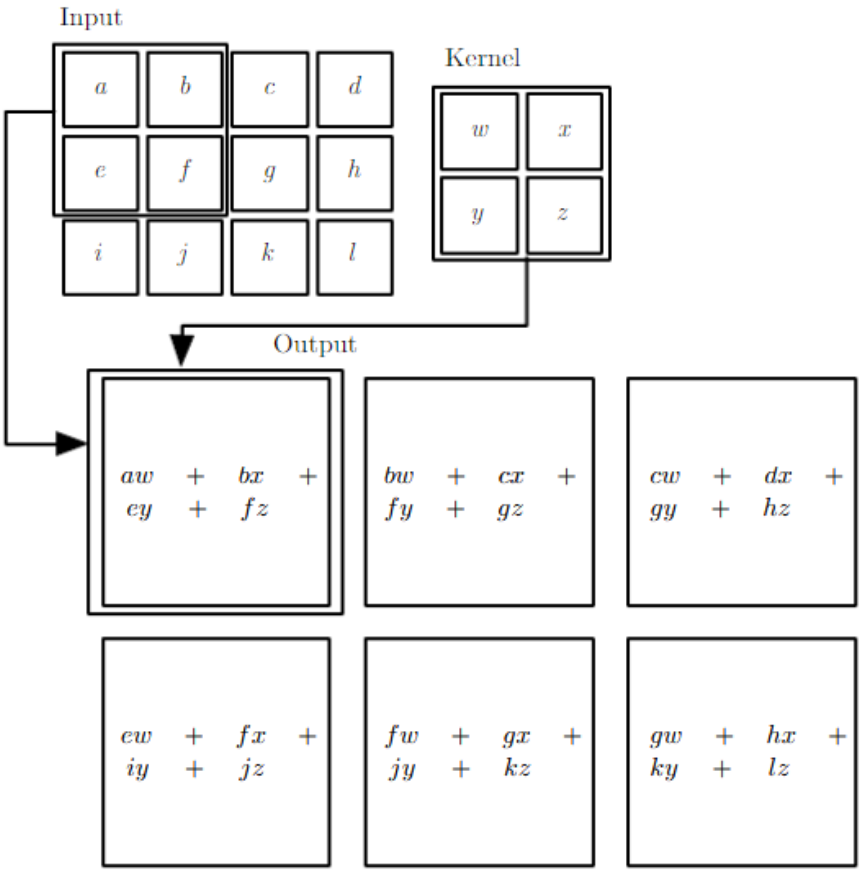
Convolutional Neural Networks are a special variation of neural networks, and it is the most common type of network used to work with image data. This is no coincidence, since the convolution operation has a few advantages for processing images.

The convolution in terms of deep learning refers to an operation of scalar product between the input values and the weight matrix (also called a kernel) of the given layer. This means the evaluation of the inputs is heavily biased to local patterns. Typically, the weight matrix is much smaller than the input tensor, in such a manner that the kernel "runs" the input, so it can cover all of its dimensions. This property that the kernel runs the entire input without changing any parameters is called *parameter sharing*. Figure 3 demonstrates how a convolutional operation works.

ReLU activation function

A usual choice for activation function, specially for images, is the ReLU activation function (NAIR; HINTON, 2010) and its variations (XU et al., 2015). This class of activation functions require little computation and in practice present good results with complex data.

Figure 3 – Convolution Operation



Source: Goodfellow, Bengio e Courville (2016)

2.4 Biases in Deep Learning

Traditionally, it was believed that convolutional neural networks learn about geometry of the objects, starting with the first layers learning with simple geometry such as lines and curves, while the latter layers would learn and recognize complex shapes such as tires, eyes and faces. However, more recent studies have indicated that there is a bias towards textures over shapes. Geirhos et al. (2018) used a stylized dataset to analyze how CNNs trained on the ImageNet (DENG et al., 2009) are biased towards texture information. They generated the dataset using style transfer techniques to ImageNet images, and passed these images into networks trained on ImageNet to perform classification tasks. This experiment shows how a CNN can be biased towards texture.

The desire and, perhaps more accurately, the need to understand how a network converges and what is happening in its weights and biases have sparkled a series of areas that try to better understand deep leaning. One such area is Explainable Artificial Intelligence (DOŠILOVIĆ; BRČIĆ; HLUPIĆ, 2018), that confronts the black-box model that most deep learning is based on, and tries to develop a deep learning model where humans are able to explain how the algorithm has arrived to its conclusion. Still, most of the deep learning methods are still black-box, and thus only ensuring the quality of the dataset and by varying the hyperparameters we can attempt to obtain better results.

2.5 Task Definition

2.5.1 Task Modelling

There are several questions that must be answered to limit the scope of the task of grasping: Are we employing an analytic or data-driven (Deep Learning) approach? Are we dealing only with rigid bodies or deformable bodies as well? Are we dealing with only a single object, or is the object in a cluttered environment? How is our gripper (is it a two-finger parallel gripper or a suction gripper? Or something else?). These questions must be answered to define the scope of the project. We use as base the work of Du, Wang e Lian (2019) and Morrison (2021) to define these terminologies and to define how a grasping can be decomposed in a series of minor tasks.

2.5.2 Model-free and Model-based approaches

An important distinction between models is their dependency in relation to the object's models. Model-free approaches try to generalize the grasp proposal to any model, be it known or not. Model-based approaches require knowledge about the object's shape, and use this data to infer the pose or grasp information.

2.5.3 Object Grasping Task

Following the division of described in Du, Wang e Lian (2019), we can enumerate a step by step process for the task of robotic grasping:

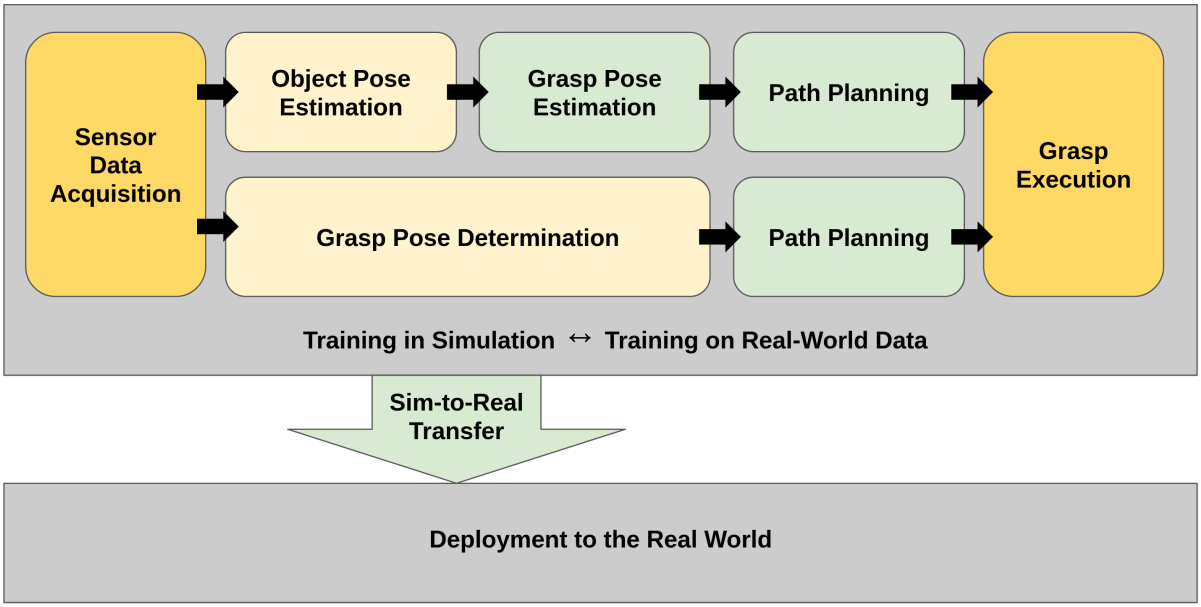
1. Object Detection
2. Object Distance and Pose Calculation
3. Calculate Best Grasp
4. Calculate Inverse Kinematics to Perform grasp
5. Attempt Grasping
6. Finish Grasp and Move Object to Other Position

How different projects approach these steps can vary greatly, e.g.: skipping steps, performing two steps at the same time, or implementing feedbacks on pipeline. Overall, we can split the work in this area into two groups: The first group attempts to calculate the

object's pose before trying to calculate the best grasp. The second group tries to calculate the grasp, without explicitly trying to calculate the object's pose first. After this step, depending on whether artificial data was used to train the neural network, there might an additional step so the network can be deployed in real applications. The figure 4 shows the pipeline for both these strategies. Here we mainly focus on the second group. However, we also bring information and relevant research about the first group, specially when it brings important discussion topics to be addressed.

In the next sections, we briefly talk about a few works that use these strategies to perform robot grasping, citing relevant information about the different characteristics between them.

Figure 4 – Basic Grasp Pipeline



Source: Based on Kleeberger et al. (2020)

2.6 Object Detection, Pose detection, Grasping Detection and Reinforcement Learning Strategies

In this section, we give a brief overview of object detection, pose detection and grasp detection techniques.

2.6.1 Object Detection

Classification and Object Detection problems are among the first problems that CNNs were employed to solve. Classification is a task that, given the image, the network will try to classify it into a given category. Object Detection not only classifies the object, but it also identifies where in the image the object is, usually the answer of the network is a bounding box. It is very similar to image segmentation, but in this case, the answer of

the network is the group of pixels the object belongs to. A throughout review of these algorithms was done by Zhao et al. (2019).

2.6.2 Pose Detection

For this section, we use as a basis the works of Sahin et al. (2020) and Kleeberger et al. (2020).

In pose detection, not only we try to identify which class an object belongs to, but also position and orientation of the object not only in image coordinates, but in world coordinates as well. Usually, only RGB data from the camera is used.

We will briefly explain the most common strategies and how they attempt to predict the pose of an object. In **Template Matching**, from a given 3D model, a series of templates in different poses are generated. These templates consist of robust feature descriptors, such as available shape, geometry and appearance. The information received from the sensor is then compared to the known templates to estimate the pose. Hinterstoisser et al. (2012) is an example of this class of methods.

Point-Pair feature matching, on the other hand, make a global representation of the object by extracting point-pair features (PPF) and storing they in a hash table. After this, it extracts the point-pair features of the scene and compares to what it has stored, generating a series of potential matches for pose candidates. Hinterstoisser et al. (2016) is a good example of this type of strategy.

Regression methods concentrate most of the work based on deep learning. They usually attempt to classify the object and it's estimated 6D pose. They can be further divided into two groups: The first group's objective is to estimate the 6D pose directly, while the second group first attempt to extract features and then finds the correspondence from image coordinates to world coordinates, usually with PnP algorithms. Xiang et al. (2017) and Do et al. (2018) belong to the first group, while Tremblay et al. (2018b) and Hu et al. (2019) belong to the second.

In the cases where the objective of the algorithm is to find a series of feature points, it is still necessary to find the translation between image coordinates and world coordinates. This problem is known as the Perspective-n-Point (PnP). Popular algorithms that solve and are employed to solve object's coordinates are Li, Xu e Xie (2012) and Lepetit, Moreno-Noguer e Fua (2009).

2.6.3 Grasping Detection

For this section, we use mostly the work of Kleeberger et al. (2020) and Newbury et al. (2022).

Grasp detection attempts to determine the best grasp for an object directly from the input data, without an explicit step to find an object's pose estimation. Unlike the methods cited previously, the most important input data here is a depth image (usually obtained from a stereo camera).

This strategy can be further subdivided in two classes of techniques: Discriminative Approaches and Generative Approaches.

In **Discriminative Approaches**, the neural network typically does not create a proposal for a grasp. Instead, a series of grasps are produced by an algorithm, and the neural network will rank which grasps it believes that will have the highest probability of a successful score. The work of Mahler et al. (2017) and Mahler et al. (2018) can be classified as a discriminative method.

In **Generative Approaches**, on the other hand, attempt to generate a grasp proposal directly. Usually, this type of method can be seen somewhat as an extension from the classification and object detection methods, discussed in 2.6.1. This class of methods usually attempt to output the center of the grasp, the width of the gripper, and it's opening size (if looked through the image coordinate system, the output will be similar to a rectangle). Projects that can be labeled as generative are Morrison, Corke e Leitner (2020b).

2.6.4 Reinforcement Learning

Reinforcement Learning strategies are also largely employed to train neural networks for grasping. Here, the neural network will be given a score for each grasp attempt, and the network will try to maximize the score for that particular task. Choosing the appropriate scores for desired and undesired behaviors is essential for this type of strategy. Kalashnikov et al. (2018) is a great example of this class of algorithms.

2.7 Further strategies to robotic grasping

2.7.1 Closed-Loop Grasping and Active Perception

A variation of the pipeline presented in 2.5.1 is the implementation of feedback loops involving the camera, so the sensor can be used more efficiently. The application of closed-loop solutions is important, specially in unstable or unstructured environments, where changes in the environment can happen between the step from data acquisition to grasping (e.g. an object rolling to another position). The concept of using a continuous stream of data to adjust the grasping as necessary is called closed-loop grasping (MORRISON, 2021). A related concept, but not quite the same thing, is what is called active perception. In Active perception, the robot actively chooses the best viewpoint to observe the object, the one where the algorithm will provide the best grasp suggestions to a given object.

(MORRISON, 2021)

2.7.2 Pre-Grasp Strategies

Certain object placement configurations can make a robotic system job difficult. The object may be in a cluttered environment, and it may be difficult to access it on a single movement, or maybe if the object was in another pose it would be easier to be grasped. A robotic system can be taught to realize certain actions that change an object's pose before a grasp, so the chance's of success can improve. Usually, the approaches are trained in a reinforcement learning fashion, such as the work of Zeng et al. (2018).

2.7.3 Artificial Data Generation

Neural Networks that belong to Supervised Learning class are data-hungry algorithms. Since they are good interpolators of data, the more diverse and representative the dataset is, the more reliable the answer of the given network is. However, gathering such a dataset is no easy job. It is still commonplace the manual gathering and labeling of data. The Lin et al. (2014) and Deng et al. (2009) datasets, two of the most relevant datasets in object detection and classification problems, rely heavily on manual labor to correctly identify the classes and where they are presented in the image.

For Pose Estimation and Grasp Annotation Datasets, an additional challenge is presented. How will a human correct label the vertices coordinates of an object in a given image? Or will a human correctly discern which grasp in image coordinate systems will result in a successful grasp? Since it is complex for human labelling to create robust data for training for these tasks, other strategies were developed.

Artificially generating the data has a number of advantages. Given that all coordinates are known inside a rendering scene, you have the exact location of an object and all of its vertices. And with pipeline rendering, it is possible to produce a large quantity of custom-made data to a specific application relatively fast. However, this strategy also has its downsides. Major among them is the reality gap. That is, the gap in the quality of data that is inherited from the method it was generated. There are a few ways in the literature that people have tried to deal with this problem, the most common methods are Photorealistic data and Domain Randomization.

In the **Photorealistic Data** strategy, the objective is to reproduce data appearance as faithful to the real world as possible. Illumination and light reflectance in materials are specially important here. It is usually employed in scenes simulation, both indoors (MCCORMAC et al., 2016) and outdoors (ROS et al., 2016; GAIDON et al., 2016; TSIRIKOGLOU et al., 2017).

In **Domain Randomization**, the objective can be seen as the opposite from Photorealistic Data. Here, we aim to change the scene in various unrealistic ways, so that

when the network is presented to real-world data, it will only be seen as another variation from this trend (TOBIN et al., 2017). This approach, however, may prove to be insufficient on its own, requiring fine-tuning from real-world data (TREMBLAY et al., 2018a).

Both strategies can be used together in a **blended strategy** to create a more robust dataset, as can be seen with the works of Tremblay et al. (2018b) and Loing, Marlet e Aubry (2018).

2.7.3.1 Domain Adaptation

Just as Domain Randomization, in **Domain Adaptation** we aim to variate the parameters so that the reality will only appear as another variation to the neural network. We usually have a source domain (simulation) and aim to transfer to it characteristics from the target domain (real-world). It is commonplace to use GANs (BOUSMALIS et al., 2018) as to transfer unlabeled data from the real-world, reducing the fine-tuning necessary with real-world data.

2.7.4 Simulation

There are works that have been trained solely on a laboratory setup. Kalashnikov et al. (2018) had trained his network for a total of 800 hours across 7 robots to complete his research. Not only it is time-consuming, but it also relies on expensive equipment and additional manual labor from the research team. An alternative to this is computer simulation using a suitable simulation engine for robotics (KÖRBER et al., 2021). This can be used in a number of ways, to directly train scenes that use reinforcement learning, but also in addition to generate data from supervised learning techniques. (MAHLER et al., 2017; DEPIERRE; DELLANDRÉA; CHEN, 2018).

2.8 Neural Networks for Robotic Grasping

2.8.1 GG-CNN Network

Morrison, Corke e Leitner (2020b) presented a network that can be classified as a generative approach, has low computation demands and can be used in a closed-loop fashion (see 2.7.1). In this section, we explain the modelling behind the algorithm.

Grasp Modelling

A grasp is defined in 3D space by Equation 2.2.

$$\mathbf{g} = (q, \mathbf{p}, \phi, w) \tag{2.2}$$

Where q is a quality measure representing the chances of grasp success, $\mathbf{p}$ is the gripper's center position $\mathbf{p} = (x, y, z)$, Φ is the gripper's rotation around the z-axis and w is the gripper's width

The same grasp is defined in image space by Equation 2.3, being differentiated by the tilde operator.

$$\tilde{\mathbf{g}} = (q, \tilde{\mathbf{s}}, \tilde{\phi}, \tilde{w}) \tag{2.3}$$

Where $\tilde{\mathbf{s}}$ is the gripper's center position in image coordinates $\tilde{\mathbf{s}} = (u, v)$, $\tilde{\Phi}$ is the gripper's rotation in the camera's reference frame and w is the gripper's width in image space.

The conversion between a grasp in Image Space ($\tilde{\mathbf{g}}$) and 3D space (g) is given by the Equation 2.4.

$$\mathbf{g} = t_{RC}(t_{CI}(\tilde{\mathbf{g}})) \tag{2.4}$$

t_{CI} is the conversion from Image Space to the Cartesian Space and t_{RC} is the conversion between the world frame and camera frame, following the Pinhole's camera model (described in subsection 2.2.1) and robot-camera calibration

Grasp Map and Finding the Best Grasp

The Set of possible grasps in the Image Space is denoted as *Grasp Map* and is given by Equation 2.5.

$$\tilde{\mathbf{G}} = (\tilde{\mathbf{Q}}, \tilde{\Phi}, \tilde{W}) \tag{2.5}$$

We wish to find the function M, so that, given the Input Image $\mathbf{I}$ (RGB Image, Depth, or both), we obtain $M(I) = \tilde{\mathbf{G}}$ and find the optimal grasp $\tilde{\mathbf{g}}^* = max_{\tilde{\mathbf{Q}}}\tilde{\mathbf{G}}$

Finding M

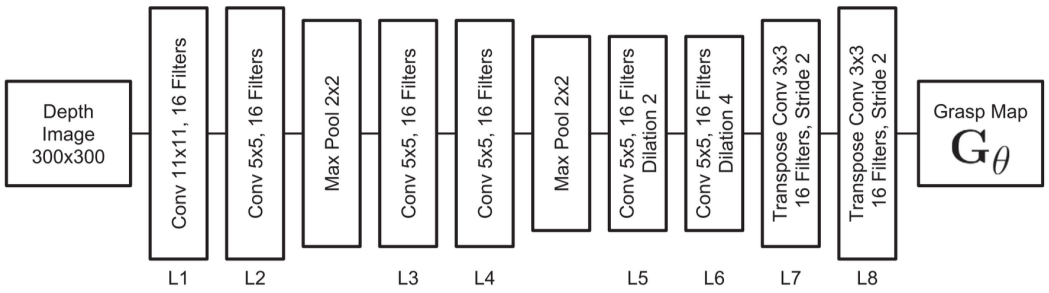
We use a solution M_Θ that converges to the desired solution M. M_Θ is the trained neural network with weights Θ. M_Θ takes the I and outputs the approximate grasp map $\tilde{\mathbf{G}}_\Theta$.The network is parameterized by its weights Θ and computes the function $M_\Theta(I) = (\tilde{\mathbf{Q}}, \tilde{\Phi}, \tilde{W}) \approx M(I)$.

GG-CNN Architecture

Morrison, Corke e Leitner (2020b) proposed a fully convolutional architecture as a solution to the problem. The detection pipelines consists of four stages: image processing, generation of pixel-wise grasp quality through the network, filtering and computation of the best grasp pose.

Figure 5 shows the architecture of the network. The image is cropped to the size of 300x300 pixels. The final network contains 66,000 parameters, has an average inference time of 3 ms.

Figure 5 – GG-CNN Architecture



Source: Morrison, Corke e Leitner (2020b)

2.9 Benchmarking for Pose Detection

From this point, we start to evaluate datasets, precision metrics and how these projects that aim to generate data and measure the quality of the algorithms they are trained on. All these datasets in this section evaluate only the Pose Detection step that is shown in 4, and do not take into consideration how the grasp proposal that the algorithm must employ to a successful grasp.

2.9.1 LINEMOD

A very popular dataset is the one created by Hinterstoisser Stefan Holzer (2011) and Hinterstoisser et al. (2012). They proposed a dataset with objects without texture in cluttered environments, with illumination under control and minimizing occlusion.

Several works, such as Brachmann et al. (2016), Rad e Lepetit (2017), Kehl et al. (2017) and Tekin, Sinha e Fua (2018) use this dataset.

The precision metric m is a comparison between the real rotation $\bar{R}$ and translation $\bar{T}$ with the estimated rotation $\hat{R}$ and translation $\hat{T}$, for a series of points in the model. This is shown in equation 2.6, where M represents the set points belonging to the model

and x represents a specific point belonging to M.

$$m = avg_{x \in M} \parallel (\bar{\boldsymbol{R}}x + \bar{\boldsymbol{T}}) - (\hat{\boldsymbol{R}}x + \hat{\boldsymbol{T}}) \parallel \tag{2.6}$$

It is considered that the model has been correctly identified if $k_m d \geq m$, where k_m is an arbitrary coefficient and d is the model's diameter. If there is symmetry involved, the precision is computed as shown in equation 2.7.

$$m = avg_{x_1 \in M} \; min_{x_2 \in M} \parallel (\bar{\boldsymbol{R}}x_1 + \bar{\boldsymbol{T}}) - (\hat{\boldsymbol{R}}x_2 + \hat{\boldsymbol{T}}) \parallel \tag{2.7}$$

2.9.2 T-LESS

Hodan et al. (2017) is a dataset that was developed thinking in industrial applications. It contains little texture information, and the objects tend to have many symmetries. It uses the same metric as shown in section 2.9.1, with the coefficient determined to be 0.1.

2.9.3 YCB Object and Model Set

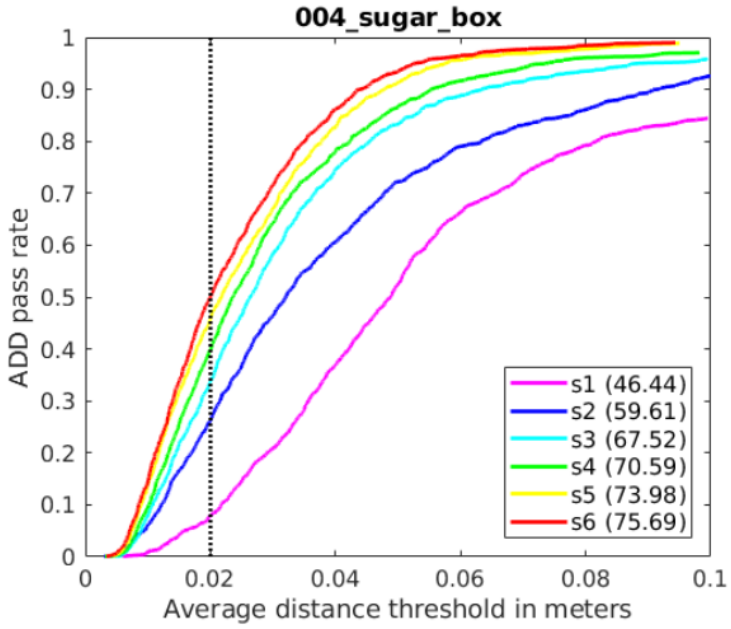
Calli et al. (2015) proposed a set of common household items with a large range in size, colors, textures, materials and geometric shapes. Many projects (XIANG et al., 2017; TREMBLAY et al., 2018b) use this dataset, and it does not propose a precision metric. The quality metric used in these works is based on the metric presented in section 2.9.1. However, instead of defining an arbitrary value for a threshold, several threshold values are considered, creating a graph of threshold vs recall. The quality metric taken into consideration is the area under the curve. The figure 6 shows an example of curves for different objects.

2.9.4 BOP Challenge

Hodaň, Matas e Obdržálek (2016) proposed a unified framework to test pose estimation algorithms. They gather 15 of the most relevant datasets and unify the format they are provided on, while proposing a different precision metric, called Visual Surface Discrepancy (VSD). This metric aims to deal better with occlusion, be it self-occlusion or by other objects or environment.

Given an estimated pose $\hat{P}$ and a real pose $\bar{P}$ from the object M, two distance maps are calculated, $\hat{S}$ e $\bar{S}$. For each pixel ρ, the map computes the euclidean distance between the camera's center and the point x_ρ (both represented in world coordinates), where x_ρ is the point in the world coordinate system that is projected to the point ρ in the

Figure 6 – Metric used by Xiang et al. (2017) e Tremblay et al. (2018b)



Source: Tremblay et al. (2018b)

image. $\hat{V}$ and $\bar{V}$ are the poses' visibility masks, in other words, they are the pixels that are visible in the image. Given a threshold τ, the error is computed according to equation 2.8.

$$e_{VSD}(\bar{S}, \hat{S}, \bar{V}, \hat{V}, \tau) = avg_{\rho \in \bar{V} \cup \hat{V}} \begin{cases} 0 \; se \; \rho \in \bar{V} \cap \hat{V} \wedge |\bar{S}(\rho) - \hat{S}(\rho)| < \tau \\ 1 \; otherwise \end{cases} \tag{2.8}$$

2.10 Benchmarking for Grasp Detection

2.10.1 Cornell Dataset

The Cornell dataset (JIANG; MOSESON; SAXENA, 2011; LENZ; LEE; SAXENA, 2015) was one of the first datasets consisting of real objects and grasps annotation data. It contains the RGB-D information from the images, with additional annotation of possible successful grips. The annotations are in the image coordinate system, it consists of the center of the grasp, the orientation of the grasp, and the size and opening of the gripper. This data can also be seen as a rectangle in the image.

In this setup, a successful grasp is considered when two conditions are met: first, the IoU (Intersection over Union) between the ground-truth annotation and the proposed grasp must be over 25%. The second condition is that the proposed grasp must be with 30° of the ground-truth label.

2.10.2 Jacquard Dataset

The Jacquard Dataset (DEPIERRE; DELLANDRÉA; CHEN, 2018) was created to tackle problems that were present in the Cornell dataset (described in section 2.10.1). It improved not only the size of the dataset, but also the variety of the objects. Since manually creating the dataset would be labor-intensive, they decided to create the data artificially. To accomplish this, the authors decided to create the dataset using both artificial image rendering and simulation to rank the best grasps. Rendering is done using Blender and the Cycles renderer, while the simulation is done using PyBullet (COUMANS; BAI, 2016–2022).

The data generated for this dataset were based on the cad models originally present in the ShapeNetSem Dataset (CHANG et al., 2015). The cad sizes vary widely in the original dataset, so they were all resized so that the longest size of the objects is between 8-90cm. Also, weight for the objects were inferred by their size (80 g for an 8 cm object and 900 g for a 90 cm one).

The pipeline for Dataset creation was organized as it follows:

1. The camera is positioned in a fixed altitude from a top-down perspective over a white floor
2. The Model is loaded into a scene and the simulation runs until the object is stable
3. Several random grasps proposals are calculated for the object
4. Texture is loaded, and the image is rendered (Gaussian noise is applied to the image)
5. Simulation is performed in the random grasps using a grasp of size 2 cm and max opening of 10 cm, the grasp is considered successful if the object is completely lifted, moved away and dropped at a specific point
6. The simulation is then repeated with jaw grippers with sizes: 1, 3, 4 and 6 cm.
7. After all successful grasps are saved, a clean-up is performed so that only one grasp is saved in case that grasps are too similar one to another

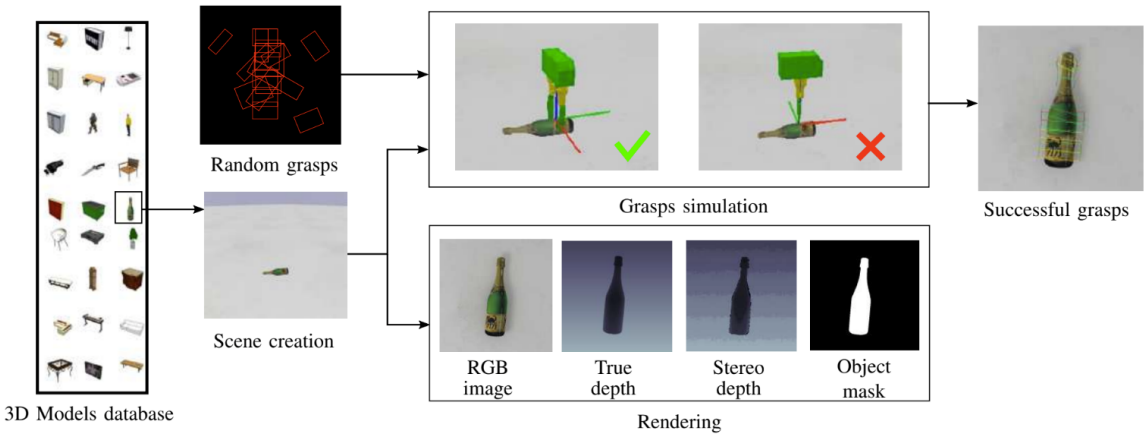
The pipeline is also described in figure 7.

The Jacquard Dataset has several strengths compared to the Cornell, it has a much wider variety of objects. Not only it generates RGB-D data, but also creates the object mask.

However, the dataset also has its downsides. First, the models were loaded from the ShapeNetSem dataset. Which means that the acquisition of physical objects that are equivalent to the models is not an easy task. Another weakness of this dataset is that the pipeline is not available for other researchers, and evaluation of networks is performed by a closed server.

The Dataset follows the following annotation pattern for each grasp proposal:

Figure 7 – Pipeline proposed by the Jacquard Dataset



Source: Depierre, Dellandréa e Chen (2018)

- center of grasp : (x, y) coordinates in the image
- orientation of the grasp: theta angle in degrees
- jaw opening size
- jaw size

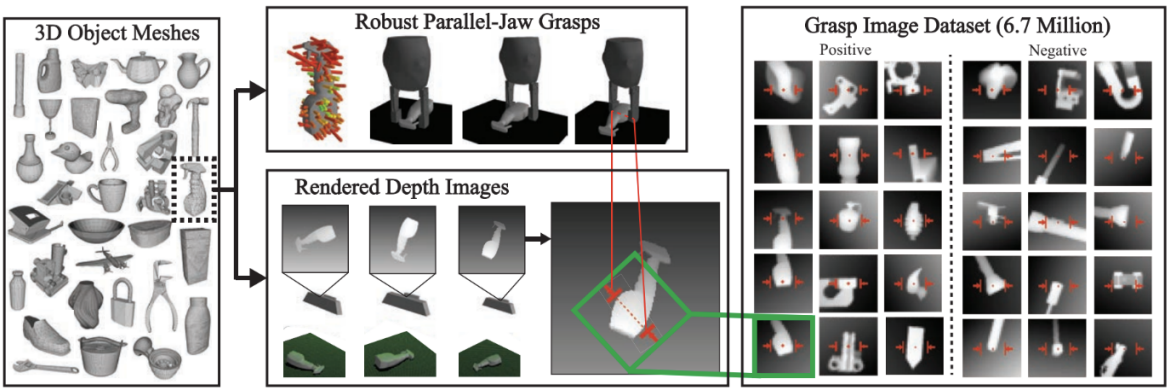
2.10.3 Dex-Net Family Dataset

The Dex-Net (MAHLER et al., 2017) dataset was created using a different approach from the Jacquard Dataset (see 2.10.2). By taking a series of objects (MAHLER et al., 2016), and simulating them in randomized poses on a table. Each object is labeled with up to 100 parallel-jaw grasps, that was proposed using a robust grasping policy and labeled with force closure and the expected epsilon quality metric (POKORNY; KRAGIC, 2013). The provided data consists of a depth image for each stable pose, a random noise is added during the image during the network training. Figure 8 demonstrates how the pipeline is structured.

A relevant addition to this dataset was the inclusion of a group of adversarial 3D-printed objects. This is particularly relevant because many datasets do not include objects that can be easily obtained for physical experiments. By adding the models of objects that can be easily 3D printed, the reproducibility of a work greatly increases.

Subsequent projects focused on creating grasp annotations for different gripper formats, such as suction grippers (MAHLER et al., 2018) and ambidextrous robots (MAHLER et al., 2019).

Figure 8 – Pipeline proposed by the Dex-Net

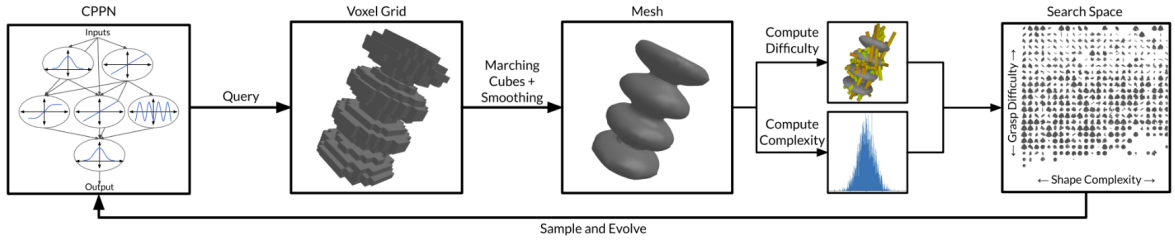


Source: Mahler et al. (2017)

2.10.4 EGAD!

Morrison, Corke e Leitner (2020a) saw a few problems with previous works. The first problem was that most of the previous datasets were not easily reproducible in real life scenarios. Many of these datasets were composed of common household objects. And obtaining the real objects for experiments is not trivial in this case. Following the work of Mahler et al. (2017), Morrison, Corke e Leitner (2020a) decided to expand the concept of using 3D printing to create easily reproducible physical datasets. In his work, an evolutionary algorithm is used to balance between two characteristics of the objects. The first characteristic is shape complexity, that is measured by morphological complexity. The second characteristic is an object's grasp difficulty, defined by the difficulty that is for a parallel gripper to grasp the object, this is measured by using an analytical grasp planner and computing a grasp quality metric. The grasp difficulty feature is then obtained by taking the 75th percentile grasp quality of all sampled grasps. Figure 9 demonstrates how the pipeline is structured.

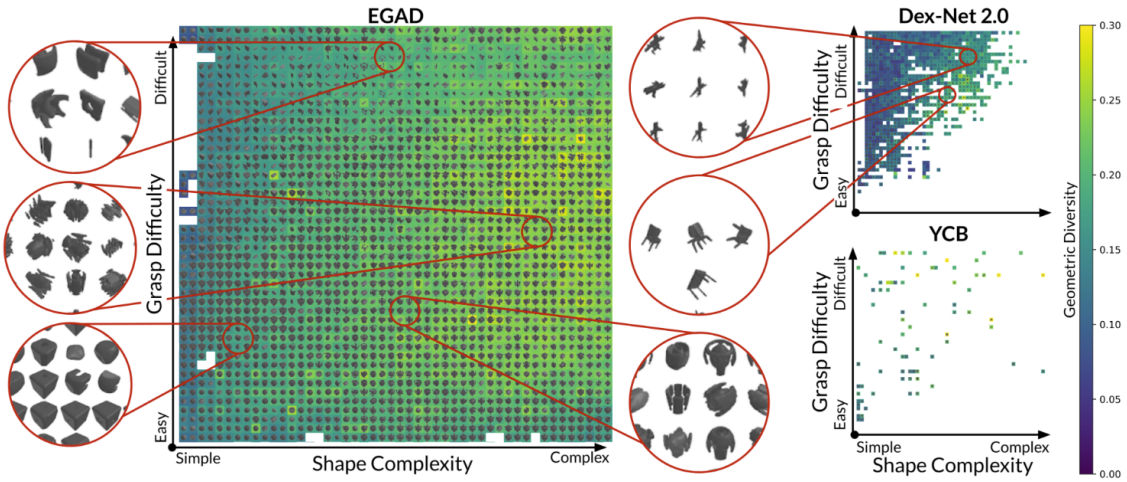
Figure 9 – Pipeline proposed by the EGAD! Dataset



Source: Morrison, Corke e Leitner (2020a)

Since the evolutionary algorithm was configured to generate several object proposals that cover different measures of shape complexity and grasp difficulty, the dataset presents a much more balanced representation in this space, as can be seen in the figure 11.

Figure 10 – EGAD! comparison to other datasets



Source: Morrison, Corke e Leitner (2020a)

This approach has a series of advantages compared to previous datasets. Since the datasets can be easily printed, real-world tests can easily be performed, and since the objects were not randomly selected, but created using metrics of shape complexity and grasp difficulty in mind, it is much more representative of the types of objects it needs to learn how to grasp. Furthermore, the authors selected a group of 49 objects that are representative of the space so that it can be used for evaluation.

Another relevant contribution of the work is the analysis comparing the ratio between the gripper width and the object's size. Figure 11 shows the non-linear relation between relative size and grasp quality. For fair comparison, in this work they set that the relation between object's size and gripper width would be no more than 80%.

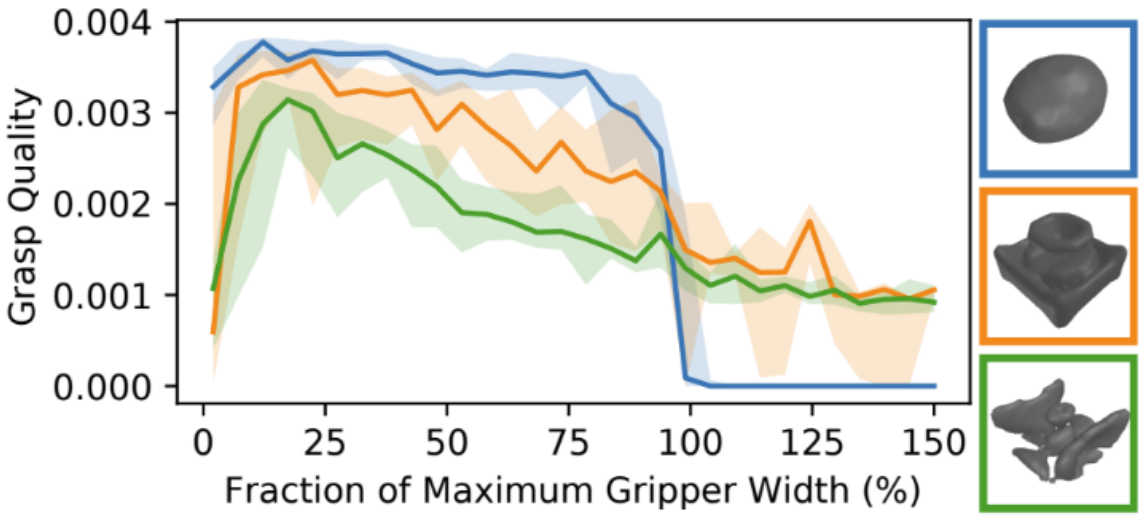
2.10.5 Precision Metrics

Datasets in the Direct Grasp Detection Family tend to measure grasp quality in one of two ways. A first approach is to use a grasp quality metric that is well established in the literature, and usually has an origin together with analytic methods. A common metric is the robust Ferrary-Canny metric (FERRARI; CANNY, 1992), used by Mahler et al. (2017). The other approach is to, either in simulation or in real-world experiments, attempt a certain grasping task. For example, the Jacquard (see 2.10.2), attempts to grab the object, move it to a specific location and drop it. The grasping is considered successful only if the robot is capable to perform this entire process.

2.10.6 Gripper Shape and Size Effect on Results

Another important aspect for robotic grasping is the gripper's type (parallel gripper, suction gripper, etc.) and size's in relation to the object that is about to be grasped.

Figure 11 – Graph relating quality metric to an object's relative size to the gripper's width



Source: Morrison, Corke e Leitner (2020a)

Datasets cannot easily translate a gripper's characteristics, so the data is highly correlated to a gripper format and size.

Most of the datasets focus on parallel grippers (LENZ; LEE; SAXENA, 2015; DEPIERRE; DELLANDRÉA; CHEN, 2018; MAHLER et al., 2017), with a few datasets focusing on suction grippers (MAHLER et al., 2018) and ambidextrous robots (MAHLER et al., 2019).

For parallel grippers, the ratio of object size to gripper's width in also significant, as grasp successes tend to decrease when this ratio is over the 80% threshold (see section 2.10.4).

2.11 Datasets Comparison

In this section, we compare and evaluate the characteristics of each dataset, the result can be seen in Table 1.

2.12 Frameworks for Data Generation

2.12.1 BlenderProc

After a few preliminary works that attempted to generate artificial data to train networks, more robust tools were developed to generate data and labels for training artificial intelligence algorithms.

Denninger et al. (2019) created a framework that uses the blender API to generate high-quality images suitable for training deep neural networks. Since blender is an open-

Table 1 – Datasets Comparison

Dataset	Application	Data Pattern	Real or Synthetic	Precision Metric
LINEMOD (HINTERSTOISSER STEFAN HOLZER, 2011)	Pose Detection	RGB Data	Real Data	Average Difference in Vectors
T-LESS (HODAN et al., 2017)	Pose Detection	RGB Data	Real Data	Average Difference in Vectors
YCB (CALLI et al., 2015)	Pose Detection	RGB Data	Real Data	Area under the curve for Average Difference in Vectors
BOP Challenge (HODAŇ; MATAS; OBDRŽÁLEK, 2016)	Pose Detection	RGB Data	Real and Synthetic Data	Visual Surface Discrepancy(VSD)
Cornell (LENZ; LEE; SAXENA, 2015)	Direct Grasping	RGB Data	Real	Successful Grasp Rate
Jacquard (DEPIERRE; DELLANDRÉA; CHEN, 2018)	Direct Grasping	RGB Data and Depth Data	Synthetic Data	Successful Grasp Rate
Dexnet (MAHLER et al., 2017)	Direct Grasping	Depth Data	Synthetic	Successful Grasp Rate
EGAD (MORRISON; CORKE; LEITNER, 2020a)	Direct Grasping	Depth Data	Synthetic	Successful Grasp Rate

source program, it is a great choice for this type of application. The framework uses the Cycles renderer, that is suitable for realistic renderings.

The framework is set to be able to render a series of relevant types of image data that can be used for a vast range of simulated scenarios, such as enabling an object's physics positioning, material randomization, semantic segmentation, stereo vision simulation, apply coco annotations, among other features. It is also highly integrated with the BOP Challenge (HODAŇ; MATAS; OBDRŽÁLEK, 2016), providing the interface to the challenge's metrics.

Although the framework is powerful for providing realistic artificial image data in several formats, a drawback is that it is not integrated with a simulation engine, limiting its usage when believable physics simulations are necessary to generate data.

2.12.2 Kubric

The work of Greff et al. (2022) can be seen as an extension of the work performed by Denninger et al. (2019). It offers a framework where a common scene configuration can be provided to both a renderer and a physics simulation engine. Providing a necessary tool for developments in robotics. It realizes the difficult task of integrating robotics simulation and data generation.

While the renderer is automatically configured to generate RGB, depth and segmentation data (among others), the integration with the engine allows for a more realistic behavior of object's interaction. By default, it is built on top of Blender (for rendering) and PyBullet (for simulation), but the code is kept modular, and it is possible to swap the back engines.

3 MATERIALS

3.1 Kubric Framework

We use as a base the Kubric Framework to generate our dataset. The reason this particular framework was chosen is that it is a state-of-the-art tool that is able to create a unified scene for both rendering and simulation.

3.2 Machine Learning Libraries and Tools

Machine Learning libraries and tools have become widely available in several programming languages. In this group, two libraries are largely used: TensorFlow (TENSORFLOW, 2020) and PyTorch (PYTORCH, 2020). This work uses PyTorch since the language syntax has been more stable along the library different versions, making it more cohesive.

3.3 OpenCV

The OpenCV library (Open-Source Computer Vision Library) (OPENCV, 2020) was developed with the intention to make it easier to developer to access computer vision algorithms for many areas and applications. It is implemented in C++, but has a wrapper available in python.

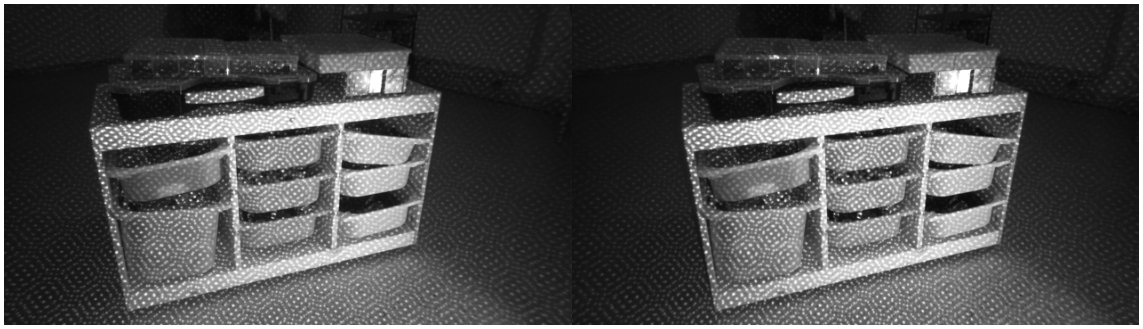
3.4 Real Sense Camera

The Real Sense Camera (KESELMAN et al., 2017) family was developed by Intel, and consists of a set of depth cameras developed for both indoor and outdoor use cases.

The Real Sense is a stereo camera, meaning that it calculates the depth data by using two rgb cameras, that identify the same interest points in the two images and allow for calculation of the third dimension. An overview of the general principle of how stereo cameras work can be seen in section 2.2.2. It is relevant to highlight that traditionally, stereo cameras have been known to not perform well with low-texture images, since the stereo algorithm cannot detect enough interest points to give trustworthy depth data. The Real Sense tries to address this by projecting several infrared light rays, not visible to the human eye, but detectable to the camera sensors, and thus "forcing" a texture in an otherwise low-texture image. This can be seen in the figure 12, and 13 shows the improvement in data acquisition. This technique is also called Active Stereo. Further specifications and descriptions about the camera can be seen in Keselman et al. (2017).

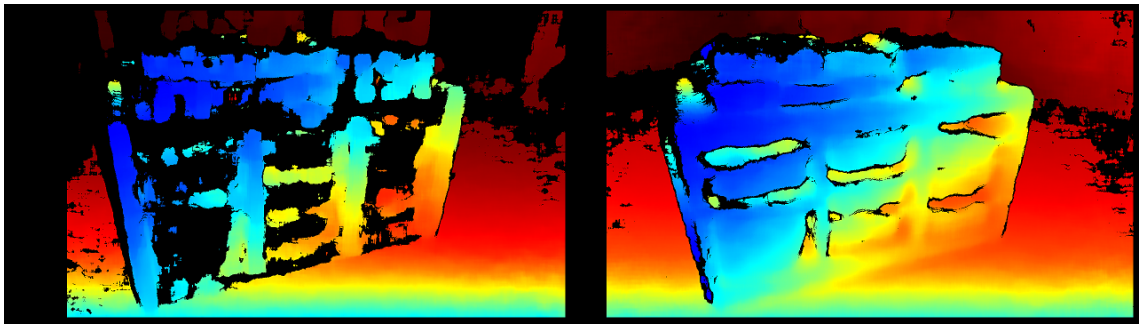
In this work, we use an Intel Real Sense D535.

Figure 12 – Projection on image - Left and right cameras



Source: Intel (2023)

Figure 13 – Results improvement using active stereo



Source: Intel (2023)

3.4.1 Real Sense SDK and API

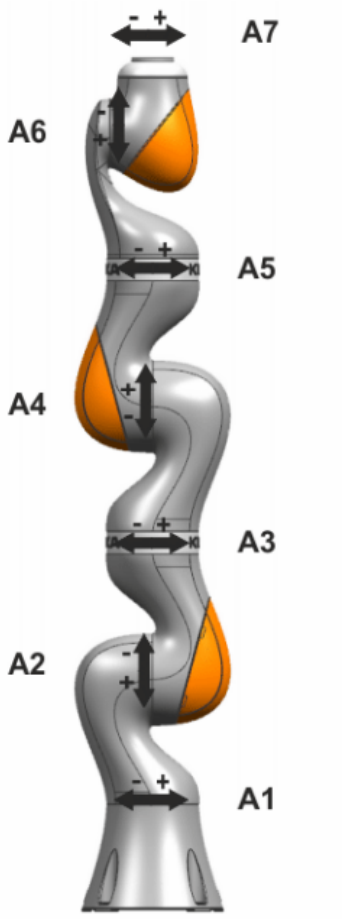
The Real Sense API offers a series of useful algorithms and tools for applications with the real sense. The API was developed in C++ but also offers wrappers to another languages, such as python.

Among the many relevant tools that the API provides are algorithms for image coordinates to world coordinates (See sections **??** and 2.2.1), calibration procedures, rgb and depth image alignment, filters, among other features.

3.5 KUKA LBR IIWA 14 R820

The KUKA LBR IIWA 14 R820 (illustrated in figure 14) is a 7-DOF redundant robotic manipulator with serial kinematic chain designed with several key features to be able to allow it to be used in proximity to humans, and even in human-machine interaction applications. It has position and force-torque feedback sensors that allow its usage from basic robotic control to critical human interaction control.

Figure 14 – KUKA LBR IIWA 14 R820 Robot



Source: KUKA (2016)

3.5.1 KUKA ROS API

Mokaram et al. (2017) has developed an API that allows controlling the robot from another computer, using ROS and TCP communication. It encapsulates most of the most common commands for the robot, such as PTP movement and Linear movement, and can easily be expanded if necessary.

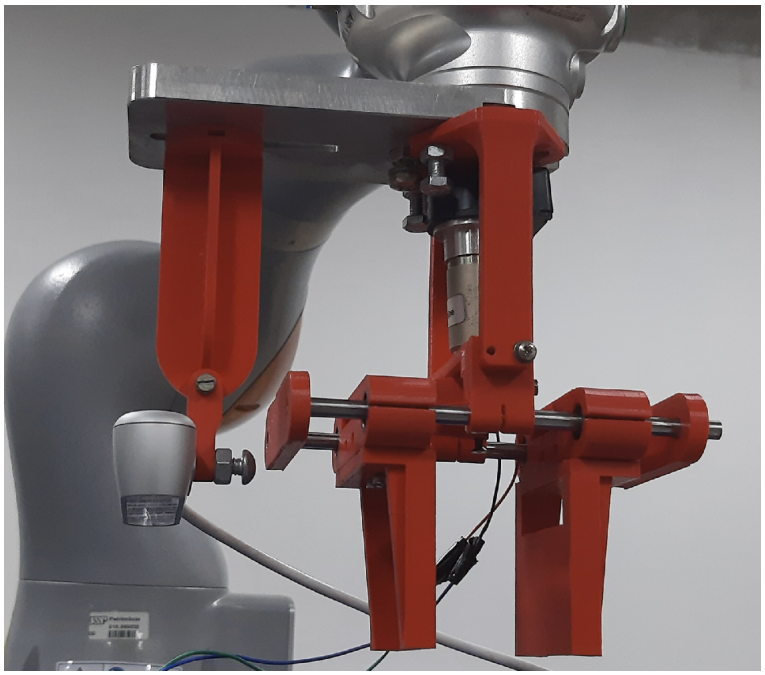
3.5.2 Security Concerns

Even if the robot has features that allow for safe use, these features must be correctly implemented to allow safe usage not only for the people but to avoid damaging the robot as well. In the first configuration, we implemented a zone that no part of the robot can go to. This was configured to avoid collision between the robot and the rest of the equipment. A second configuration was implemented so that the robot's TCP cannot exit a certain area. This was configured so that possible ill-configured movements would not result in damage to the robot or a possible collision with the operator.

3.6 Parallel Gripper

For the experimental setup, we used a parallel gripper made in-house. The gripper was mostly 3D printed. Except for the base, that is made from aluminum and the guides that are made from steel. An additional stand for the camera was also 3D printed as well. Figure 15 demonstrates the gripper assembly for real-world experiments.

Figure 15 – In-house Gripper



Source: Author

We work with a jaw opening of 7 cm.

The relative distance between the gripper's center and the camera are: 4 cm in the x axis and 11 cm in the y axis. These offsets are taken into consideration when calculating the position that the robot must move to attempt to grasp the objects.

The communication is done with a Toradex vf-50 that hosts a TCP server and awaits commands to calibrate, open or close the gripper.

3.7 Datasets

We use the objects in the EGAD! dataset (described in section 2.10.4) as baselines for experiments. First, there is a study associated with the shapes of particular objects, meaning that they represent a more diverse range of difficulties that impact not only the visual data processing but in the grasps tests as well. The second reason this dataset was chosen is that it is easily 3-Printable, allowing for reproducibility in lab tests (figure 16).

Figure 16 – 3D printed dataset

Source: Author

For lab tests, we limit ourselves to the A5, A1, E5 and B1 objects (figures 17, 18, 19 and 20).

Figure 17 – Object A5

Source: Author

Figure 18 – Object A1 different sizes

Source: Author

Figure 19 – Object E5

Source: Author

Figure 20 – Object B1

Source: Author

4 METHODS

4.1 Artificial Data Generation

Based on the work developed by Depierre, Dellandréa e Chen (2018), we propose a pipeline for generating artificial data for training neural networks, by using Greff et al. (2022) framework. We allow for personalization for output format, grasp proposal methods, end-effector types and robot models.

The original pipeline was described in detail in section 2.10.2. Our pipeline presents key differences to the original, notably, while the paper's original authors did not make the source code of the pipeline available, all the source code for this project can be found online. The main reasons the jacquard dataset was chosen as a reference are that the pipeline provides both RGB and depth data, allowing for rich visual information. Another reason is that grasps are obtained through simulation, instead of hand-labelled. As a base for the development, we use Kubric framework, since it provides the possibility to describe Blender and PyBullet environments in a unified fashion.

Since the Jacquard dataset is widely used, we follow the same output data format for an easier comparison. Just as in the original paper, we limit ourselves to a fixed-camera, top-down view intended for top-down grasps. However, a key difference is that we use Morrison, Corke e Leitner (2020a) generated objects.

4.1.1 Pipeline

In this section, we explain the overall pipeline for data generation. The pipeline has a main function, that needs the path to where the .URDF files are located in the local system, and how many instances of each object will be generated. When the entry point of the pipeline is called, the main function will call a number of subprocesses to render the data, the following steps are what each subprocess computes.

1. A Scene is generated, with a floor, a camera and walls that will limit the space the object will be allowed to be in
2. The grasp object will be loaded in a random position and pose
3. The simulation runs until the object is in a stable pose
4. The scene is rendered
5. The camera is deleted, and a new camera is loaded 10 cm to the right
6. The scene is rendered again
7. By using the images rendered in both scenes, we calculate the stereo disparity and create a stereo depth image
8. We save the visual data: RGB image, true depth and image segmentation from the

first camera's viewpoint, and the generated stereo depth

9. A grasp proposal method runs, and generates grasp proposals in image coordinates that will be used in the simulation
10. At this point, we have generated all visual data for the scene, and we can start the grasping simulations. The following steps happen for each grasp proposal.
    a) The grasp's proposal coordinates are converted to world coordinates
    b) The robot goes to the coordinates and attempts to grasp the object. A grasp is considered successful if the robot is able to grasp the object and move it to a drop point
    c) If the grasp is successful, the grasp coordinates (in image coordinates), are saved

4.1.2 Code Architecture and Components

As already mentioned, an important premise when developing the Code for the Artificial Data Generation pipeline was the ability to customize simulation, so it would be better suited for many use cases.

We encapsulate four elements of the pipeline: The Data Exporter Component, The Grasp Proposal Algorithm, The Stereo Matching Algorithm and The Robot Control Component.

We provide a default behavior for each component: The Data Exporter Component is set to export data in the same format as the Jacquard Dataset, the Grasp Proposal Algorithm is set to a Gaussian grasp proposal strategy, The Stereo Matching Algorithm uses the OpenCV's implementation of Hirschmuller (2007) and The Robot Control Component implements the Control for a Cartesian Robot.

4.1.3 Simulation Considerations

Collision Considerations

Since the original 3D objects are not designed for physics simulation, it was necessary to design each object counterpart to be able to obtain a reliable grasp simulation for generating ground-truths. These new generated objects are referred to as collision objects in this document. How this was achieved is explained in section 4.1.3.1. For simulating the grasping, a simple cartesian robot was used, and this is described in section 4.1.3.2.

Size Considerations

The Bullet engine historically has a problem with small-sized objects. Although most discussions about this topic are already quite old (TASORA, 2011; EJTTTJE, 2011), in this work we took a safe approach and scaled up the system to ensure collision stability between the elements in simulation. The collision object is scaled, so its largest dimension

is scaled to 1 m. The robot gripper has an opening width of 2 m, and the plane where the object can be in is limited to an area of 6 m x 6 m.

4.1.3.1 Correcting Meshes Geometry for Simulation

A pipeline was designed to format the original objects to have a reliable behaviour in simulation. The pipeline can be described as follows:

1. Resize object so the biggest size of the object is equal to 1 meter
2. Voxelize object
3. Calculate center of mass
4. Normalize points coordinates around center of mass
5. Calculate moment Of inertia matrix
6. Use v-hacd algorithm to simplify collision surface
7. Generate .URDF file for each object

Resizing

First, we scale the object, so the biggest dimension is equal to 1 m.

Voxelization

The next step in the pipeline is the voxelization of the mesh. This is necessary to be able to discretize the mesh volumetrically, and calculate it's physical properties. The result can be seen in Figure 21.

Calculate Center of Mass

In the next step we calculate the center of mass of the object, using the voxelized object. For that, we use the definition of center of mass (equation 4.1). We consider that density is uniform foe the entire object, and this the mass for each voxel will be the same.

$$\mathbf{cm} = \frac{\sum m_i * r_i}{\sum m_i} \tag{4.1}$$

Normalize Coordinates Points around center of mass

After calculating the center of mass coordinates, we centralize all the vertices so that the origin of the mesh will be the center of mass.

Calculate Moment Of Inertia Matrix

We also calculate the moment of inertia matrix around the center of mass. For this, we follow the definition as defined by equations in 4.2. The .URDF file format follows a

negative convention for the matrix (ROS, 2023).

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{4.2}$$

$$I_{xx} = \sum_{k=1}^{N} m_k(y_k^2 + z_k^2) \qquad I_{xy} \stackrel{def}{=} \sum_{k=1}^{N} m_k x_k y_k$$

$$I_{yy} = \sum_{k=1}^{N} m_k(x_k^2 + z_k^2) \qquad I_{xz} \stackrel{def}{=} \sum_{k=1}^{N} m_k y_k z_k$$

$$I_{zz} = \sum_{k=1}^{N} m_k(x_k^2 + y_k^2) \qquad I_{yz} \stackrel{def}{=} \sum_{k=1}^{N} m_k y_k z_k$$

Use v-hacd algorithm to simplify collision surface

The morphological complexity of the collision objects must be taken into consideration so that we obtain believable simulation results.

As a rule of thumb, the simpler a collision object is, the more reliable the dynamic data will be obtained for the simulation. That is because the probability of false force vectors appearing is higher in complex geometries. This is also done to speed up the time to calculate the collision. A throughout explanation of how the collision detection algorithm works can be found in Weller (2013). We use the V-HACD algorithm (LENGYEL, 2016) to obtain objects with these desired characteristics. The result can be seen in figure 21.

Generate .URDF file for each object

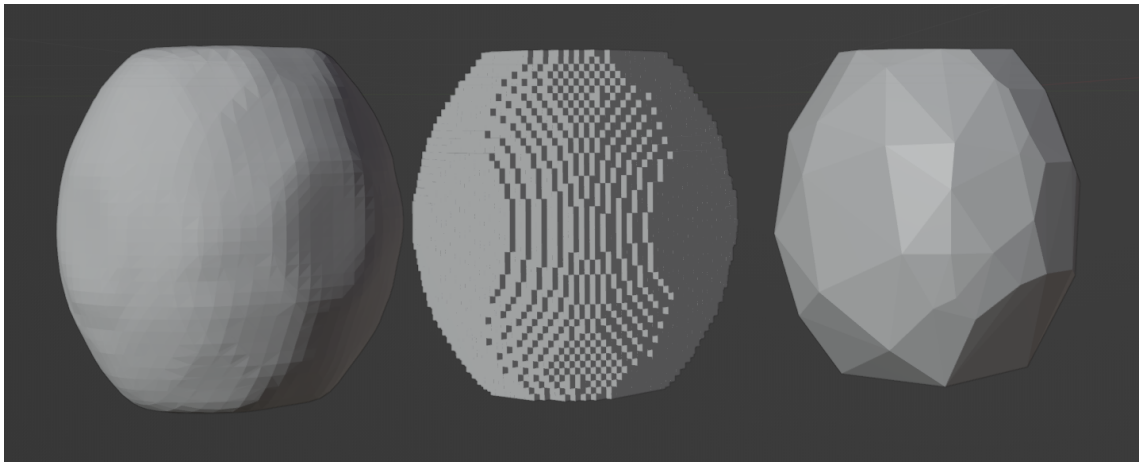
At the end of the pipeline, we generate the .URDF file that references both visual and collision objects, contains the inertia matrix and object's mass.

4.1.3.2 Robot for Simulation

For performing simulation grasping, a simple cartesian robot was created in the .URDF format file. Since for simulation purposes the area where the objects can be is limited, and our simulations are limited to top-down grasps, a cartesian robot is enough to suffice the requirements for the simulation (figure 22 shows the model of the robot used in simulation).

The motors were controlled in Velocity Control mode, and grasping logic was developed so that the robot is continuously in one state until the next state. The gripper size was chosen so it would be the double of the size of the objects.

Figure 21 – Objects transformations



Source: Author. From left to right: **(a)** Original Object Shape. **(b)** Voxelized Object. **(c)** Collision object after the v-hacd algorithm

4.2 Laboratory Tests Setup

4.2.1 KUKA Communication

This robotic system is dependent on several agents that must work together to complete the grasping task. The Camera, the Neural Network, the Gripper and the Robot must act in a synchronized manner. We explain how the communication is implemented in a UML sequence diagram. The camera is directly connected to the computer, and the visual data is processed in the computer as well. The Gripper has its own embedded system that controls the motor, and the robot is controlled by its own computer. Figure 23 shows the complete setup for lab experiments.

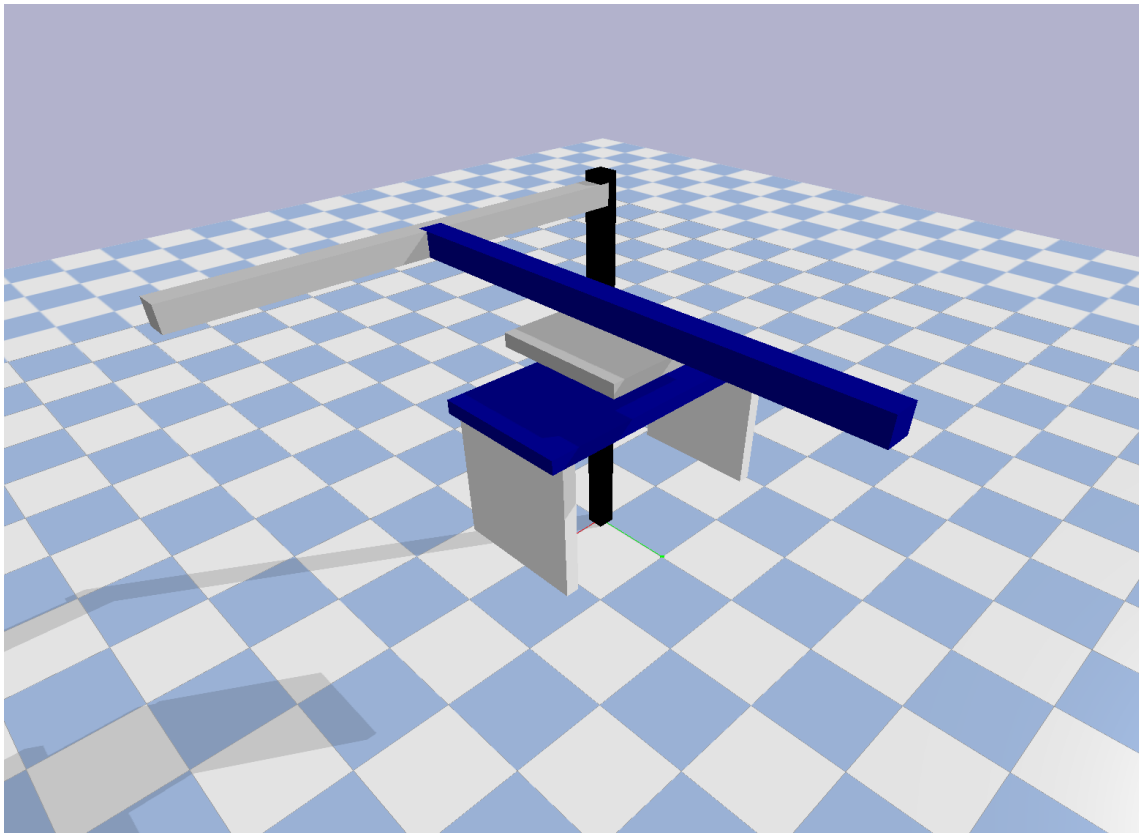
There are two stages for a grasp: the calibration stage and grasp stage. In the calibration stage (Figure 24), the robot goes to an initial known position and the gripper performs a calibration procedure, so that the grasp stage can start. In the grasp stage (Figure 25), the camera obtains an image that is sent to the neural network to process, the network computes and returns a grasp proposal, and the robot attempts to perform the grasp. The grasp is considered successful if the robot grasps the object and drops it at a specific drop point.

4.2.2 GG-CNN

To make possible the comparison between previous datasets and the generated dataset, we use the GG-CNN network without modifications. The main reasons this network was chosen are:

1. It is a Generative Neural Network. That means that it does not depend on additional algorithms to generate grasps

Figure 22 – Cartesian Robot used in simulations



Source: Author

2. The network can take as an input RGB data, depth data or both
3. The network outputs a probability map for the entire image, allowing for more complete analysis of how it is processing the data

4.2.3 Camera Data Processing

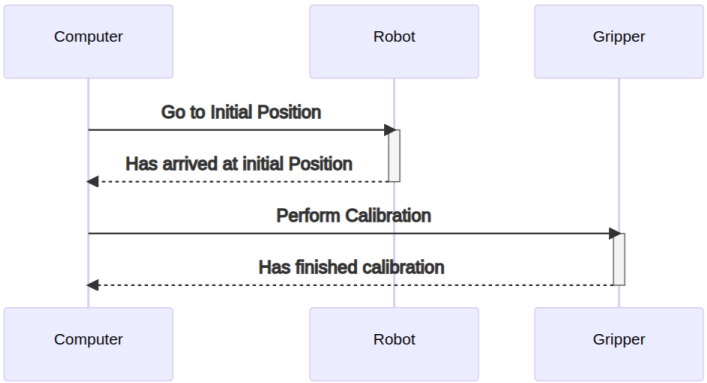
Before the camera is ready for use, it is necessary to calibrate the camera so that the depth sensor can properly calculate depth data. In figure 26 it can be seen how a calibration camera can be prone to depth noise.

In the Real Sense camera, the depth data has a slightly larger field of view than the RGB camera. Since we need the same image coordinates for both channels so that the neural network can be properly trained, it is necessary to align the data from both channels. This feature is available in the Real Sense SDK.

Figure 23 – Lab Setup for Tests

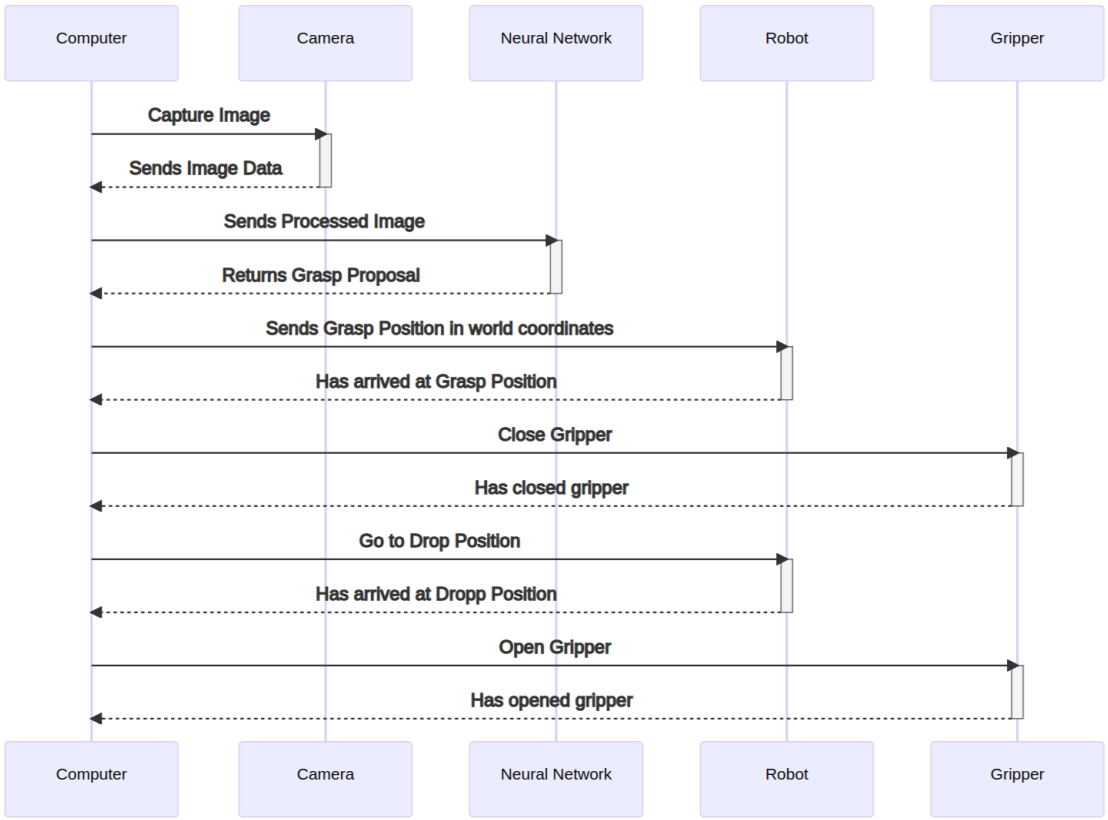
Source: Author

Figure 24 – Calibration Stage UML Sequence Diagram



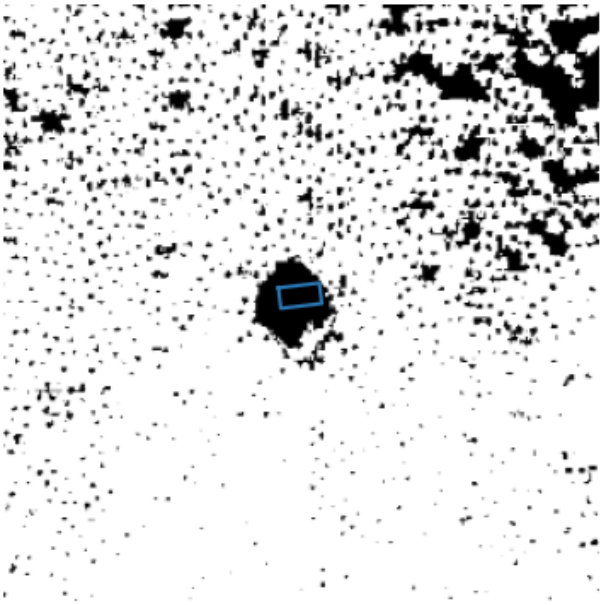
Source: Author

Figure 25 – Grasp Stage UML Sequence Diagram



Source: Author

Figure 26 – Depth Noise



Source: Author

5 RESULTS

5.1 Baseline for GG-CNN

First, it is necessary to measure the performance of the GG-CNN network, we calculated the number of successful grasps that the network was able to identify. For this computer evaluation, we used a metric similar to the one defined in the Cornell dataset (see 2.10.1), while training in the Jacquard dataset (see 2.10.2). Here, we evaluate a grasp as successful if the IoU metric between the proposed grasp and the ground-truth label is over 0.25 and if the angle of the grasp is within a difference of 30 degrees of the ground-truth label.

We performed three experiments to evaluate how much each input channel is relevant to the output of the network. For the first experiment, we trained the network with both RGB and Depth data. The second experiment was trained only on depth data and the last one only on RGB data. For each experiment, we trained the network for 30 epochs. We rank only the grasp that the network considers most successful.

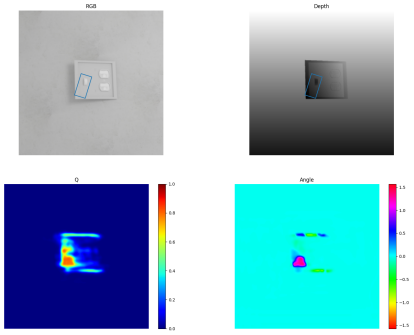
For evaluation, we tested on 465 samples. When using both data, we obtained a success rate of 86%, using only depth data, we also obtain a rate of 86% and when using only RGB data, the success rate is 77%. A few examples of generated grasp information can be seen in Figures 27 and 28.

These results indicate that the network is biased towards depth data, having more impact on the end-result. This behavior could also be seen in Lab Experiments. For example, when the camera was not calibrated, or when there was a mismatch between the camera and the plane where the objects were, the networked tended to use data from the depth channel, even when there was a clear vision of the object in the RGB channel. This behavior can be seen in figures 35 and 36.

5.2 Data Generation Pipeline Results

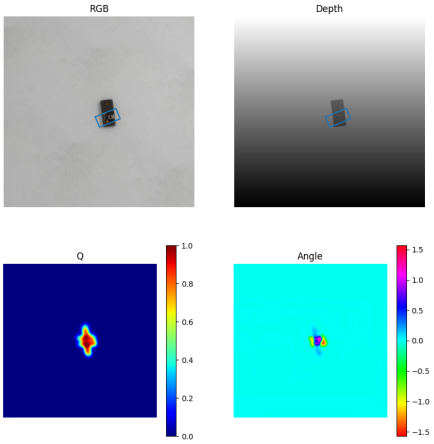
The pipeline was used to generate the 5 instances of each object (49 objects in total). Not all generated data had a valid configuration for testing the network, so this data was manually excluded from the dataset. The Figures 29, 30, 31 and 32 show a sample result from the pipeline. All images were generated to have the resolution of 1024 x 1024 pixels. From this point forward, the text will refer to the generated dataset as customized dataset.

Figure 27 – GG-CNN trained on Jacquard Dataset - Example 1



Source: Author

Figure 28 – GG-CNN trained on Jacquard Dataset - Example 2



Source: Author

5.3 Simulation Results

5.3.1 GG-CNN trained on Customized Dataset

We trained the network on the customized dataset using both rgb and depth channels. We obtain a rate of success of 90%. We use a fixed grasp annotation size of 340x170 pixels, which is approximately the size of the gripper (2x1 meters) in relation to the workspace size (6x6 meters). Examples of generated grasps can be seen in figures 33 and 34.

It is noticeable, if compared to figures 27 and 28, that the network cannot determine the object as easily as it could when trained with the jacquard dataset. Most likely this is because, unlike the jacquard dataset pipeline, we did not change the gripper size and test the grasp for different sizes. The result is that the network cannot as easily determine the borders of the object. Still, for the define gripper size, the network presented a high rate of success.

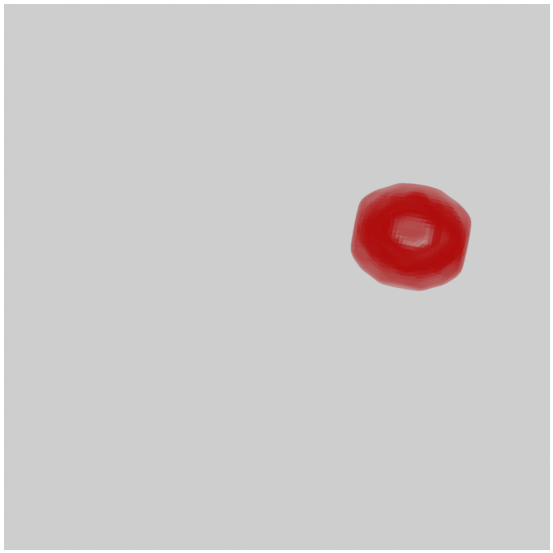
5.4 Lab Experiments

5.4.1 Baseline for Lab Experiments

For the laboratory experiments, we test both the network trained on jacquard data and the one trained on customized data. Before the data can be obtained, we have to make sure that the camera (specially the depth channel) is correctly calibrated.

Just as indicated in section 5.1, we confirm that the network is heavily based on depth data. In the figures 35 and 36 we can see the in the upper row input data that

Figure 29 – RGB Generated Image



Source: Author

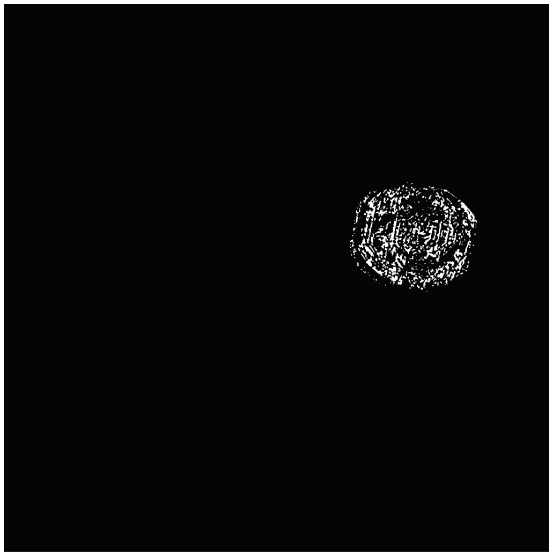
Figure 30 – Segmentation Data

Source: Author

Figure 31 – Perfect Depth

Source: Author

Figure 32 – Stereo Depth

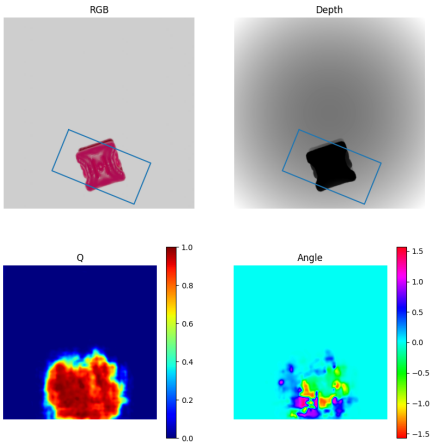


Source: Author

the camera has acquired (RGB and depth channels), as well as the grasp that the neural network has decided on. In the second row we can see the Q measure (a pixel-wise measure of confidence that the network has about the grasp) and the respective angle measure in radians (for each pixel, the respective grasp's angle).

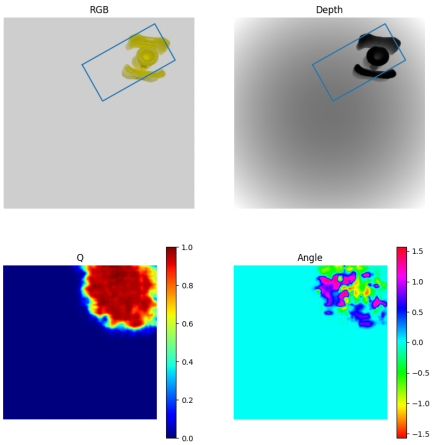
In Figure 35, we can see how with a clear RGB data but an unclear depth channel, the network was unable to correctly predict a good grasp. The Q measure, is highly correlated to the depth channel information. In Figure 36 we can notice a similar pattern.

Figure 33 – Customized Dataset - Grasp 1



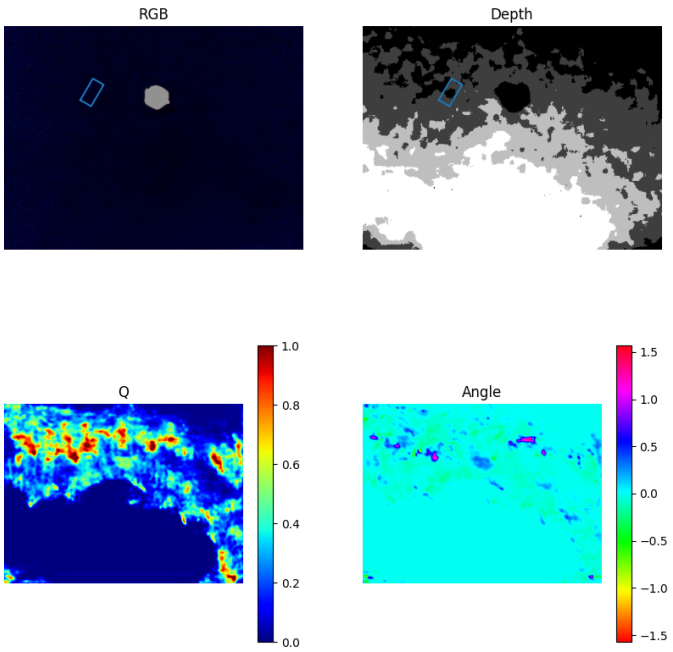
Source: Author

Figure 34 – Customized Dataset - Grasp 2



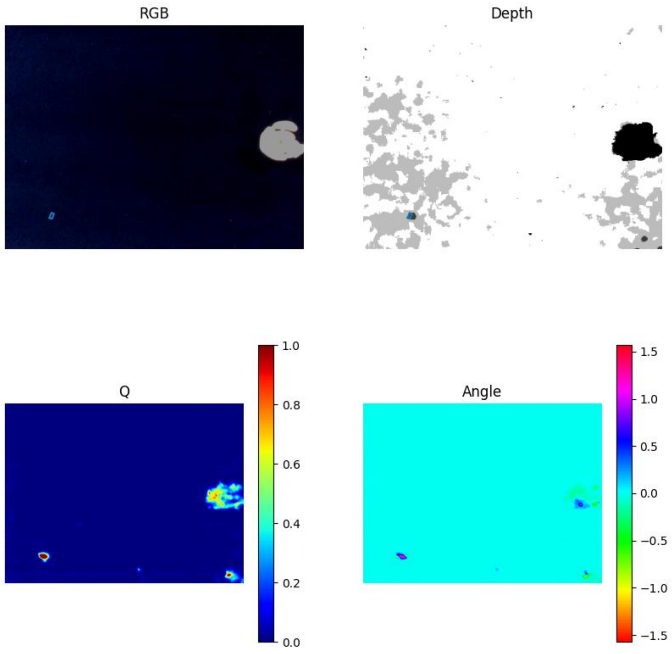
Source: Author

Figure 35 – Bias towards depth 1



Source: Author

Figure 36 – Bias towards depth 2



Source: Author

5.4.2 GG-CNN trained on Jacquard

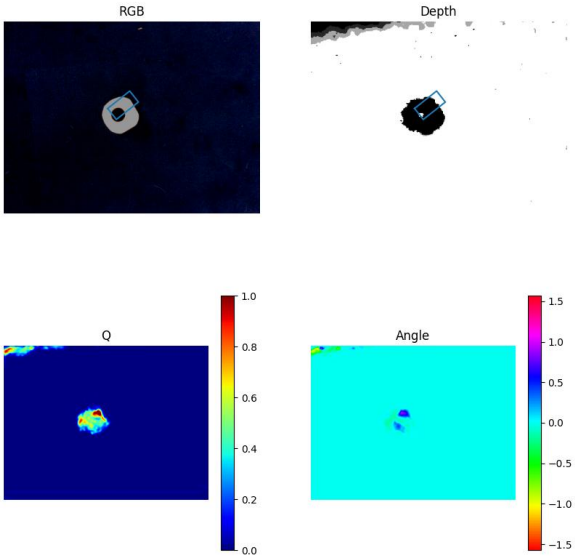
We first tested the four objects trained on the jacquard dataset. For each object, we attempted to perform the grasp 10 times. The results can be shown in table 2. The rate of success is only slightly lower than the overall value of 86% defined in section 5.1

A few grasp propositions can be seen in Figures 42, 43, 44 and 45. The network's sensibility to noise is noticeable in the images. During the experiments, a few grasp proposals were off-track because of the noise error, as can be seen in Figure 41.

Table 2 – Rate of Success - GG-CNN trained on jacquard dataset

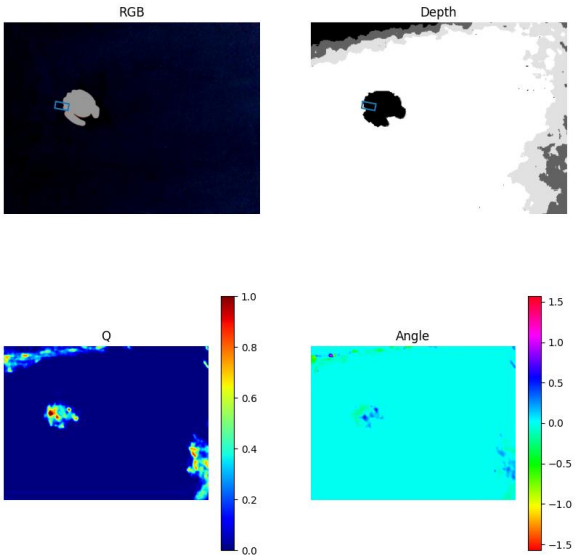
Object	**Rate of success**
A5	90%
A1	80%
E5	70%
B1	70%

Figure 37 – Grasp Example 1



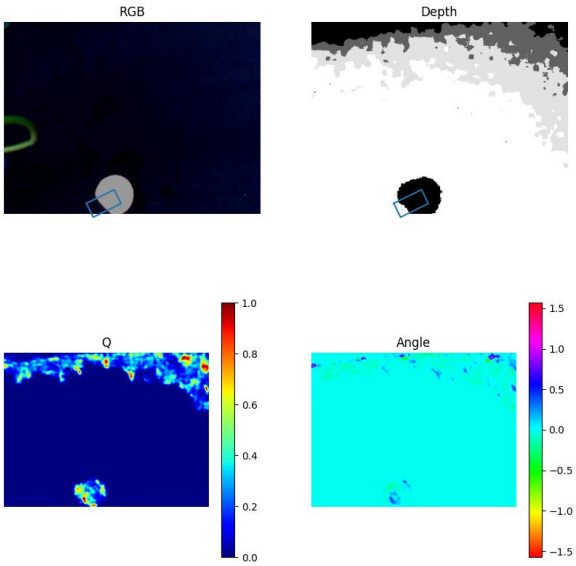
Source: Author

Figure 38 – Grasp Example 2



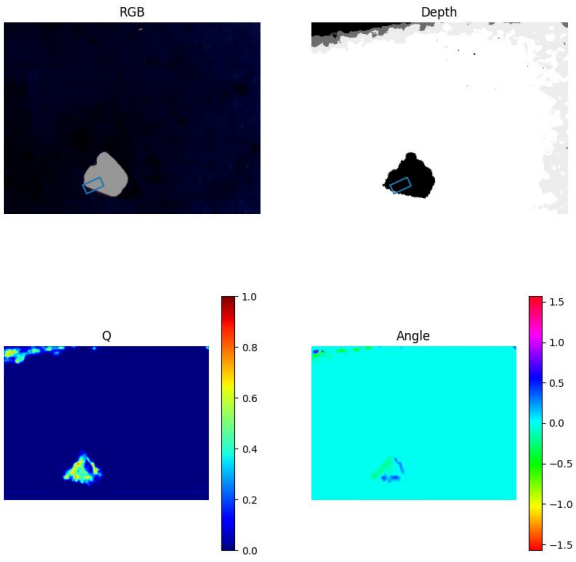
Source: Author

Figure 39 – Grasp Example 3



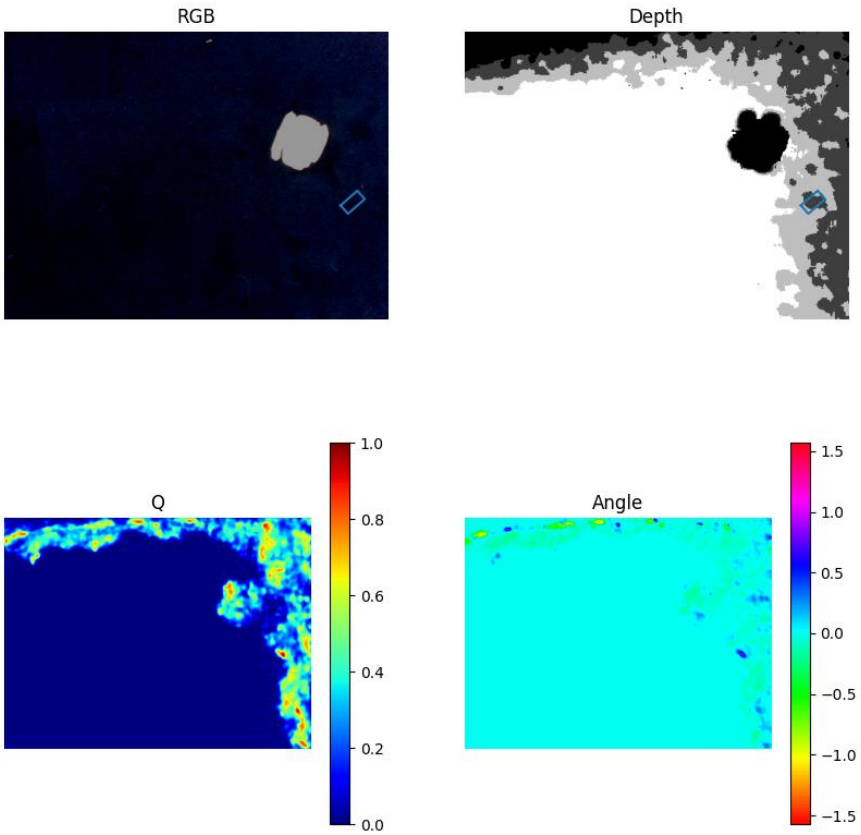
Source: Author

Figure 40 – Grasp Example 4



Source: Author

Figure 41 – Grasp Example - False grasp proposition



Source: Author

Table 3 – Rate of Success - GG-CNN trained on customized dataset

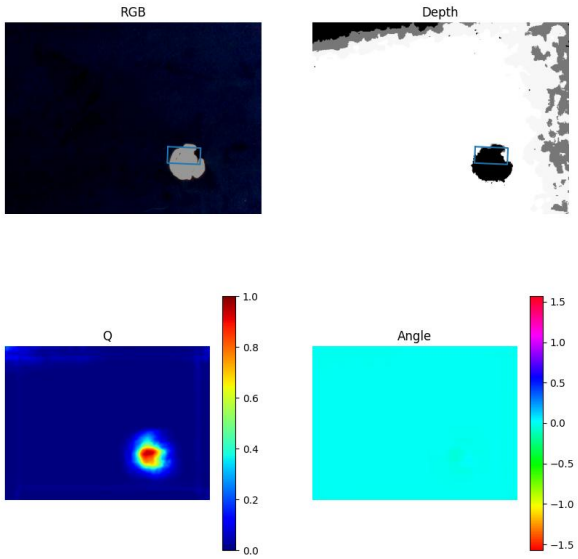
Object	**Rate of success**	**Previous Rate of sucess**
A5	30%	90%
A1	90%	80%
E5	90%	70%
B1	70%	70%

5.4.3 GG-CNN trained on Customized Dataset

The network trained on the customized dataset performed not as well as the one trained on the jacquard. This is specially visible in the results regarding the A5 object, that presented a drop from 90% to 30%. The other objects presented a similar rate of success. The network was not able to provide a precise grasp, and there was collision with the object, making the grasp attempt fail.

Another notable difference is that the network was not as sensible to depth noise. This can be seen in Figures 42, 43, 44 and 45. Also, no false grasps such as the one shown in Figure 41 did not happen in experiments.

Figure 42 – Grasp Example 5



Source: Author

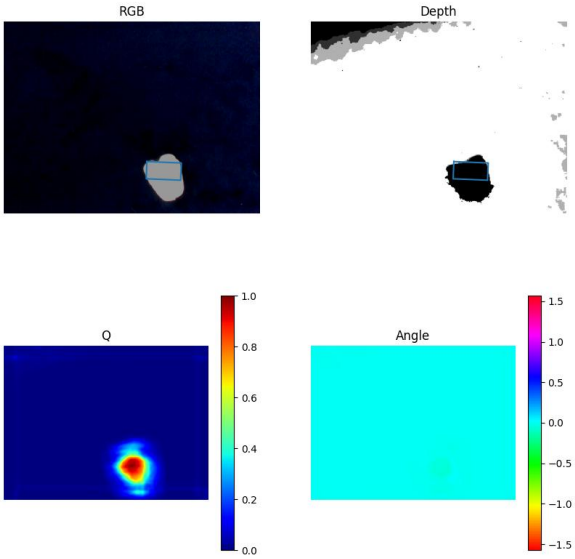
5.4.4 Comparison between both networks

Overall, the network trained on the customized dataset did not present a performance as good as the one trained on the jacquard dataset. This can be seen specially in the results related to the A5 object. Between the four objects, it is the one that has the most complex geometry. This may be related to the reason why the network failed more in attempting this grasp, however, more tests are required to understand this phenomenon better.

Overall, the network trained on the customized dataset was more robust to noise, albeit less precise than the network trained on the jacquard dataset. Not being able to identify the object as clearly as the first network. This behavior can be clearly seen by comparing the Q maps between the two networks. The first network is able to locate the object more precisely, but presents more susceptibility to the depth information. Because this network has learned that even very small objects can have a viable grasp, it's Q map is highly "fragmented".

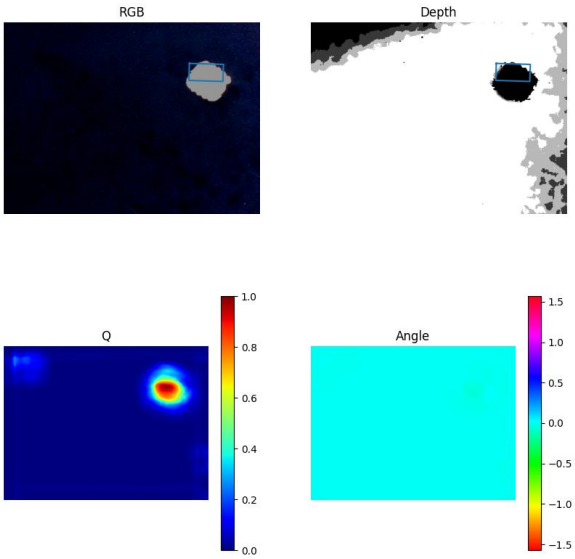
The second network was trained on a dataset where the grasp size was not changed.

Figure 43 – Grasp Example 6



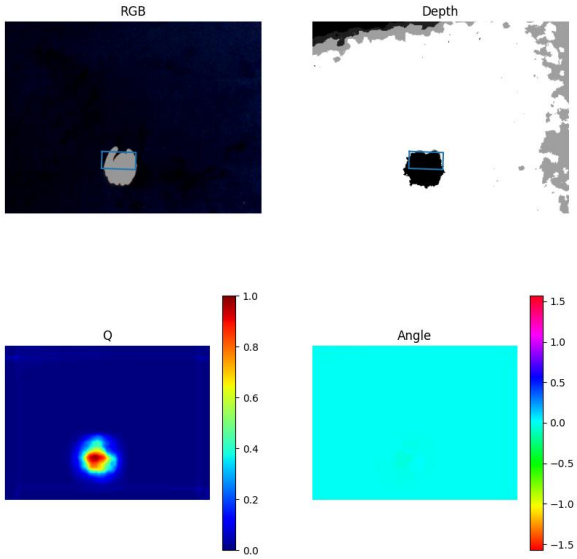
Source: Author

Figure 44 – Grasp Example 7



Source: Author

Figure 45 – Grasp Example 8



Source: Author

That made the network to not be able to define as precisely the position of the object, but it also made the network to only consider as viable grasps certain types of noise data, making it more robust to depth noise. It is valid to highlight that the ratio between gripper opening and object size was very high (gripper opening: 7 cm / object size: 4 cm), and the network might have had worse results if the gripper opening was smaller.

These results show that a trade-off between precision and robustness can be achieved by adjusting for gripper size.

6 CONCLUSIONS

In this work, we present a pipeline for generating customized dataset for robotic grasping. We compare it to state-of-art works by using a state-of-art generative network in both simulation and lab experiments, proving that the current pipeline is able to generate data for real-world applications.

This pipeline is limited to the same scope as the original paper, with the difference that it allows the creation of a dataset customized for a set of objects. This approach allows for datasets customized for the application, where classification and object detection can be generated together with grasps ground-truths. It also allows for variation in parameters, so there can be a trade-off between precision and robustness. Furthermore, there are a number of improvements that can still be applied to improve the results of networks trained in this dataset. More features can be added to the pipeline, such as allowing for loading of different textures, data randomization strategies, improving stereo vision data quality, generating data for other gripper's shapes and implementing active vision strategies.

Finally, it shows the potentials that artificial data, rendering and simulation have for training neural networks for robotics.

BIBLIOGRAPHY

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBRISO10218-1: Robôs e dispositivos robóticos - Requisitos de segurança para robôs industriais Parte 1: Robôs**. 2018. Rio de Janeiro.

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (slam): Part ii. **IEEE Robotics & Automation Magazine**, IEEE, v. 13, n. 3, p. 108–117, 2006.

BOUSMALIS, K.; IRPAN, A.; WOHLHART, P.; BAI, Y.; KELCEY, M.; KALAKRISHNAN, M.; DOWNS, L.; IBARZ, J.; PASTOR, P.; KONOLIGE, K. et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In: IEEE. **2018 IEEE international conference on robotics and automation (ICRA)**. [S.l.], 2018. p. 4243–4250.

BRACHMANN, E.; MICHEL, F.; KRULL, A.; YANG, M. Y.; GUMHOLD, S. et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016. p. 3364–3372.

CALLI, B.; SINGH, A.; WALSMAN, A.; SRINIVASA, S.; ABBEEL, P.; DOLLAR, A. M. The ycb object and model set: Towards common benchmarks for manipulation research. In: IEEE. **2015 international conference on advanced robotics (ICAR)**. [S.l.], 2015. p. 510–517.

CHANG, A. X.; FUNKHOUSER, T.; GUIBAS, L.; HANRAHAN, P.; HUANG, Q.; LI, Z.; SAVARESE, S.; SAVVA, M.; SONG, S.; SU, H. et al. Shapenet: An information-rich 3d model repository. **arXiv preprint arXiv:1512.03012**, 2015.

COUMANS, E.; BAI, Y. **PyBullet, a Python module for physics simulation for games, robotics and machine learning**. 2016–2022. <http://pybullet.org>.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 248–255. ISSN 1063-6919.

DENNINGER, M.; SUNDERMEYER, M.; WINKELBAUER, D.; ZIDAN, Y.; OLEFIR, D.; ELBADRAWY, M.; LODHI, A.; KATAM, H. Blenderproc. **arXiv preprint arXiv:1911.01911**, 2019.

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.

DO, T.-T.; CAI, M.; PHAM, T.; REID, I. Deep-6dpose: Recovering 6d object pose from a single rgb image. **arXiv preprint arXiv:1802.10367**, 2018.

DOŠILOVIĆ, F. K.; BRČIĆ, M.; HLUPIĆ, N. Explainable artificial intelligence: A survey. In: IEEE. **2018 41st International convention on information and**

communication technology, electronics and microelectronics (MIPRO). [S.l.], 2018. p. 0210–0215.

DU, G.; WANG, K.; LIAN, S. Vision-based robotic grasping from object localization, pose estimation, grasp detection to motion planning: A review. **arXiv preprint arXiv:1905.06658**, 2019.

DUANE, C. B. Close-range camera calibration. **Photogramm. Eng**, v. 37, n. 8, p. 855–866, 1971.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **IEEE robotics & automation magazine**, IEEE, v. 13, n. 2, p. 99–110, 2006.

EJTTTJE. **Small object jitter: approaches summary?** 2011. Disponível em: <https://pybullet.org/Bullet/phpBB3/viewtopic.php?t=7402>. Acesso em: 21-03-2023.

FERRARI, C.; CANNY, J. F. Planning optimal grasps. In: **ICRA**. [S.l.: s.n.], 1992. v. 3, n. 4, p. 6.

GAIDON, A.; WANG, Q.; CABON, Y.; VIG, E. Virtual worlds as proxy for multi-object tracking analysis. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 4340–4349.

GEIRHOS, R.; RUBISCH, P.; MICHAELIS, C.; BETHGE, M.; WICHMANN, F. A.; BRENDEL, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. **arXiv preprint arXiv:1811.12231**, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GREFF, K.; BELLETTI, F.; BEYER, L.; DOERSCH, C.; DU, Y.; DUCKWORTH, D.; FLEET, D. J.; GNANAPRAGASAM, D.; GOLEMO, F.; HERRMANN, C.; KIPF, T.; KUNDU, A.; LAGUN, D.; LARADJI, I.; LIU, H.-T. D.; MEYER, H.; MIAO, Y.; NOWROUZEZAHRAI, D.; OZTIRELI, C.; POT, E.; RADWAN, N.; REBAIN, D.; SABOUR, S.; SAJJADI, M. S. M.; SELA, M.; SITZMANN, V.; STONE, A.; SUN, D.; VORA, S.; WANG, Z.; WU, T.; YI, K. M.; ZHONG, F.; TAGLIASACCHI, A. Kubric: a scalable dataset generator. 2022.

HARTLEY, R.; ZISSERMAN, A. **Multiple view geometry in computer vision**. [S.l.]: Cambridge university press, 2003.

HE, K.; ZHANG, X.; REN, S.; SUN, J. 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.

HINTERSTOISSER, S.; LEPETIT, V.; ILIC, S.; HOLZER, S.; BRADSKI, G.; KONOLIGE, K.; NAVAB, N. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: SPRINGER. **Asian conference on computer vision**. [S.l.], 2012. p. 548–562.

HINTERSTOISSER, S.; LEPETIT, V.; RAJKUMAR, N.; KONOLIGE, K. Going further with point pair features. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 834–848.

HINTERSTOISSER STEFAN HOLZER, C. C. S. I. K. K. N. N. V. L. S. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: **2011 international conference on computer vision. IEEE**. [S.l.: s.n.], 2011.

HIRSCHMULLER, H. Stereo processing by semiglobal matching and mutual information. **IEEE Transactions on pattern analysis and machine intelligence**, IEEE, v. 30, n. 2, p. 328–341, 2007.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HODAN, T.; HALUZA, P.; OBDRŽÁLEK, Š.; MATAS, J.; LOURAKIS, M.; ZABULIS, X. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In: IEEE. **2017 IEEE Winter Conference on Applications of Computer Vision (WACV)**. [S.l.], 2017. p. 880–888.

HODAŇ, T.; MATAS, J.; OBDRŽÁLEK, Š. On evaluation of 6d object pose estimation. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2016. p. 606–619.

HU, Y.; HUGONOT, J.; FUA, P.; SALZMANN, M. Segmentation-driven 6d object pose estimation. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 3385–3394.

HVILSHøJ, M. et al;. Autonomous industrial mobile manipulation (aimm): past, present and future. **Industrial Robot: An International Journal**, v. 39, n. 2, p. 120–135, 2012.

Intel. **Intel Realsense SDK Github page** . 2023. Disponível em: <https://github.com/IntelRealSense/librealsense>. Acesso em: 11-01-2023.

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.

KALASHNIKOV, D.; IRPAN, A.; PASTOR, P.; IBARZ, J.; HERZOG, A.; JANG, E.; QUILLEN, D.; HOLLY, E.; KALAKRISHNAN, M.; VANHOUCKE, V. et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In: PMLR. **Conference on Robot Learning**. [S.l.], 2018. p. 651–673.

KEHL, W.; MANHARDT, F.; TOMBARI, F.; ILIC, S.; NAVAB, N. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2017. p. 1521–1529.

KESELMAN, L.; WOODFILL, J. I.; GRUNNET-JEPSEN, A.; BHOWMIK, A. Intel realsense stereoscopic depth cameras. In: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. [S.l.: s.n.], 2017. p. 1–10.

KINGMA, D. P.; BA, J. **Adam: A method for stochastic optimization. CoRR abs/1412.6980**. 2014.

KLEEBERGER, K.; BORMANN, R.; KRAUS, W.; HUBER, M. F. A survey on learning-based robotic grasping. **Current Robotics Reports**, Springer, v. 1, n. 4, p. 239–249, 2020.

KÖRBER, M.; LANGE, J.; REDISKE, S.; STEINMANN, S.; GLÜCK, R. Comparing popular simulation environments in the scope of robotics and reinforcement learning. **arXiv preprint arXiv:2103.04616**, 2021.

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.

KUKA. **Spez LBR iiwa V7**. 2016.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LENGYEL, E. **Game engine gems 3**. [S.l.]: CRC Press, 2016.

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.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o (n) solution to the pnp problem. **International journal of computer vision**, Springer, v. 81, n. 2, p. 155, 2009.

LI, S.; XU, C.; XIE, M. A robust o (n) solution to the perspective-n-point problem. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 34, n. 7, p. 1444–1450, 2012.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755.

LOING, V.; MARLET, R.; AUBRY, M. Virtual training for a real application: Accurate object-robot relative localization without calibration. **International Journal of Computer Vision**, Springer, v. 126, n. 9, p. 1045–1060, 2018.

LOOP, C.; ZHANG, Z. Computing rectifying homographies for stereo vision. In: IEEE. **Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)**. [S.l.], 1999. v. 1, p. 125–131.

LOUHICHI, H.; FOURNEL, T.; LAVEST, J.; AISSIA, H. B. Self-calibration of scheimpflug cameras: an easy protocol. **Measurement Science and Technology**, IOP Publishing, v. 18, n. 8, p. 2616, 2007.

MAHLER, J.; LIANG, J.; NIYAZ, S.; LASKEY, M.; DOAN, R.; LIU, X.; OJEA, J. A.; GOLDBERG, K. 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.; MATL, M.; LIU, X.; LI, A.; GEALY, D.; GOLDBERG, K. 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. 5620–5627.

MAHLER, J.; MATL, M.; SATISH, V.; DANIELCZUK, M.; DEROSE, B.; MCKINLEY, S.; GOLDBERG, K. Learning ambidextrous robot grasping policies. **Science Robotics**, American Association for the Advancement of Science, v. 4, n. 26, p. eaau4984, 2019.

MAHLER, J.; POKORNY, F. T.; HOU, B.; RODERICK, M.; LASKEY, M.; AUBRY, M.; KOHLHOFF, K.; KRÖGER, T.; KUFFNER, J.; GOLDBERG, K. 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.

MATTHIES, L.; BROCKERS, R.; KUWATA, Y.; WEISS, S. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In: IEEE. **2014 IEEE international conference on robotics and automation (ICRA)**. [S.l.], 2014. p. 3242–3249.

MCCORMAC, J.; HANDA, A.; LEUTENEGGER, S.; DAVISON, A. J. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. **arXiv preprint arXiv:1612.05079**, 2016.

MOKARAM, S.; AITKEN, J. M.; MARTINEZ-HERNANDEZ, U.; EIMONTAITE, I.; CAMERON, D.; ROLPH, J.; GWILT, I.; MCAREE, O.; LAW, J. A ros-integrated api for the kuka lbr iiwa collaborative robot. **IFAC-PapersOnLine**, Elsevier, v. 50, n. 1, p. 15859–15864, 2017.

MORRISON, D. **Robotic grasping in unstructured and dynamic environments**. 2021. Tese (Doutorado) — Queensland University of Technology, 2021.

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, v. 5, n. 3, p. 4368–4375, 2020.

_____. 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.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th international conference on machine learning (ICML-10)**. [S.l.: s.n.], 2010. p. 807–814.

NEWBURY, R.; GU, M.; CHUMBLEY, L.; MOUSAVIAN, A.; EPPNER, C.; LEITNER, J.; BOHG, J.; MORALES, A.; ASFOUR, T.; KRAGIC, D. et al. Deep learning approaches to grasp synthesis: A review. **arXiv preprint arXiv:2207.02556**, 2022.

OPENCV. 2020. Disponível em: <https://opencv.org>. Acesso em: 14 mar. 2020.

OpenCV. **Camera Calibration and 3D reconstruction** . 2023. Disponível em: <https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html>. Acesso em: 03-01-2023.

POKORNY, F. T.; KRAGIC, D. Classical grasp quality evaluation: New algorithms and theory. In: IEEE. **2013 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2013. p. 3493–3500.

PYTORCH. 2020. Disponível em: <https://pytorch.org>. Acesso em: 14 mar. 2020.

RAD, M.; LEPETIT, V. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2017. p. 3828–3836.

RENSINK, R. A. The dynamic representation of scenes. **Visual cognition**, Taylor & Francis, v. 7, n. 1-3, p. 17–42, 2000.

ROS. **URDF documentation**. 2023. Disponível em: <http://wiki.ros.org/urdf/XML/link>. Acesso em: 14-01-2023.

ROS, G.; SELLART, L.; MATERZYNSKA, J.; VAZQUEZ, D.; LOPEZ, A. M. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 3234–3243.

ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. [S.l.]: Cornell Aeronautical Laboratory, 1957.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [S.l.], 1985.

SAHBANI, A.; EL-KHOURY, S.; BIDAUD, P. An overview of 3d object grasp synthesis algorithms. **Robotics and Autonomous Systems**, Elsevier, v. 60, n. 3, p. 326–336, 2012.

SAHIN, C.; GARCIA-HERNANDO, G.; SOCK, J.; KIM, T.-K. A review on object pose recovery: From 3d bounding box detectors to full 6d pose estimators. **Image and Vision Computing**, Elsevier, p. 103898, 2020.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SÜNDERHAUF, N.; BROCK, O.; SCHEIRER, W.; HADSELL, R.; FOX, D.; LEITNER, J.; UPCROFT, B.; ABBEEL, P.; BURGARD, W.; MILFORD, M. et al. The limits and potentials of deep learning for robotics. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 37, n. 4-5, p. 405–420, 2018.

TASORA. **Very small objects**. 2011. Disponível em: <https://pybullet.org/Bullet/phpBB3/viewtopic.php?t=5037>. Acesso em: 21-03-2023.

TEKIN, B.; SINHA, S. N.; FUA, P. Real-time seamless single shot 6d object pose prediction. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 292–301.

TENSORFLOW. 2020. Disponível em: <https://www.tensorflow.org>. Acesso em: 14 mar. 2020.

TOBIN, J.; FONG, R.; RAY, A.; SCHNEIDER, J.; ZAREMBA, W.; ABBEEL, P. 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.

TREMBLAY, J.; PRAKASH, A.; ACUNA, D.; BROPHY, M.; JAMPANI, V.; ANIL, C.; TO, T.; CAMERACCI, E.; BOOCHOON, S.; BIRCHFIELD, S. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2018. p. 969–977.

TREMBLAY, J.; TO, T.; SUNDARALINGAM, B.; XIANG, Y.; FOX, D.; BIRCHFIELD, S. Deep object pose estimation for semantic robotic grasping of household objects. **arXiv preprint arXiv:1809.10790**, 2018.

TSIRIKOGLOU, A.; KRONANDER, J.; WRENNINGE, M.; UNGER, J. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. **arXiv preprint arXiv:1710.06270**, 2017.

WEISZ, J.; ALLEN, P. K. Pose error robust grasping from contact wrench space metrics. In: IEEE. **2012 IEEE international conference on robotics and automation**. [S.l.], 2012. p. 557–562.

WELLER, R. A brief overview of collision detection. **New Geometric Data Structures for Collision Detection and Haptics**, Springer, p. 9–46, 2013.

WIDROW, B. **An adaptive'adaline'neuron using chemical'memistors', 1553-1552**. [S.l.]: Stanford Electronics Laboratories, 1960.

XIANG, Y.; SCHMIDT, T.; NARAYANAN, V.; FOX, D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. **arXiv preprint arXiv:1711.00199**, 2017.

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. **arXiv preprint arXiv:1505.00853**, 2015.

ZENG, A.; SONG, S.; WELKER, S.; LEE, J.; RODRIGUEZ, A.; FUNKHOUSER, T. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In: IEEE. **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2018. p. 4238–4245.

Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object detection with deep learning: A review. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–21, 2019. ISSN 2162-237X.