UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SÍNTESIS DE AGARRES PARA GRASPING ROBÓTICO A PARTIR DE NUBES DE PUNTOS 3D

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

RODRIGO ANDRÉS SCHULZ SERRANO

PROFESOR GUÍA:
PABLO GUERRERO PÉREZ

PROFESOR GUÍA:
BENJAMÍN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
AIDAN HOGAN
JOSE MANUEL SAAVEDRA
RICARDO ÑANCULEF

Este trabajo ha sido parcialmente financiado por Proyecto FONDECYT 1140783

SANTIAGO DE CHILE
2017

Resumen

A lo largo de la historia, la humanidad se ha esforzado por desarrollar máquinas que faciliten la vida de las personas, frecuentemente concentrándose en producir máquinas autónomas que logren reemplazar a los humanos en tareas específicas. En dicho contexto, la Robótica es a menudo identificada como la disciplina que mejor simboliza el desarrollo de máquinas autónomas. En general, un robot autónomo debe exhibir al menos 3 capacidades: i) percepción de su entorno, ii) navegación y desplazamiento, e iii) interacción con su entorno. En particular, dentro de las formas interacción posibles, la manipulación de objetos es interesante, puesto que faculta al robot para modificar el ambiente de acuerdo con sus necesidades.

Para que un robot pueda manipular un objeto es necesario definir una estrategia de agarre, es decir, un punto de agarre y una parametrización del efector que permita realizar el agarre según las restricciones de la tarea en ejecución. La generación de dichas estrategias es conocida como *síntesis de agarres* y durante la última década ha sido abordada principalmente mediante el uso de información sensorial, con el fin de identificar las características del objeto que permiten definir una estrategia de agarre adecuada.

Esta tesis propone un descriptor local 3D original, denominado *Directed Curvature Histograms* (DCH), el cual describe la vecindad de un punto utilizando la forma en que varía la curvatura de la superficie en distintas direcciones. A diferencia de los enfoques basados en curvatura existentes en la literatura, DCH utiliza la dirección en la que varía la curvatura como un elemento significativo para la descripción de los datos.

El presente trabajo también propone un método para la síntesis de agarres que utiliza DCH, combinando técnicas de aprendizaje no-supervisado y supervisado. Tal método difiere de los enfoques actuales puesto que basa sus hipótesis en el estudio del comportamiento humano en la manipulación de objetos, presentado por Feix et al. [1]. Además, el método propuesto saca provecho de la información sensorial 3D disponible, aprendiendo simultáneamente a identificar puntos de agarre adecuados, así como parametrizaciones apropiadas para manipular un objeto arbitrario.

Los resultados experimentales muestran que DCH es capaz de reflejar las variaciones en la curvatura de una superficie, en diferentes direcciones. Tales resultados también muestran que la información suministrada por DCH permite que el método de síntesis de agarres propuesto pueda entrenar un clasificador (índice ROC igual a 0.961), con el objetivo de identificar los puntos de agarre y parametrizaciones adecuadas para manipular un objeto arbitrario.

Abstract

People have historically striven for the development of machines to aid in day-to-day life, frequently focusing on the production of autonomous machines that may even be capable of replacing humans in particular tasks. In such a context, Robotics is usually identified as the discipline that best symbolizes the quest for the development of autonomous machines. In general, to be autonomous a robot must exhibit at least 3 abilities: i) perception, ii) navigation and displacement, and iii) interaction with its environment. Particularly, among the possible ways of interaction, object manipulation is one of the most relevant choices, since it allows a robot to actively modify the environment according to the needs of the task in execution.

To be able to manipulate an object a robot must define a grasping strategy, that is a grasping point and a parameterization for the effector, which would allow the robot to perform the grasp according to the restrictions of the ongoing task. The generation of such strategies is known as *grasp synthesis* and over the last few years it has been approached mainly through the use of sensory data to identify the features of an object which allows the definition of a suitable grasping strategy.

This thesis proposes a novel local 3D descriptor, named *Directed Curvature Histograms* (DCH), which describes the vicinity of a point based on how the curvature of the surface changes along different directions. Unlike the curvature based approaches existing in the literature, DCH uses the direction of the curvature changes as a meaningful feature for the description.

This work also proposes a grasp synthesis method which uses DCH, combining non-supervised and supervised learning techniques. Such a method differs from the current approaches in that it bases its hypothesis on the human grasping behavior study presented by Feix et al. [1]. Additionally, this method takes advantage of the 3D sensory information available, simultaneously learning to identify suitable grasping points, as well as suitable parameterizations to grasp an arbitrary object.

The experimental results show that DCH is capable of reflecting the changes in the curvature of a surface, along with different directions. Such results also show that DCH provides information that allows the proposed grasping method to train a classifier (showing a ROC index equals to 0.961), which can be later used to identify what grasping points and parameterizations are suitable for grasping an arbitrary object.

Con profundo amor a mi madre, única constante en mi vida,

y a Viviana, fuente inagotable de amor, alegría y apoyo incondicional.

"You shall not pass!"

— Gandalf

...also some lecturers.

"I don't know what I'm doing, but my incompetence has never stopped my enthusiasm."

— *Woody Allen*

Acknowledgements

First, I would like to thank my family, and particularly my mother Susana, for her constant, unconditional and endless love and support. I would not be the person I am today without her help and her deep and infinite love.

I would like to also thank Viviana Gonzalez, the best thing in my life and the best partner somebody could ask for. Thank you for your love and support from the beginning of this journey, thank you for bearing with me along this path, and thank you for taking the time to understand me and love me. I love you above everything.

I would also like to say thanks to my advisers. Thanks to Professor Pablo Guerrero for always finding some time to talk, for his support and confidence when I was in doubt, and for all the valuable advice. Thanks to Professor Benjamín Bustos for all the valuable advice and support, for his guidance, and for taking the time to get involved in this thesis which I'm sure it seemed a bit off-topic.

Also, I would like to thank the revision committee, professors Aidan Hogan, Jose Manuel Saavedra and Ricardo Ñanculef, for their time and detailed comments on this work. And finally, last but not least, I would like to thank Ren Cerro for taking the time to proof-read this document and the papers involved.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machines that are capable of helping and assisting people in all kinds of tasks, effectively even replacing them to perform unpleasant tasks, has been one of the ambitions of the human race for a long time. Such ambition has motivated the development of countless inventions, has boosted the research of multiple topics and has transformed diverse aspects of day-to-day life.

In many ways, Robotics is probably the field that best symbolizes the quest for a machine capable of performing the same tasks that a person can. However, Robotics has proven to be a complex field to approach because of the high number and wide range of problems to resolve, in order to create an autonomous machine. Such reasons have driven the development of Robotics to be mainly constrained to some problems and environments. One example is the productive industry, where Robotics has found a niche for its development and, because of that, has been present for years. But even there, it has also been constrained to places where it can take advantage of highly structured and controlled environments, like production lines for instance. This can be explained in part because common human environments (a kitchen, a mall, a garden, etc.) are usually highly dynamic, unstructured and usually lack permanent reference points. In consequence, to successfully perform in such environments, a robot must have skills that allow it to dynamically interact with the environment, such as autonomous navigation, sensory perception, and object manipulation.

On the other hand, Machine Learning, despite being a fairly recent discipline, has been quickly developing and strongly increasing its presence in a wide range of activities over the last few years. It is no longer a novelty to get automatic recommendations of movies, TV shows, books, music and more, just by visiting a website or buying a product. It is also not uncommon to apply these techniques to diverse industry problems, such as parameter estimation, the forecast of variables or discovery of associations between variables.

To combine Robotics and Machine Learning seems to be an obvious mix on the quest for the development of autonomous machines. Machine Learning could be considered as one of the most flexible ways in which a robot could learn the skills that might allow it to perform more complex tasks such as grasping and manipulating objects.

This thesis addresses the robotic grasping problem, described by León et al. [2] as *"to determine the grasp required to carry out certain manipulation tasks on an object"*. In particular, this work focuses on the development of a method for the generation of strategies that would allow a robot to grasp an arbitrary object. Such a problem is usually known as grasp synthesis, and it is not new. Previous advances have been done in the area, focusing on the development of methods to define the correct pose of a robotic hand to grasp an object, or in the recognition of features that may ensure the proper grasp of an object. This thesis differs from the current approaches in that it aims to directly take advantage of 3D sensory data. With such a goal, this thesis proposes a novel 3D local descriptor, together with a method to approach the grasp synthesis problem, based on the use of such a descriptor.

1.1 Why Robotic Grasping?

Most of the natural abilities present in humans focus on perception, like sight, hearing and touch. Only a part of the natural abilities focus on motor skills, and typically they have to be developed as a person grows up. Among them, the ability to manipulate objects is likely to be the most relevant way humans have to modify their environment, if not the only.

Given the dynamic nature of human environments, to be capable of interacting with the elements present in them is almost a must. For example, children which have not developed fine motor skills are unable to modify their environments in a meaningful way or in a form that may help them to accomplish a goal. In contrast, a grown up person can perform actions which have a direct effect over the state of the environment, hence making these such actions more meaningful than those which do not produce any effect. The difference between them is rooted, among others, in the fact that a grown up is capable of manipulating the surrounding objects, modifying the environment for its benefit. This reason makes the grasping problem interesting and relevant. An autonomous machine which is capable of manipulating objects is better prepared to face simple, but common problems that arise in any human environment, for example, to move an object which is blocking the way or to pick up something which fell on the floor. Finally, an autonomous machine which is capable of manipulating objects is one step closer to be a machine capable of assisting people in the full range of their needs.

1.2 Goals and Objectives

1.2.1 Main Goal

The main goal of this work is to develop a data-driven grasp synthesis method (Chapter 3), that is, a method to determine a suitable way of grasping an object using a robotic hand, based on Machine Learning techniques and a novel 3D local descriptor, proposed as part of this thesis, which characterizes the vicinity of a point in terms of the curvature observed on the surface of the object underlying the sensory data.

1.2.2 Specific Objectives

The main goal of this thesis can be further specified into a set of particular objectives as follows:

Obj. I Design a 3D local descriptor, focusing on the characterization of the vicinity of a point, based on the curvature observed in the surface of an object in different directions.

Obj. II Develop the computation process for the designed 3D local descriptor, implemented as a C++ library and developed to work with sensory data structured as clouds of 3D points, also known as point-clouds.

Obj. III Determine a set of vectors for use as a reduced representation of the feature space, implicitly defined by the proposed descriptor, using an unsupervised learning algorithm and following the approach of the BoVW model (presented in Section 2.1.6).

Obj. IV Perform grasping experiments, with the goal of collecting information about the value of the proposed descriptor and robot parameterization when a successful/unsuccessful grasp is executed.

Obj. V Use the collected grasping data in a supervised learning process for training a classifier, with the goal of identifying suitable grasping zones and robot parameterizations, which would allow a robot to successfully grasp an object.

Obj. VI Assess the ability of the proposed grasp synthesis method to identify suitable grasping zones and to define proper robot parameterizations for grasping.

Obj. VII Perform the evaluation of the proposed grasp synthesis method using a simulated environment.

1.3 Hypotheses

The following list presents the hypotheses on which this work is based.

Hyp. I The curvature of an object in different directions along its surface, observed in the vicinity of a point, encodes enough information to distinguish between suitable and unsuitable zones to grasp such an object.

Hyp. II It is feasible to design and develop a 3D local descriptor which encodes information about the curvature of the object underlying the sensory data, in different directions.

Hyp. III A reduced representation of the feature space implicitly defined by the proposed descriptor can be generated using an unsupervised learning algorithm, following the approach of the BoVW model.

Hyp. IV A supervised learning algorithm can use previously collected grasping information to learn how to identify suitable grasping zones and to learn prototypical robot parameterizations.

Hyp. V The ability to identify suitable grasping zones and to learn prototypical robot parameterizations allows the system to learn prototypical grasps, which can be

used to grasp diverse types of objects.

Hyp. VI The success or failure of a grasping attempt depends on many factors, but it is strongly related to the correct positioning of the robot and the orientation used to attempt a grasp in such a way that a correct positioning greatly boosts the probability of a successful grasp.

Hyp. VII The proper orientation of a robotic hand to successfully grasp an object is defined as an orientation such that it aligns the grip of the robotic hand with the direction of higher curvature in the object.

1.4 Contributions

All the code generated as a result of this work is open-source, available under license GPLv3. The following list presents the contributions of this thesis.

Cont. I The design of the proposed 3D local descriptor.

Cont. II An implementation of the proposed descriptor, developed as a C++ library.[1]

Cont. III The experimental setup used to perform the experiments in a simulated environment.[2]

Cont. IV A grasp synthesis method developed to learn the proper way to grasp an arbitrary object, based on a training process using the trial and error approach.

Cont. V A ROS node, which provides the means to use the proposed descriptor, together with the developed grasp synthesis method in the simulated environment, as well as in a real PR2 robot.[2]

Cont. VI A set of utility applications[3] and scripts[4], that focus on helping in the preparation of data for the learning process, and on the evaluation of the performance of the trained classifier.

Cont. VII A publication containing part of the work done in this thesis, titled "Directed Curvature Histograms for Robotic Grasping" [3] and published in the proceedings of the Eurographics Workshop on 3D Object Retrieval.

1.5 Thesis Outline

Figure 1.1 presents a diagram outlining the contents of this thesis. In detail, this work is structured as follows:

- Chapter 2 presents a set of concepts and techniques which comprise the basic background required to understand the remaining chapters of this thesis.

[1] Available at `https://github.com/rodschulz/descriptor_lib`
[2] Available at `https://github.com/rodschulz/pr2_grasping`
[3] Available at `https://github.com/rodschulz/descriptor_apps`
[4] Available at `https://github.com/rodschulz/descriptor_tools`

- Chapter 3 describes the grasp synthesis problem and presents a review of the methods existing in the literature to face it.
- Chapter 4 presents a revision of the methods for 3D local description which exist in the literature and discusses their applicability to the grasp synthesis problem.
- Chapter 5 presents a novel descriptor named Directed Curvature Histograms (DCH), proposed as part of this thesis.
- Chapter 6 describes the grasp synthesis method proposed by this thesis, which makes use of the proposed descriptor, DCH.
- Chapter 7 describes the setup used to collect experimental data which was used to evaluate the proposed grasp synthesis method.
- Chapter 8 presents and discusses the results obtained from the evaluation of the proposed grasp synthesis method.
- Chapter 9 ends this thesis by drawing its conclusions and presenting possible steps to continue this research.

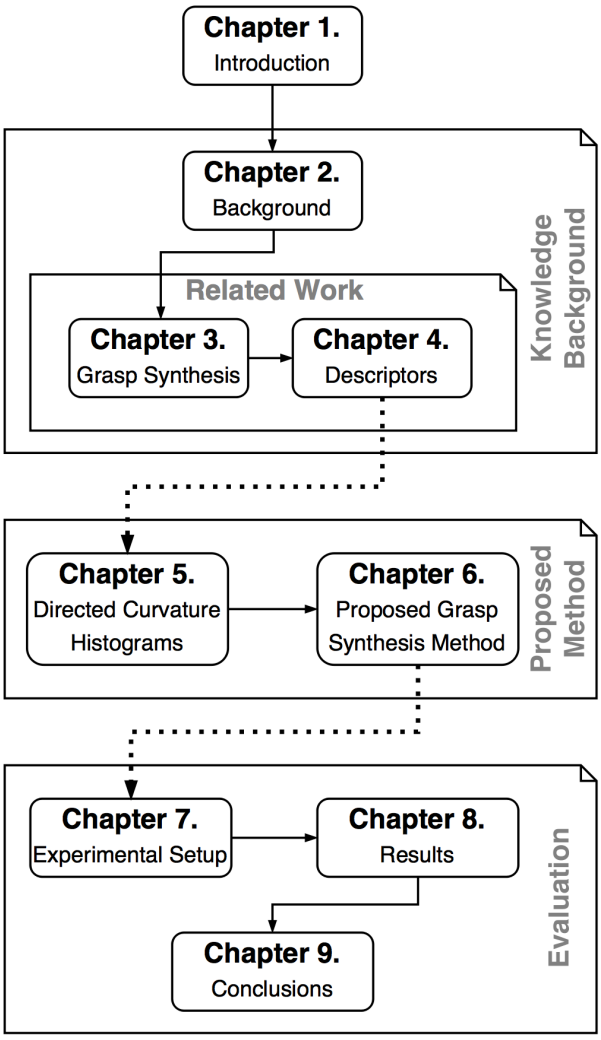


Figure 1.1: *Diagram depicting the outline of this thesis.*

Chapter 2

Background

2.1 Basic Concepts

The following section introduces a set of concepts required as background knowledge, in order to properly understand the work presented in the later sections.

2.1.1 Robotic Systems and The Grasping Problem

Sensory and motion systems are a central part of any robot. Similarly to humans, a robot processes sensory data in order to extract information that drives its behavior, decisions, and ultimately, its motion. Most robotic systems model the flow of this information by defining 5 stages, each one characterized by a different abstraction level and a particular goal. In most cases, the information flows across these stages going from the most concrete to the most abstract and back, as shown in Figure 2.1a.

In general terms, these stages can be defined as:

I **Perception:** in this stage, the sensory data is processed to retrieve relevant information, such as the presence of elements, features of the environment, etc. Tasks like segmentation and recognition are usually performed in this stage.

II **State Estimation:** in this stage, the state of the world is estimated, which typically includes the pose (position and orientation) of the robot, as well as the pose of the surrounding objects.

III **Learning:** in this stage, the knowledge collected along the execution is combined with the new sensory information to learn and update previous knowledge. Here is also where long term plans are evaluated and decided.

IV **Decision Making:** this stage controls the short-term decisions, such as in which direction the sensors should be pointed next, how should an obstacle be avoided in front of the robot, how should an object be grasped, etc.

V **Actuation:** this stage converts the short-term decisions into commands which can be sent to the motors, effectors, and sensors of the robot.

In the case of the robotic grasping problem, these stages are typically associated with 5 key tasks, respectively (Figure 2.1b):

i) Object segmentation.
ii) Feature extraction.
iii) Learning grasp strategies.
iv) Suitable grasps synthesis.
v) Effector control.

For the remaining part of this document, all the solutions and approaches to the robotic grasping problem described and discussed will be understood in the frame of the information flow presented in this section.

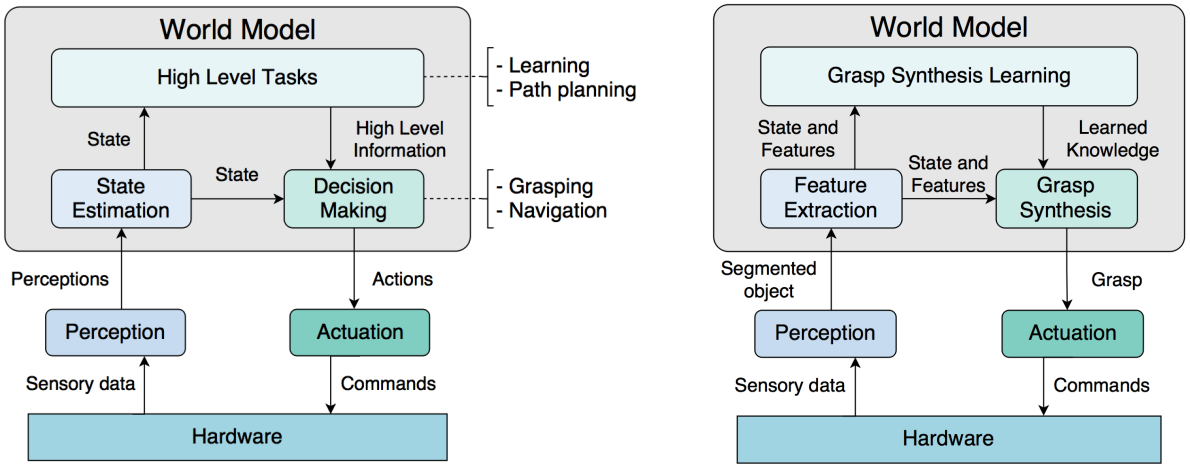
2.1.2 Machine Learning

Machine Learning is a field in Computer Science which focuses on giving computers the ability to learn without being explicitly programmed. In general, such an ability is obtained through the analysis of datasets containing information relevant to the knowledge which is to be learned. The analysis of the data allows the machine learning algorithms to extract or "learn" patterns and statistical models, which provide the knowledge obtained through the learning process.

Machine learning techniques usually can be classified into three categories:

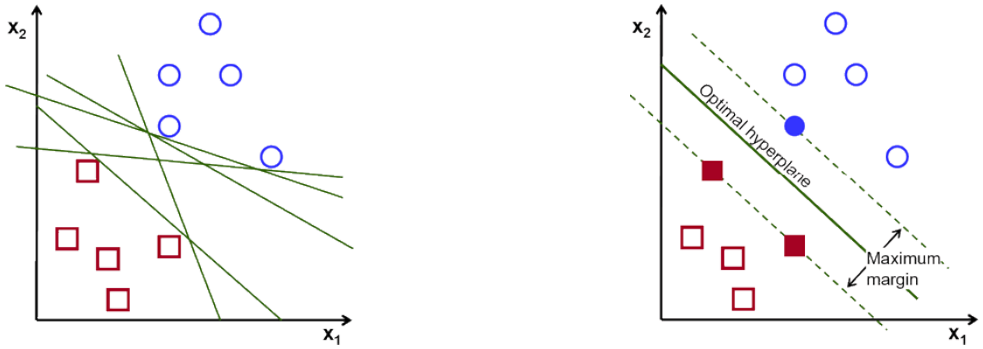
- **Supervised Learning**, also known as *classification methods*, group algorithms which learn based on a set of examples of what is to be learned. Each example includes the data used for learning (input) and its associated target value (output). These algorithms focus on finding a function (also known as *classifier*) capable of mapping the given inputs to the corresponding outputs. The output associated to each input is also known as the *class* of the item, hence the name classifier. One example of this type of algorithms is Support Vector Machine (SVM) (Section 2.1.3).
- **Unsupervised Learning**, also known as *clustering methods*, group algorithms which do not use examples to learn. The algorithms in this category focus on finding relevant information in the input data, such as patterns and non-trivial relations between data. Examples of this type of algorithms are K-Means (Section 2.1.4) and DBSCAN (Section 2.1.5).
- **Reinforcement Learning** group algorithms which focus on finding a function capable of producing a suitable output associated to a given input, following a trial and error learning process. Such process is driven by a value-function used to evaluate the outcome generated for each input data.

This thesis makes use of several machine learning techniques to identify and extract relevant information from sensory data. The details of the particular techniques used by this work are be presented in the following subsections.



(a) *General information flow in robotic systems.* **(b)** *Grasping problem information flow.*

Figure 2.1: *Information flow comparison between a general case and the grasping problem.*



(a) *Several planes exist which can effectively separate the classes.* **(b)** *Optimal plane.*

Figure 2.2: *Plane defined by SVM.*[1]

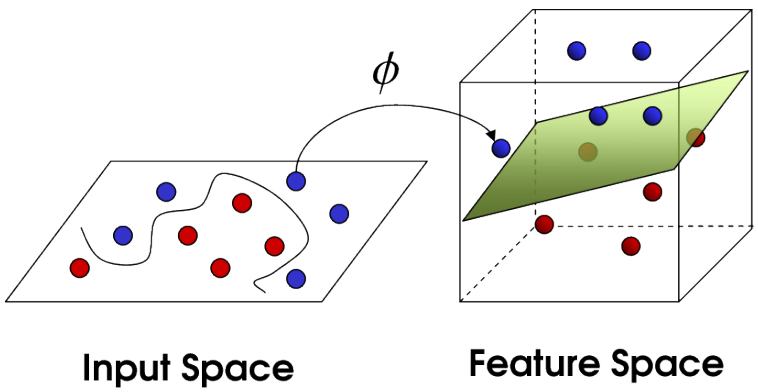


Figure 2.3: *SVM kernel trick. Function Φ transforms input data into a feature space in which it becomes linearly separable.*[2]

[1] Images from `http://docs.opencv.org/2.4/doc/tutorials/tutorials.html`.
[2] Image from "Bagging Support Vector Machines for Leukemia Classification" by Zararsiz et al. [6]

2.1.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method. The core of the SVM algorithm was originally developed by Vapnik et al. [4] in 1963. However, the current standard SVM algorithm corresponds to a variation proposed by Cortes et al. [5] in 1995. This technique is used in this thesis as a means to identify grasping zones suitable to perform a successful grasp, among all the possible grasping zones (Section 6.2.3).

This section briefly describes the motivation and main results of this technique. A detailed explanation of the SVM algorithm, its capabilities and applications can be found in *Support Vector Machines* by Steinwart et al. [7].

The core idea of SVM consists in defining a hyperplane capable of dividing the space into two regions, such that the data on each side of the hyperplane belongs to the same class. In general, for any dataset may exist many hyperplanes that might correctly classify the data (Figure 2.2a). The SVM algorithm finds the hyperplane which makes an optimal division of the space, that is an hyperplane such that the distance to the closest elements to the plane is maximal (Figure 2.2b). Such a distance is commonly denominated *margin* and the closest elements are addressed as *support vectors*.

In general terms, a hyperplane can be defined using the equation $p(x) = \beta_0 + \beta^\intercal x$, where β is known as the *weight vector* and β_0 as the bias of the hyperplane. Among the possible representations of a plane given by $p(x)$, the formulation of SVM uses the representation shown in Equation 2.1 as a convention.

$$| \beta_0 + \beta^\intercal x | = 1 \tag{2.1}$$

For any point x, the distance to a hyperplane defined by coefficients β_0 and β is given by Equation 2.2.

$$\mathrm{d} = \frac{| \beta_0 + \beta^\intercal x |}{|| \beta ||} = \frac{1}{|| \beta ||} \tag{2.2}$$

Since the margin corresponds to the distance from the hyperplane to the support vectors in both sides of the hyperplane (Figure 2.2b), then the it can be expressed as shown in Equation 2.3.

$$m = 2 \cdot \mathrm{d} = \frac{2}{|| \beta ||} \tag{2.3}$$

Finally, the optimal hyperplane is obtained solving a constraint optimization problem focused on maximizing margin, or minimizing its inverse.

$$\min_{\beta,\, \beta_0} L(\beta) = \frac{1}{2} || \beta ||^2 \qquad \text{s.t.} \quad y_\mathrm{i} (\beta_0 + \beta^\intercal x_\mathrm{i}) \geq 1 \ \ \forall \mathrm{i} \tag{2.4}$$

In Equation 2.4, x_i represents the i-th example of the dataset used in the learning process, while y_i corresponds to the class of x_i.

SVM is called a linear classifier, because the function used to define the classification frontier (an hyperplane) is linear. This necessarily imposes a restriction on the input data,

which must allow its division using a linear function, condition known as *linearly separable data*. However, despite being a linear classifier, an extension exist to successfully use SVM with nonlinearly separable data. Such an extension is typically referred to as *the kernel trick* and in its essence it consists in performing a change of variables in order to increase the dimensionality of the space, with the hope that the data will be linearly separable in such a new space (Figure 2.3).

2.1.4 K-Means

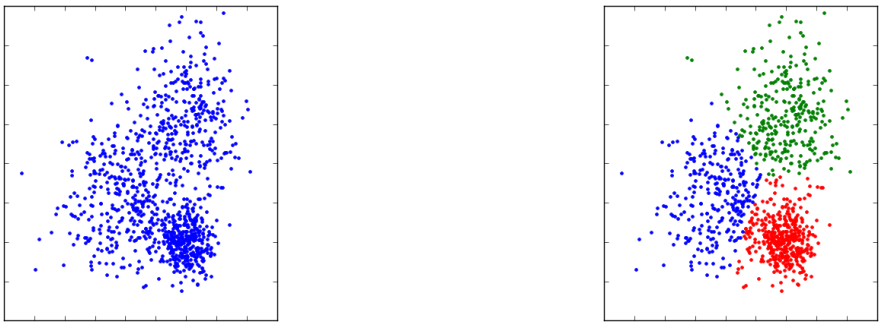
The K-Means algorithm is an iterative method for the quantization of vectors, proposed by MacQueen [8] in 1967. Despite being originally developed in the context of signal processing, this algorithm is highly popular for cluster analysis because of its simplicity and wide applicability. This algorithm is used during two processes in this work, therefore it is advisable to understand its core idea and capabilities.

K-Means is a partitional clustering algorithm, it divides a dataset into non-overlapping subsets, such that each element is exactly in one subset. This algorithm performs a distance-based clustering, grouping those points which are closer to a central point (called centroid), without taking into account the density of the data. Given the number of clusters desired, k, the algorithm first randomly determines the initial k centroids in the dataset. Then, it assigns each element to its closer centroid, and finally, updates the position of each centroid using the mean of all the points assigned to the current cluster (hence the name K-Means). This process is repeated several times until the centroids have converged to a stable set of positions, or until a maximum number of allowed iterations is reached. A pseudo code implementation of K-Means is available in Appendix A.1.

Since K-Means is a distance based clustering method, the results and behavior of the algorithm will be strongly influenced by the distance function used. In particular, the distance function will influence the shape of the clusters. For example, clusters found using a Euclidean distance will be circular-shaped, while clusters found using a Manhattan distance will be square-shaped. In general terms, L_p distances (also know as *Lebesgue distances*) will generate clusters with a globular shape or blob-like shaped. Figure 2.4 shows an example of clustering using K-Means.

Advantages

- K-Means is simple and easy to implement, which has been an important reason for its popularity.
- K-Means is fast in comparison to other clustering algorithms. Given a dataset D, the desired number of clusters k, and a maximum number of iterations allowed $maxIters$, the execution time of the algorithm will be in the worst case $\mathcal{O}(\|D\| \cdot k \cdot maxIters)$.



(a) *Unclustered data.* **(b)** *Data clustered using K-Means with $k = 3$.*

Figure 2.4: *Clustering example using K-Means with Euclidean distance.*

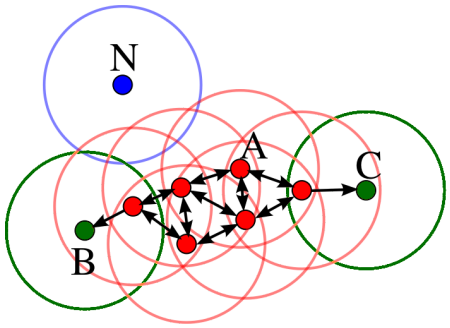
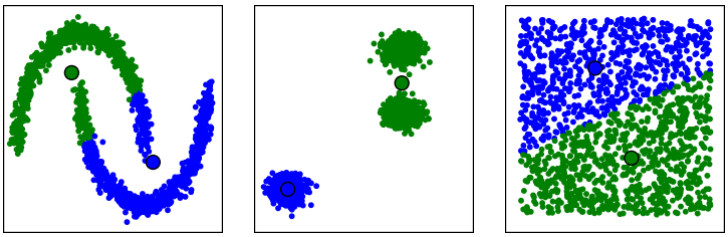
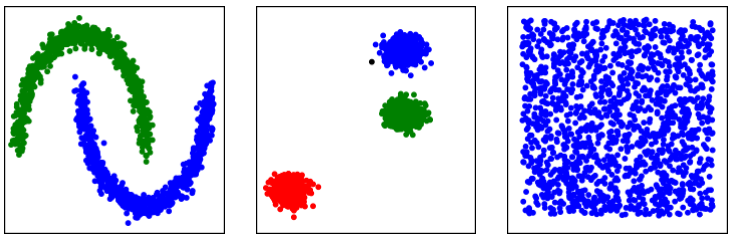


Figure 2.5: *Point types defined by DBSCAN. Core points are red, reachable points green and outliers blue.*[3]



(a) *Clusters found using K-Means on different datasets ($k = 2$).*



(b) *Clusters found using DBSCAN on different datasets.*

Figure 2.6: *Clusters found by K-Means and DBSCAN in different datasets.*[4]

[3] Image from `https://en.wikipedia.org/wiki/DBSCAN`.
[4] Images from `http://scikit-learn.org/stable/modules/clustering.html`.

Disadvantages

- The Number of clusters to be found (k) has to be determined a-priori. Depending on the context, to determine a proper value for k might be highly difficult.
- K-Means is not capable of evaluating by itself the validity of the clusters found. The algorithm can be successfully applied to data that do not present groups of any kind, therefore producing results which might be misleading if the user is not aware of the nature of the data. Even if the data is uniformly distributed across the space, the algorithm will find a set of clusters, in which case it will represent only a partition of the space.
- The centroids found by K-Means are not unique as they usually depend on the initial conditions of the algorithm, the number of the iterations evaluated and the distance function used. Such dependencies may be strong or weak, which is determined by the data.
- The convergence of the centroids to a stable position is not ensured. Depending on the data and the distance function used, the positions of the centroids may not converge, may oscillate, or may converge very slowly. K-Means ++ was proposed by Arthur et al. [9] as an algorithm for the selection of the initial positions of the centroids. It was shown that the use of K-Means ++ improves the accuracy and convergence speed of K-Means.
- K-Means can be used only when the *mean* operation is defined. For example, it might not be applicable in non-metric spaces.
- K-Means can converge to local minima, where the clusters are not a correct solution for the problem, but their position is stable.
- K-Means is sensitive to scale, therefore scaling a dataset may change the resulting clusters, if the execution parameters of the algorithm are not properly updated.
- K-Means is sensitive to noise.
- K-Means is usually not suitable to find clusters with non-convex shapes.

2.1.5 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering method proposed by Ester et al. [10]. This algorithm presents a different strategy for clustering, focusing on the density of the data instead of the distance between the elements, as presented by K-Means algorithm (Section 2.1.4). DBSCAN is applied in this thesis as a means to detect blobs of data sharing similar features.

Given a set of points, DBSCAN groups together the points which are closely packed, and marks as outliers the points located in zones of low density. To achieve this, DBSCAN defines 3 types of elements: i) core points, ii) reachable points, and iii) unreachable points or outliers (Figure 2.5). For every point, a vicinity of radius ε is defined around it. Then, core points are defined as points with at least N neighbors in its vicinity. Reachable points are defined as points located in the vicinity of a core point, but with fewer than N neighbors in its own vicinity. Unreachable points are defined as points which have fewer than N neighbors in

its vicinity and are not present in the vicinity of any core point. The DBSCAN algorithm traverses the data identifying the type of each point. Finally, the clusters are formed from connected groups of core and reachable points. A pseudo code implementation of DBSCAN is available in Appendix A.2.

Advantages

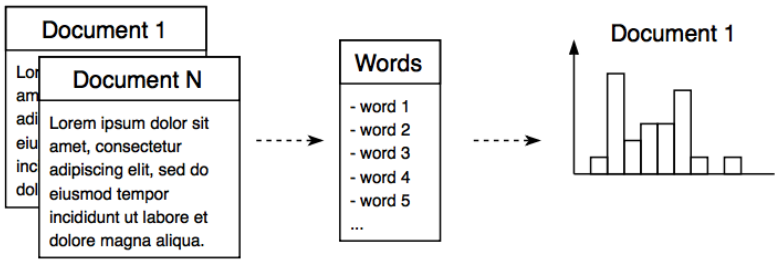
- DBSCAN does not require the definition a-priori of the number of clusters to search for. It is determined directly from the data.
- DBSCAN is robust to noise, discarding the effect of outliers in the determination of the clusters.
- DBSCAN can find clusters with an arbitrary shape, without following a particular shape pattern. It can even find clusters which are completely surrounded by (but not connected to) a different cluster. Figure 2.6b shows an example of the cluster types which can be found using DBSCAN.

Disadvantages

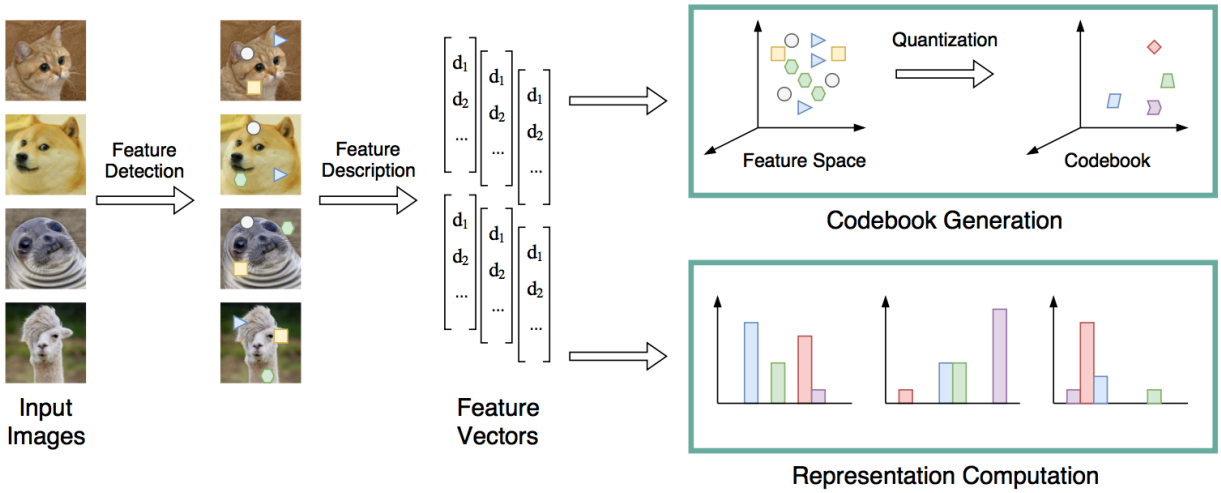
- DBSCAN requires a good understanding and knowledge of the dataset to correctly define ε and N. If the data and scale are not well understood, choosing a meaningful distance threshold ε can be particularly difficult.
- DBSCAN is not well suited to cluster datasets with large differences in density, since ε and N can not be chosen appropriately for such conditions.
- Border points that are reachable from more than one cluster can be part of either cluster. This makes DBSCAN not completely deterministic, since the clustering may change slightly depending on the order the data is processed.
- The quality of the clustering results depend on the distance function used. For high dimensional data some distance functions, such as Euclidean distance, can have poor performances (due to the *Curse of Dimensionality*). For this reason, it can be difficult to choose a meaningful value for ε.
- DBSCAN is not well suited for small datasets. In such a case, the algorithm might not be able to find any cluster if ε and N are not properly set.

2.1.6 Bag of Visual Words Model

Bag-of-Words (BoW) is a model used in Natural Language Processing and Information Retrieval to generate a simplified representation of a document or a set of documents. In this model, a text document is treated as a set (or bag) of the words composing it, disregarding grammar, order, and the relation between the words, but counting the number of times that each word is present in the document (Figure 2.7a). In the BoW model, each document is represented by a vector with the frequencies of every word, which acts as a histogram of the words contained by the document. The generation of such a vector usually involve the construction of a list of the most meaningful words (denominated *text corpus*), where connectors,



(a) *BoW model.*



(b) *BoVW model.*

Figure 2.7: *Comparison between BoW and BoVW models.*

articles, prepositions and similar words are discarded since they typically do not contribute to the core meaning of a sentence.

The Bag-of-Visual-Words (BoVW) model is an extension of the BoW model, which can be used on documents that are not text documents. This model is frequently used in Computer Vision to generate simplified feature vectors, which can be used for classification and recognition. Such is the case of this thesis, which uses the BoW to produce data which is later used for classification.

In the BoVW model, images are represented as a set (or bag) of features (or visual words), where the latter are commonly encoded using a descriptor. The construction of the simplified representation of a set of images is done following 4 consecutive steps (Figure 2.7b):

I. **Feature detection:** the relevant features of each image are identified.
II. **Feature description:** the detected features are encoded using a suitable descriptor. This transforms the features to a vector.
III. **Codebook generation:** the set of descriptors coming from all the images constitute the *feature space*. Using K-Means, the feature space is reduced to k representative vectors or *visual words*. The set of k visual words is denominated the *codebook* or *visual vocabulary* of the model.
IV. **Simplified representation:** to generate the representation, first the closest vector

in the visual vocabulary is found for every feature in the image. Then, the image is represented by a vector of the frequencies of the visual words in it.

In practice, step III (along with steps I and II) is executed once, only to produce a codebook suitable for the data. Later, during the regular usage of the BoVW model, only steps I, II and IV are executed, since a suitable codebook is already available.

Note that, although the standard formulation of the BoVW model uses K-Means to produce a codebook (step III), depending on the conditions of the problem a different clustering algorithm may be used. Such a change may produce not only a different codebook, but also could favor the presence of some features, as well as diminish the presence of others. Similarly, depending on the characteristics of the feature space, some clustering algorithms may produce a more accurate codebook than others.

One alternative algorithm is Mean-Shift [11], which has an approach similar to K-Means (successive updates of the mean of a group of points), but with the benefit that it does not require the prior definition of the number of clusters desired. Jurie et al. [12] presented a codebook generation approach using this algorithm, which tends to produce a codebook better suited than K-Means when the feature space is highly non-uniform.

Issues of the BoVW Model

The BoVW model presents some practical issues that commonly arise when it is applied to real data.

- The clustering stage performed in order to generate the codebook typically requires a high number of clusters k, even for simple problems. This usually makes the construction of the codebook a computationally expensive task.
- In most of the cases, the ability to correctly represent diverse data (for example, images showing diverse content) is directly related to the number of visual words computed in the clustering stage. This implies that a better codebook will typically take longer to compute.
- Depending on the complexity of the problem, a codebook large enough to perform well may be impossible to achieve, given the computational and time costs involved in its calculation.

2.1.7 K-Fold Cross-Validation

Cross-Validation is a technique for assessing the performance of a statistical model, which mainly evaluates if the model will be able to properly generalize to a new independent set of data. This technique is typically used in settings where the goal is to make predictions. It is used to estimate how accurately a predictive model will perform in practice. But, they are also used for the estimation of the parameters of a given model that best combines fitting the available examples and generalizing to new data. This technique is used to evaluate the

performance of the models generated as part of this thesis.

The motivation for this technique comes from the observation that, for any real system with noise and perturbations, if a statistical model perfectly fits the sample data used for its estimation, then such a model may not be able to generalize properly to a new independent sample. This effect is called *overfitting* and is produced for the presence of noise and perturbations in the sampled data, as well as for the inability of a sample to exhaustively embody the extent and complexity of the process from which it was taken.

In the literature, there are several different forms of cross-validation, for example, repeated random sub-sampling, Hold-Out, K-Fold, and leave-one-out, among others.

Hold-Out is the simplest way of cross-validation. In this technique, the dataset is divided into two sets, typically called the *training set* and the *test set*. Then, the parameters of the statistical model are estimated using the training set and the performance of the model is assessed using the test set.

K-Fold cross-validation is a technique which improves the results obtained by the Hold-Out method. In this method, the dataset is divided into K random subsets, then the Hold-Out method is repeated K times. During each repetition, one of the K subsets is used as the test set, while the union of the remaining $K-1$ subsets is used as the training set. Finally, the results of the K iterations can be combined to produce a single estimation.

The advantage of the K-Fold method resides in that all observations are used for both training and validation, which improves the efficiency of the data usage. Also, each observation is used for validation exactly once, which prevents a single observation from becoming too influential in the final model. Additionally, since the data is divided into several parts, this method reduces the importance of how these parts are generated, in opposition to the Hold-Out method, where the division of the subset strongly influences the estimated model.

K-Fold cross-validation is commonly used with $K = 10$. Also, other typical uses are a K-Fold with $K = sample\ size$, known as Leave-One-Out; or K-Fold with $K = 2$, which corresponds to the Hold-Out method.

2.1.8 ROC Curves

A Receiver Operating Characteristic Curve, also know as *ROC Curve*, is a graphical plot that illustrates the performance of a binary classifier as the threshold for discrimination of each class is changed. This curve can be extended to multi-label classification problems by means of binarization of the outputs. Along with K-Fold Cross-Validation, ROC curves are used to assess the performance of the models generated as part of this thesis.

A ROC plot features the true positive rate (TPR) of the binary classifier in the vertical axis and the false positive rate (FPR) on the horizontal axis. The ROC curve shows the relation between both, computed as the discrimination threshold is changed. The TPR is usually referred to as the sensitivity or probability of detection of the classifier, while the

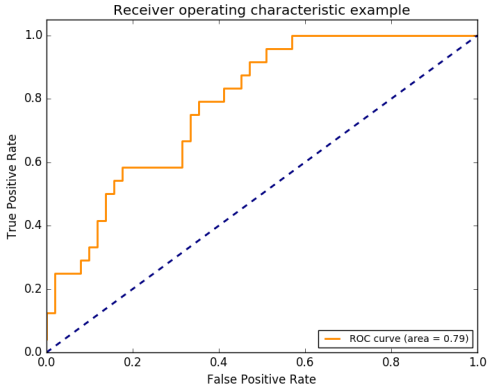


Figure 2.8: *ROC curve example*

(a) *A color image showing a kitchen scene.*

(b) *A depth image. Lighter greys represents a deeper spots.*

Figure 2.9: *Comparison of a color image and a depth image showing the same scene.*[5]

FPR is usually referred as the fall-out or probability of false alarm of the classifier. The latter is typically expressed as $(1 - specificity)$.

ROC analysis provides tools for the evaluation of the performance of binary classifiers. According to the definition of the axes, the top left corner would represent an ideal classifier (a curve describing an inverted "L" shape), capable of successfully identify both classes without producing false positives. On the other hand, a diagonal line represents the performance that would be obtained by randomly selecting the class for each element in the dataset.

The area under a ROC curve can also be used as a tool for assessing the performance of a binary classifier. Such a tool is typically called *ROC Index*. Since the ROC Curve is plotted in a unitary square, the ROC Index for an ideal classifier will be 1, while the random selection of classes will have a ROC Index of 0.5.

[5] Images from `http://rgbd-dataset.cs.washington.edu/`.

2.1.9 RGB-D Images and Point-clouds

An unavoidable task when a sensor captures data is to structure such data into a convenient format to store the sensory information, so it can be used later or can be shared. Different formats exists to store 3D sensory data, which usually differ in the nature of the data stored, as well as in the nature of the captured data.

RGB-D Images

Depth images are one of the most common formats for storing pure 3D sensory information. A depth image is equivalent to a color image where each pixel, instead of storing color information, store the depth of the scene measured starting from sensor, as shown in Figure 2.9b.

A RGB-D image is simply a color image with an additional channel describing the depth of the scene at each pixel, in consequence, a RGB-D image can be understood as a regular color image concatenated with a depth image, as shown in Figure 2.9

Point-clouds

Point-clouds are one of the most common and ubiquitous formats used to store and share 3D sensory information, given their simple definition, flexibility and powerful set of tools provided by Point Cloud Library (PCL) [13].

A point-cloud stores information as a set of independent 3D points, placed arbitrarily in space, therefore forming a "cloud" of data. Besides the spatial coordinates (x, y, z), each point in a point-cloud can hold an arbitrary number of additional fields with information regarding the point, such as the color of the scene at it, transparency level of the color, curvature, normal vector, a meaningful label, etc. These additional fields make point-clouds a flexible container to store and share 3D sensory information.

A RGB-D image can be easily transformed into a point-cloud by translating the depth of each pixel into a (x, y, z) position, using the coordinate reference frame of the sensor. Also, given the flexibility of point-clouds, each translated point can keep the color of each pixel and alpha level of the source pixel.

2.2 Software

2.2.1 ROS: The Robot Operating System

The Robot Operating System (ROS) [14] is an open-source software framework for the generation of robot software, currently developed by the Open Source Robotics Foundation

(OSRF) [15]. This framework acts as a meta-operating system for heterogeneous robot platforms, providing hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes and package management.

ROS is built as a collection of tools, libraries, and conventions which aim to simplify the development of complex robot routines, across a wide variety of robotic platforms, as well as sharing the available implementations. In addition, ROS is integrated with OpenCV [16], which provides access to computer vision algorithms; and Point Cloud Library (PCL) [13], which provides 3D point cloud manipulation and visualization.

Runtime Architecture

At runtime, ROS is structured as a peer-to-peer network of processes called ROS nodes. These nodes can be distributed across different machines, and are loosely coupled using the communication infrastructure provided by ROS. This infrastructure implements different styles of communication, including synchronous remote-procedure-call communication, asynchronous streaming of data and storage of data on a Parameter Server.

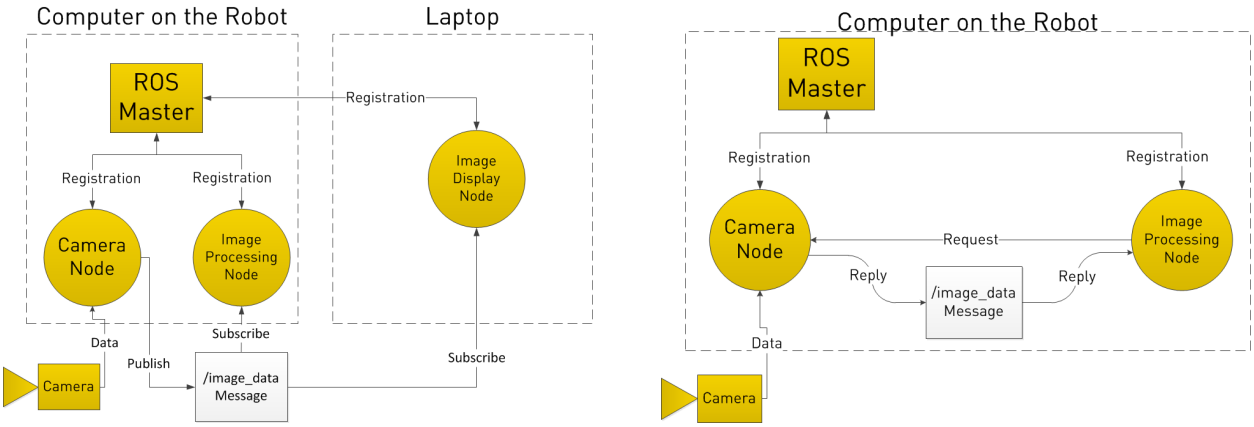
There are 6 main elements in this architecture:

- **Nodes:** are processes which perform the routines in a robotic platform. ROS is designed to be modular at a fine-grained scale; a system usually comprises many nodes.
- **Master:** provides name registration and lookup to the rest of the runtime architecture, allowing nodes to find each other, exchange messages or invoke services.
- **Parameter Server:** stores runtime parameters by key in a central location.
- **Messages:** are data structures comprising typed fields. Nodes communicate by passing messages.
- **Topics:** are the main communication method between nodes. ROS messages are routed via a transport system with publish-subscribe semantics (Figure 2.10a). A node sends a message by publishing it to a given topic. A node interested in certain data subscribes to the appropriate topic. The topic is the name used to identify the publish-subscribe interaction.

  This system allows multiple concurrent publishers and subscribers for a single topic, and multiple subscriptions and publications for a single node. This design effectively decouples the production and consumption of the information.
- **Services:** are communication methods which act as a non-exclusive alternative to topics. The publish-subscribe model provides many-to-many, one-way transport, but is not appropriate for request-reply interactions. Services provide request-reply communication between nodes (Figure 2.10b).

2.2.2 Gazebo

Gazebo [17] is a 3D dynamic simulator, developed by the Open Source Robotics Foundation (OSRF) [15]. Gazebo integrates with ROS and allows the user to accurately and efficiently



(a) *Three nodes spawned in different computers communicate through a topic.*

(b) *Two nodes communicate via remote procedure call.*

Figure 2.10: *Examples of ROS runtime architectures.*[6]

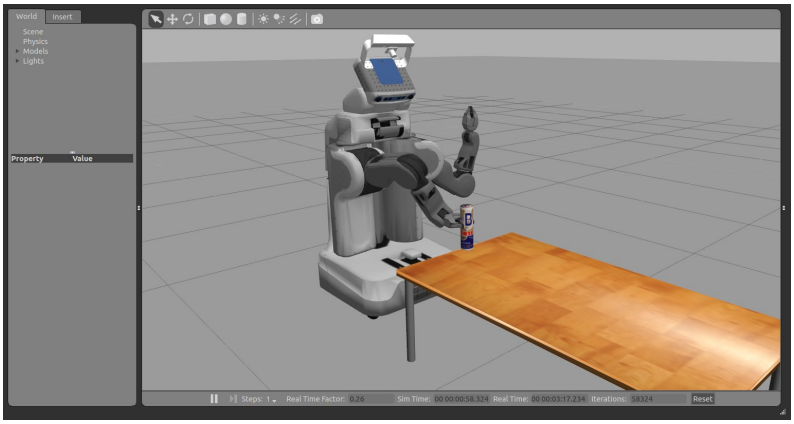


Figure 2.11: *A simulation performed with Gazebo.*

(a) *Full body PR2 robot.*

(b) *Gripper equipped on the PR2 robot.*

Figure 2.12: *The PR2 robot overview.*[7]

[6] Example runtime architecture images from `http://robohub.org/`.
[7] Images from `http://www.flickr.com/photos/willowgarage/`

simulate robots in complex indoor and outdoor environments. It offers physics simulation, simulation of sensors, and interfaces for users and programs.

Gazebo provides a list of models for several commercial robots and drones. Such models allow users to almost completely simulate different robotic platforms, including the sensory data from the sensors mounted on each robot, required hardware data (encoder positions, hardware status, etc.), physics of the environment, and, of course, the custom routines developed for the robot. Additionally, Gazebo provides a mechanism to define new models, allowing users to simulate their own robots.

In a simulation session, Gazebo spawns several ROS nodes, emulating the nodes spawned by a real robot, such as hardware control nodes or communication nodes. Simulated sensory data is generated according to the location of each sensor, geometry, and physics of the simulated environment. This data is sent to the nodes responsible for collecting it, emulating the complete sensory recollection process.

2.3 The PR2 Robot

The PR2 robot [18] (Figure 2.12a) is a humanoid manipulation platform, originally developed by Willow Garage [19], starting in the year 2010. This robot was selected as the robotic platform to be used for the development of this thesis because of its features, which makes it especially well suited to test grasping strategies. It was also selected because of its software base, which allows the user to easily test different programs, either in a real robot or in a simulation.

The PR2 robot is designed to perform in human environments and is particularly well equipped to carry out tasks involving the manipulation of objects. Its base software is completely developed using ROS, and consequently, it is fully compatible with the capabilities of ROS and Gazebo.

The hardware features of the PR2 are:

- One omnidirectional mobile base.
- Two on-board servers for sensory data recollection, communication, and execution of the programmed routines.
- Two arms, each one with 7 degrees of freedom (DoF) and equipped with a gripper with 1 DoF.
- A suite of sensors, including:
  - One Microsoft Kinect mounted on the head.
  - One 5-megapixel color camera mounted on the head.
  - One tilting laser range finder and one inertial measurement unit.
  - One laser scanner located in the base, for navigation purposes.
  - Accelerometers located on the arms and the body.

2.4 Summary

This chapter presented the concepts and methods which form the background knowledge of this work, required to successfully understand the contents of the following chapters. In particular, it is advisable to keep in mind a general image and description of the PR2 robot, since that will allow the reader to have a concrete representation of the robotic platform involved in the development of this thesis.

Chapter 3

Grasp Synthesis

Robotic grasping is a discipline in the field of Robotics focusing on the development of algorithms and methods for the definition of strategies, which allow a robot to grasp an object. León et al. [2] describe this problem as *"to determine the grasp required to carry out certain manipulation tasks on an object"*.

In general terms, a robot performs a grasp by making use of a robotic arm, together with a robotic hand, gripper or tool. These are commonly addressed as *end-effector* or, simply, *effector*. A grasp can be then defined as a particular parameterization or configuration of an effector, whose purpose is to allow a robot to hold some object to perform a certain manipulation task.

In consequence, to perform a manipulation task, a robot requires an effector and a method to define a suitable grasp, that is, a proper parameterization of the effector which allows the robot to successfully grasp the target object. Grasp synthesis is the name given to the problem of finding such a parameterization of the effector. Bohg et al. [20] refers to this as *"the problem of finding a suitable grasp configuration for a given object, that satisfies a set of criteria relevant to the grasping task"*.

The way in which the parameterization of an effector is generated to define a grasp depends on the approach used to solve the grasping problem. For some approaches a grasp is defined using a parameterization of the kinematics of the effector, that is, a definition of the required position of each joint. Some other strategies define a grasp using a high-level parameterization, where the features of the grasp are determined by information external to the effector, such as the position of the joint closer to the object, the required orientation of the effector, the position of a particular point of the effector, and so on.

In the literature, there are different ways to approach the grasp synthesis problem. Sahbani et al. [21] divided the existing methodologies into two groups: i) analytic methods; and ii) empirical or data-driven methods.

3.1 Analytic Methods

Analytic methods approach the grasp synthesis problem by focusing on modeling the physics and the dynamic behavior of the effector-object interaction. In this context, a grasp is particularly defined by León et al. [2] as *"a set of contact points on the surface of the object, whose purpose is to constrain the potential movements of the object in the event of external disturbances"*. A grasp which provides a set of contacts that allow the fingers to apply arbitrary forces and torques on the object, such that any motion is resisted, is called a *force-closure grasp* [22].

According to Shimoga [23], analytic methods focus on the generation of force-closure grasps, which are stable, in equilibrium and exhibit a certain dynamic behavior. These methods are characterized by the use of multi-fingered effectors, typically humanoid robotic hands (Figure 3.1a and Figure 3.1b). In a recent review, Bohg et al. [20] observed that analytic methods are commonly formulated as an optimization problem, constrained to the criteria of the grasping task and the dynamics of the effector.

Grasps generated using this approach are typically delivered as a set of contact points and a parameterization of the effector based on its kinematics, that is, a series of restrictions on the angles of the joints, and on the orientation and magnitude of the forces applied by each finger.

This approach was the most popular method up until the past decade. Since then, data-driven methods have been the main focus of the research community.

3.1.1 General Approach

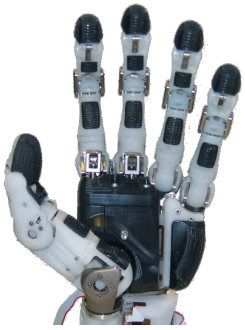
In general terms, analytic methods assume the manipulation of rigid bodies and usually dismiss the effects of friction. In this approach, a *contact point*, or simply a *contact*, is defined as the joint formed between a finger and the object, which provides an unilateral constraint that prevents the object from locally moving against the contact normal. Following, a *wrench* is defined as the force and torque applied by a finger at a contact point [2].

A wrench applied on an object is represented by a vector $w_\mathrm{i} \in \mathbb{R}^6$, as shown in Equation 3.1, where $f_\mathrm{i} \in \mathbb{R}^3$ is the force applied on the object at the contact point C_i, and $\tau_\mathrm{i} \in \mathbb{R}^3$ is the resulting moment produced at point p. Figure 3.2 presents a diagram of the contact between a finger and an object.

$$w_\mathrm{i} = \begin{pmatrix} f_\mathrm{i} \\ \tau_\mathrm{i} \end{pmatrix} = \begin{pmatrix} \widehat{n}_\mathrm{i} \\ (C_\mathrm{i} - p) \times \widehat{n}_\mathrm{i} \end{pmatrix} \tag{3.1}$$

If there are multiple contacts acting on an object, the sum of all the wrenches applied on each of them will determine the behavior of the object (Equation 3.2).

$$w_0 = \sum_{\mathrm{i}=1}^{n_c} w_\mathrm{i} \tag{3.2}$$



(a) *Humanoid robotic hand with five fingers.*[1]

(b) *Humanoid robotic hand with four fingers.*[2]

(c) *Two fingered gripper.*[3]

(d) *Three fingered gripper.*[4]

Figure 3.1: *Comparison of the different types of effectors.*

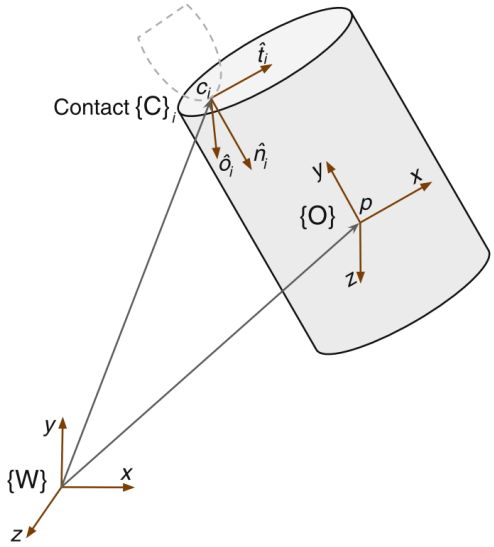


Figure 3.2: *Force diagram of a wrench applied on a contact point* [5]*. In this diagram,* $\{W\}$ *represents the coordinate system of the world,* $\{C\}$ *the coordinate system of the contact point, and* $\{O\}$ *the coordinate system of the mass center of the object.*

[1] Elumotion EH2 hand, image from `http://www.elumotion.com/`.
[2] Allegro hand, image from `http://robots.ros.org/allegro-hand/`.
[3] PR2 gripper `http://www.flickr.com/photos/willowgarage/`
[4] 3-Finger adaptive gripper `http://robotiq.com/`
[5] Image from "Robot Grasping Foundations" by León et al. [2].

Finally, to achieve a force-closure grasp, the sum of wrenches applied over the object and the external disturbances have to be in equilibrium (Equation 3.3).

$$w_0 + w_{ext} = 0 \tag{3.3}$$

Diverse methods can be applied to solve Equation 3.3, but it is commonly solved as a constraint optimization problem, as mentioned before. Equations 3.1 and 3.2 can be expanded to consider additional effects, such as friction, the stiffness of the surface, errors in the positioning of the effector and so on.

3.2 Data-Driven Methods

In contrast to analytic methods, which determine a suitable grasp according to the physics of the problem, empirical or data-driven methods rely on the use of sensory information to sample grasp candidates from the solution space. An appropriate grasp is determined from such candidates after assessing them based on previous grasping experiences, grasp prototypes and/or grasp examples provided by human experience. For this reason, this approach is sometimes also dubbed *knowledge-based* [23].

Data-driven methods typically use effectors that are simpler than the ones used by analytic methods (Figure 3.1c and Figure 3.1d). In general terms, the grasps generated with this approach are parameterized using high-level information instead of parameters relative to the kinematics of the effector. Such parameterization commonly includes one of more or the following points [24]:

- A grasping point on the surface of the object, which is typically used to position some part of the effector with the object, for example, the center point of the effector.
- An approximation vector, describing the 3D angle that the effector has to use to approach the grasping point.
- The orientation of the wrist, which defines the orientation used by the effector to approach the grasping point.
- An initial finger configuration.

Data-driven methods have been the most popular approach over the past decade. This is partly explained by the availability of simulators like GraspIt! [25], which allowed researchers to test data-driven strategies with a reduced setup time and under controlled conditions.

3.2.1 General Approach

To outline a general approach for data-driven methods it is harder than for analytic methods since there is a wider range of methods proposed by the research community. According to Sahbani et al. [21], empirical grasping approaches refer to techniques based on classification and learning methods that avoid the computational complexity of the analytical ones.

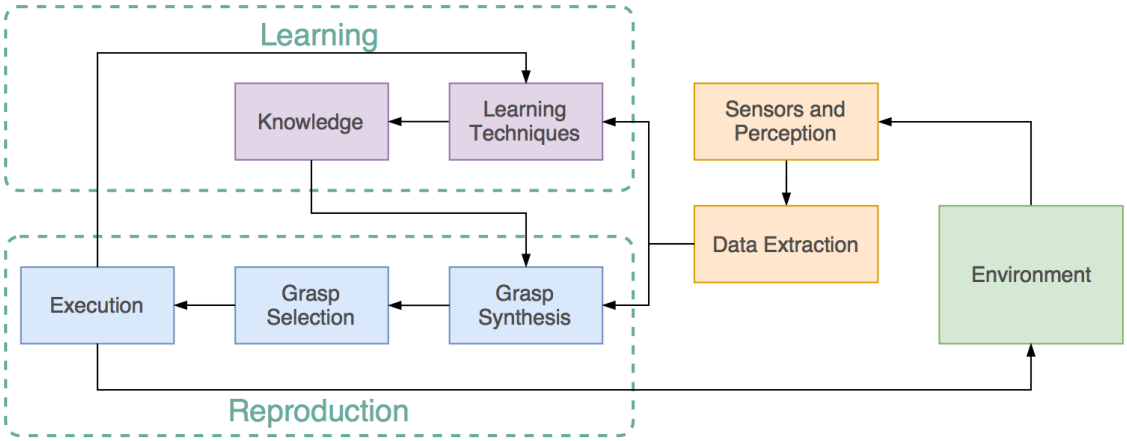


Figure 3.3: *General workflow of a data-driven grasp synthesis method.*

These methods typically involve at some point the use of descriptors to collect and extract relevant information. Also, they commonly make use of multiple techniques to analyze the collected data, ranging from basic data filtering to complex techniques such as supervised and/or unsupervised learning. In general terms, data-driven techniques involve three steps:

i) **Sensory data recollection**, where the environment is sensed and base data is collected.

ii) **Relevant data extraction**, where the sensory data is processed and relevant information is extracted according to the specific approach followed by the algorithm in use.

iii) **Learning and data analysis**, where the extracted relevant data is used to learn the features that identify a suitable grasp in diverse circumstances.

Figure 3.3 shows a diagram of a general workflow of a data-driven method.

3.2.2 Classification of Methods

In a recent review, Bohg et al. [20] divided the data-driven approaches based on what they assume to know a-priori about the object. According to that, they proposed to classify the existing methods in three groups: i) grasp of known objects, ii) grasp of familiar objects, and iii) grasp of unknown objects.

Grasp of Known Objects

The grasp-of-known-objects approach assumes that the target object has been previously manipulated and, in consequence, that at least one grasp has been already produced for it. This approach typically involves the use of a database with geometric models of several objects, along with suitable grasps for each of them.

These methods usually focus on recognizing the target object and estimating its pose, so

a proper grasp can be retrieved from the database and applied according to the pose of the object. Because of that, the main challenge in this approach is to build a database with enough meaningful grasping experience and models such that most real life scenarios can be resolved.

Nowadays, there are several strategies for the construction of the database collecting the grasping experience and models. One common strategy proposed for this is to approximate the shape of the object with shape primitives (cones, cylinders, etc.), which are used to limit the solution space of possible grasps [26, 27].

Other approaches for the generation of a database of grasps have focused on the notion of learning by example. Ciocarlie et al. [28] proposed a method which exploited studies showing that humans use a limited range of movements to grasp an object, and consequently the space of possible grasps could be considerably reduced to a small fraction of prototypical grasps. Detry et al. [29] proposed a method which used contour descriptors to produce a general view of the shape of the object, which was later combined with human grasp examples to generate a database of possible grasps. In a similar way, Kroemer et al. [30] proposed to combine contour descriptors with reinforcement learning algorithms, in order to produce grasp hypotheses automatically.

Finally, a third approach suggested constructing a database of grasps following a simple process of trial and error. Some methods using this approach combined it with reinforcement learning algorithms [31], while others used the trial and error information to produce object-specific zones with a high density of successful grasp attempts [32], which were later used to grasp the object.

Grasp of Familiar Objects

The grasp-of-familiar-objects approach assumes that the target object is similar to previously manipulated objects, where experience from previous grasps can thus be reused. This approach assumes that similar objects can be grouped into categories with similar features that can be grasped in similar ways. The similarity between objects can be evaluated using low-level features, such as color, shape or texture; as well as in high-level features, such as object type or category.

The main challenge for this approach is to find an object representation and a similarity metric that enables effectively using and transferring the grasping experience. In the literature it is possible to identify four common approaches to this problem.

The first approach attempts to learn a discriminative function in order to distinguish between good and bad grasp candidates. These methods typically differ in the features used and, consequently, in the feature space over which the objects are considered to be similar. For example, El-Khoury et al. [33] proposed a strategy which divided the object point-cloud into parts and then identified similarities using neural networks.

The second approach is centered on using examples of grasps. In this case, the grasps are synthesized for a particular object by finding the most similar object or object part in a

database for which exists a set of good grasps available. For example, some methods using this logic produce the grasp candidates using synthetic models of the objects and models of the grasps based on human examples [34]. In contrast, other methods in this category use sensor-based examples [35].

The third approach is sometimes described as *learning generative models* for the grasping process. The methods following this strategy attempt to identify common structures and stages in the grasp procedure based on multiple examples. Montesano et al. [36] proposed a method focused on identifying which actions were associated with positive outcomes (successful grasps), with the goal of reproducing such actions in order to generate a suitable grasp.

Finally, the fourth approach is centered on using high-level information to define a grasp candidate for each object. In contrast to the previous methods which linked basic information (geometry, actions, similarity, etc.) with the grasps, these approaches focus on using information such as the purpose of the object or the environment where it is stored, to influence the determination of a proper grasp [37] [38].

Grasp of Unknown Objects

The grasp-of-unknown-objects approach does not assume any kind of previous knowledge of the target object or any kind of grasping experience in relation with it. This approach focuses on identifying structural and geometric features in the sensory data, to use them to identify suitable zones for grasping and to synthesize a proper grasp.

The main challenge in this case is to identify the geometric features that can predict suitable grasping zones and produce good grasps. In the literature, it is possible to find three main focuses to face this problem. The first approach is centered on approximating the target object using shape primitives, which then act as the representation of the geometric features of the object, and are used to generate grasps accordingly [39] [40].

The second approach focuses on producing a mapping between low-level geometric or visual features, and a predefined set of grasp postures. The goal of this method is to generate grasp candidates based on the recognition of such features [41] [42].

Finally, the third approach focuses on using global characteristics of the shape of the object to produce the grasp candidates. Some methods following this approach use features such as the 2D silhouette of the object to predict a suitable grasp [43]; while others use a segmented point-cloud to generate the grasps based on the shape and the predicted mass center of the object [44].

3.3 Analytic vs. Data-Driven Methods

The differences between analytic and data-driven methods are obvious. Analytic methods focus on the physics of the problem, while data-driven methods focus on the use of high-level information.

Analytic methods place more importance on modeling the interactions between the effector and the object. This yield grasps which ensure their quality in terms of equilibrium, dexterity and dynamic stability [23]. Also, analytic methods typically can be verified and validated by analyzing the kinematics and the equations which model the problem. This represents an advantage since it allows researchers to validate the generated grasps without going through extensive testing. However, these methods usually involve a complex formulation, which might be hard to develop properly and even harder to solve.

On the other hand, data-driven methods place more importance on the representation of the object and the retrieval of perceptual information. Tasks such as feature extraction, object recognition, pose estimation and so on, become key since the information obtained from them directly drives the grasp synthesis. The parameterization in these methods is less specific, which makes them better suited for the uncertainties in the perceptual information and especially for the uncertainties and errors in the movements of the effector. Data-driven methods are easier to develop and implement, typically requiring less time for modeling the problem and more time for data processing. However, the set of grasps synthesized using this approach typically do not offer any kind of guarantee about the equilibrium, stability or dynamic properties of the grasped object. Additionally, the generated grasps can only be empirically validated, consequently requiring more time invested in testing.

In practice, data-driven approaches have been more popular in the later years. This can be explained by the availability of better sensors, increased computing power, and the extensive development of data-mining and machine learning methods and new applications, which all stimulate the use of such techniques in diverse fields.

3.4 Summary

This chapter presented a formal definition of the Grasp Synthesis Problem, introducing several concepts specific to this context. The existing methods to resolve such problem can be divided into two groups: i) analytic methods, which focus on modeling the physics involved in a grasp and producing a dynamically stable grasp candidate, and ii) data-driven methods, which focus on using sensory data to *learn* the features which identify a convenient zone to grasp an object. Methods in the first group are usually considered harder to resolve, while the methods in the second are considered technically easier to face, but consume more time and data to test their validity.

Chapter 4

3D Descriptors

A descriptor is a mathematical abstraction, that focuses on encoding a part, or the whole, of the information present in a set of data, and producing a vector as a representation of such information. Such a vector is typically referred to as *the descriptor*; what information is encoded in it and which are its features depend on the algorithm used to compute it.

Commonly, the goal of using a descriptor is to transform an unstructured problem (object recognition, voice identification, etc.), to a mathematical problem, where diverse tools and algorithms can be used to approach and solve such a problem.

In general terms, descriptors are divided into two types: i) global descriptors, and ii) local descriptors. Global descriptors aim to encode the information present in the whole set of sensory data. This usually makes them better suited for tasks like recognition and identification of global features. In contrast, local descriptors encode the information present in the sensory data, limited to the vicinity of a given point (typically called *target point*). This usually makes local descriptors better for the detection or recognition of relevant features in the data, as well as the detection of interesting zones.

Additionally, a descriptor might present a series of features that, depending on the problem, may help identify relevant information or ease the recognition of similar instances of the same descriptor. Typical features that might be found in descriptors are invariant to scale and/or rotation changes; in the case of visual data typical features might be invariant to color or illumination changes, etc.

4.1 Descriptors for 3D Sensory Data

A particular case of descriptors is 3D descriptors. A 3D descriptor is a descriptor developed to be computed using three-dimensional sensory data, such as point-clouds, meshes or surfels. Depending on the algorithm which computes the descriptor, a particular format for the sensory data might be required.

In recent years, these descriptors have experienced a rise in popularity, increasing their usage and boosting their development. This has been mainly fueled by the availability of low-cost 3D sensors (started and led by *Microsoft Kinect* in 2012 [45]), which made the generation and capture of 3D sensory information accessible and easy. Also, the introduction of libraries for the manipulation and analysis of 3D sensory data, like Point Cloud Library (PCL) [13], have boosted the use of 3D data.

For the purpose of this thesis, 3D local descriptors are of particular interest, since they can provide useful information for the identification of suitable grasping points and zones. Tombari et al. [46] suggested a categorization of the current approaches in the field, based on the underlying method used to represent the 3D information. This categorization divided descriptors into three classes: i) signature-based descriptors, ii) histogram-based descriptors, and iii) mixed methods.

4.1.1 Signature-Based Descriptors

In broad terms, signature-based descriptors characterize the surface in the neighborhood of the target point (typically named *support*) by encoding one or several geometric features of each point present in the support. Such geometric data is typically computed by using a Local Reference Frame (LRF). The descriptors in this category are typically highly descriptive since they exploit the spatial location of each point to generate a spatial characterization of the 3D surface. However, this makes them more sensitive to small variations in data, hence less robust to noise.

This category includes some of the earliest approaches for 3D descriptors, and over the last few years it has been left behind in favor of histogram-based methods and particularly in favor of mixed methods. For such reason, this approach will not be further detailed.

Some examples of methods in this category are Point Signatures [47], 3D Point Fingerprint [48] and Exponential Mapping [49].

4.1.2 Histogram-Based Descriptors

Generally speaking, histogram-based descriptors characterize the support by producing one or more histograms, where local geometrical or topological features are accumulated, for example, point counts, vector angles, mesh triangle areas, etc. Usually, the generation of such histograms requires the definition of either a Reference Axis (RA) or a Local Reference Frame (LRF), used to define the quantization of the domain which will be presented by the histogram.

In contrast to signature-based descriptors, this approach tends to be less descriptive, given the accumulation of features, which reduces the sensitivity. Nevertheless, this also makes histogram-based descriptors more robust to noise and usually better suited for the description of the macro features.

Some popular 3D local descriptors which fit this category are Spin Images [50], 3D Shape Context [51] and Unique Shape Context [52].

Spin Images

Spin Images (SI), proposed by Johnson [50], is one of the oldest approaches in local description of 3D sensory data. It was originally developed to work with meshes, but was later adapted for work with point-clouds.

Spin Images uses a cylinder as a support structure (the shape of the support). The computation process of the descriptor starts by defining a plane tangent to the surface at the target point. The data present in the support is represented using cylindrical coordinates (ρ, θ, h), centered at the target point. This data is then projected into the plane $\rho - h$, which generates a 2D version of the object, dubbed *Spin Map*. In this context, the axes ρ and h are renamed to α and β, respectively (Figure 4.1a). The generation of the spin map "compresses" the angular dimension into a plane, resulting in an effect as if the data would have been spun around the normal vector of the plane.

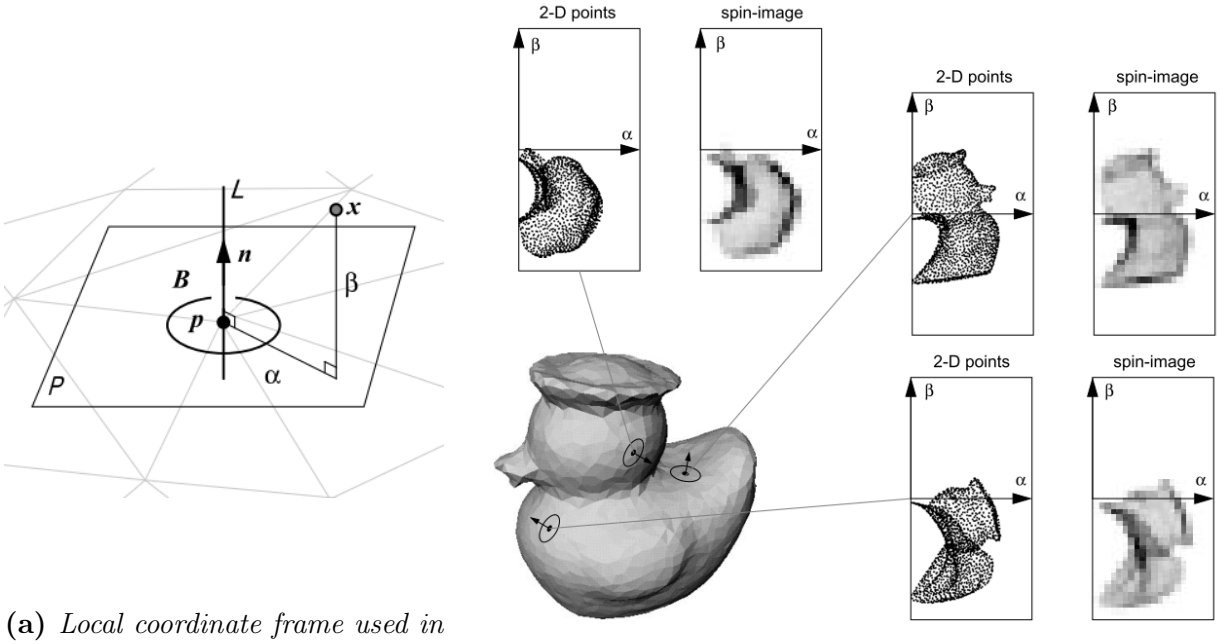
Finally, the data projected into the plane $\alpha - \beta$ is binned (like a 2D histogram) and transformed into a gray-scale image, where the darker areas denote zones of higher concentration of points (Figure 4.1b).

3D Shape Context

A common strategy for the description of spatial information is to use the spatial distribution of the data, represented through different histograms, as a feature which can be exploited. This idea was initially proposed by Belongie et al. [53], with the development of a 2D descriptor known as *Shape Context*. This approach was later extended to 3D sensory data by Frome et al. [51], with the proposal of *3D Shape Context*.

3D Shape Context (3DSC) uses a sphere centered at the target point as a support region. This sphere is oriented by aligning its north pole with the surface normal at the target point, and then, it is divided into L equally sized parts along the azimuth axis, K equally sized divisions along the elevation axis, and R logarithmically sized parts along the radial axis (Figure 4.2). The logarithmic divisions make the descriptor robust to distortions in the shape with the distance from the target point. This produces a total of $L \cdot K \cdot R$ spatial bins, where the contents of each bin correspond to one element in the feature vector.

Since the orientation of the sphere in the azimuth axis is not fixed, the spatial binning is produced multiple times in order to allow the comparison with descriptors computed in different coordinate systems. This is done by rotating the support region around the north pole axis in L positions, producing a final descriptor of size $L^2 \cdot K \cdot R$. Finally, the feature vector is generated using the count of points per spatial bin, weighted by the size of the bin.



(a) *Local coordinate frame used in the computation of the descriptor.*

(b) *Examples of the computation of Spin Images.*

Figure 4.1: *Spin Images computation.*[1]

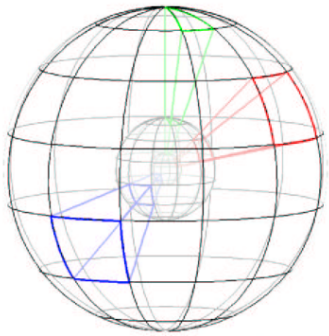


Figure 4.2: *Support structure used for the computation of 3DSC.*[2]

Unique Shape Context

Unique Shape Context (USC) was proposed by Tombari et al. [52] as a modified version of 3DSC. It shares the same support and formation with 3DSC, but includes the definition of a local reference frame used to set the orientation of each point. The advantage of using a local fixed frame lies in that the descriptor no longer needs to compute the results of the binning stage for several rotations, since a stable and repeatable orientation is available. This reduces the size of the descriptor and its computation time.

[1] Image from "Spin-images: a representation for 3-D surface matching" by Johnson [50].
[2] Image from "Matching with shape contexts" by Belongie et al. [53].

4.1.3 Mixed Descriptors

Mixed descriptors comprise approaches which mix signature-based and histogram-based approaches in an attempt to produce highly descriptive methods, which are robust to noise and distortions. Some examples following this approach are Signatures of Histograms of Orientations [46] and Point Feature Histograms [54], among others.

Signatures of Histograms of Orientations

Signatures of Histograms of Orientations (SHOT), proposed by Tombari et al. [46], uses the robustness of histograms by encoding basic information of the data inside the support, mixed with the discriminative power of signatures by generating such histograms in different specific locations.

This descriptor uses a sphere centered at the target point to define its support region. The computation process starts by splitting the sphere into 32 space regions, produced by the division of the azimuth axis in eight equally sized parts, and the elevation and radial axes in 2 equally sized parts, respectively (Figure 4.3). The support region is oriented in a unique and repetitive way by using a local reference frame, which ensures the rotation invariance of the final descriptor.

The normals of the points in the support are then encoded using one histogram per each of the 32 regions previously defined. This gives the descriptor the robustness provided by the filtering effect of the histograms, and also since each histogram is computed for a particular spatial zone in the support, it boosts the discriminative power of the descriptor by mimicking a signature. Finally, the descriptor is generated by the concatenation of the histograms computed for each spatial region defined in the support.

Point Feature Histograms

Point Feature Histograms (PFH) was proposed by Rusu et al. [54] in an attempt to capture and encode the information regarding the geometry of the surface surrounding the target point. In broad terms, this is done by capturing the variations in the orientation of the surface normal, according to a local fixed frame. The goal of this descriptor is to encode a generalized notion of surface curvature surrounding the target point.

The support region Point Feature Histograms (PFH) is defined by a sphere centered at the target point. In this case, the support region is not parametrically defined, instead, it is implicitly defined by the selection of the K nearest neighbors of the target point.

The computation process of the descriptor starts by defining a local fixed frame (known as the Darboux frame) per each pair of points present in the support. This frame is used to compute the signature associated with each point. Figure 4.4a shows the coordinate frame defined between the target point, p_t, and a point in the support, p_s, using their respective normal vectors n_t and n_s. Using this definition, the change in the orientation of the normal

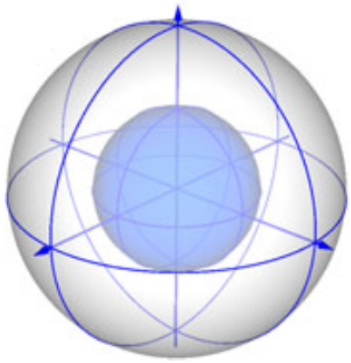
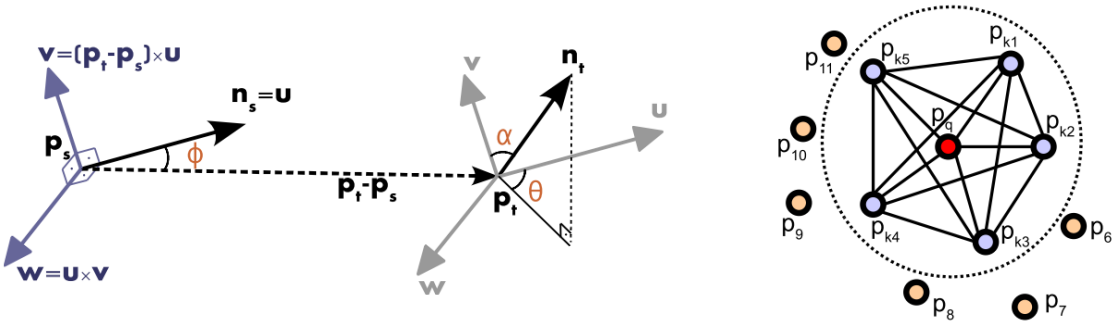


Figure 4.3: *Support structure used for the computation of SHOT.*[3]



(a) *Local fixed frame definition.*

(b) *Point influence diagram for the computation of PFH.*

Figure 4.4: *PFH computation.*[4]

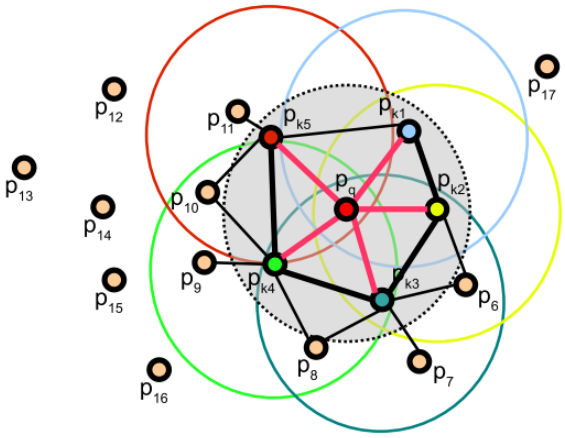


Figure 4.5: *Point influence diagram for the computation of FPFH.*[5]

[3] Image from "Unique signatures of histograms for local surface description" by Tombari et al. [46].
[4] Images from "Persistent point feature histograms for 3D point clouds" by Rusu et al. [54].
[5] Image from "Fast Point Feature Histograms (FPFH) for 3D registration" by Rusu et al. [55].

vector is measured using the angles ϕ, α and θ (Figure 4.4a). These angles, plus the distance between the points, are measured for all possible the combinations of pairs of points present in the support (Figure 4.4b). Finally, the descriptor is formed by the accumulation of such variables in four histograms. This last step provides the descriptor with robustness to noise and distortions.

Since PFH is computationally expensive to calculate, Rusu et al. proposed a lighter version of this descriptor, named Fast Point Feature Histograms (FPFH) [55]. The only difference between both is that FPFH only considers the pairs of points formed by the target point and the points in the support (Figure 4.5).

4.2 Applicability to Robotic Grasping

The descriptors mentioned in the previous section focus on capturing different features from the surface present in the support. However, when applied to the grasping task, they turn out to be useless for deciding suitable grasps for the target object (discussed in Section 8.1.2).

There are different reasons why these descriptors do not help in the grasping task, as would have been expected. Spin Images provide a rough representation of the shape of the object from the perspective of the target point, but the nature of this descriptor (an image) makes it difficult to identify and compare different zones according to their curvature.

The approaches displayed by 3DSC and SHOT prove useful for object recognition and shape retrieval. Generally speaking, spatial description is usually well suited for the identification of traits and features related to the shape of the object. However, when applied to the grasping tasks they both appear to underperform. One important drawback of 3DSC is the high dimensionality of the descriptor, which has a strong negative effect in the performance of algorithms and tasks. This issue is particularly sensitive for data-driven grasp synthesis methods, where the dimensionality of data can have a dramatic effect on the performance of the algorithms.

PFH and FPFH are specially designed to reflect the curvature of the object, and also encode as much information as possible about the surface of the object present in the support. The main disadvantage of these descriptors lies in their highly complex and computationally expensive nature. This issue remains a specially problematic matter in the context of robotics, where computing power tends to be a common bottleneck for demanding tasks, while real-time actions remain a must.

4.3 Summary

This chapter presented a review of the methods for 3D local description existing in the literature. These approaches can be grouped into three categories: i) signature-based, which are more descriptive, but sensitive to noise; ii) histogram-based, which are robust to noise, but

show less predictive power; and iii) mixed methods, which attempt to harvest the descriptive power of signatures and at the same time achieve noise robustness.

Although these methods perform well in problems such as shape retrieval and object recognition, they perform poorly when used in the context of the grasp synthesis problem.

Chapter 5

Directed Curvature Histograms

This thesis proposes a novel 3D local descriptor, dubbed Directed Curvature Histograms (DCH), designed for use in the context of the robotic grasping problem. This descriptor focuses on encoding the changes in the curvature of a surface along different directions, in the vicinity of a given point.

The proposed descriptor is designed as a set of histograms, where each histogram reflects the curvature around the target point in a particular direction. It was implemented to work with sensory data structured as clouds of 3D points (or point-clouds), but it can be easily adapted to work with different types of sensory information, such as meshes or surfels.

5.1 Premise and Design

The design of DCH relies on the premise that the local curvature of an object, analyzed in different directions, is one of the most relevant features which can be used to identify a suitable way to grasp an object. This proposition is based on a study of the grasping behavior exhibited by humans, done by Feix et al. [1], which showed that:

i) Humans tend to repeat a small set of grasps to handle different objects (5 grasp prototypes accounted for 70% - 80% of the studied cases).

ii) There is a strong relation between the perceived form or type of the object and the grasp performed.

iii) Humans commonly grasp an object by its smallest dimension.

The first observation shows that to produce suitable grasp strategies for an arbitrary object, it is only required to learn a small set of prototypical grasps. Meanwhile, observations ii) and iii) indicate that to generate good grasping strategies it is only needed to identify a general shape, in order to use the most likely graspable zone, represented by the smallest dimension.

In this case, the recognition of such dimension can be done by using the curvature of the

object, assuming that the smallest dimension corresponds to the zone of highest curvature in the general shape of the object. Therefore, the third conclusion can be interpreted as claiming that humans perform the selection of grasp strategies guided by the curvature of the target object. Formally, if a coordinate system is defined on the effector, as shown in Figure 5.1, then such conclusion can be rephrased by stating that humans favor grasps in which the vertical axis of the effector, Z_{eff}, is aligned with the axis of lower curvature of the object and perpendicular to the axis of higher curvature (Figure 5.2).

For practical purposes, the goal of defining a descriptor like DCH is to use it in the context of robotic grasping, in particular, to solve the grasping problem following a data-driven approach (presented in Section 3.2). Consequently, such a descriptor must be able to extract information from the sensory data, which is useful for the definition of a suitable grasping strategy. Additionally, it would be expected for DCH to follow the premise previously discussed. Therefore, in order to be helpful, DCH has to exhibit some particular features:

- Since the descriptor is expected to help in the identification of zones suitable for grasping an object, DCH should encode local features (as opposed to global features).
- DCH has to reflect the local curvature of an object and its evolution along different directions since such information is key to deciding a grasping strategy.
- The computation of DCH has to be fast since the robotic grasping problem usually presents time constraints which must be met to perform properly.

5.1.1 Design

The design of DCH seeks to fulfill the features identified in Section 5.1. For such a reason DCH is designed as a local descriptor, that is, a descriptor which encodes the information present in the vicinity of a given point, usually known as *target point*.

The design of DCH uses the evolution of the orientation of the vector normal to the surface, in order to reflect the curvature of the sensed object. More specifically, the curvature between two points is represented as the angle formed by the surface normal on both points. This representation has three advantages over the direct use of the curvature of the surface:

i) The angle between two vectors is typically easier to calculate and less prone to estimation errors than the curvature of the surface.
ii) Angles are limited in range ($\alpha \in [-\pi, \pi]$), in contrast to curvature, which has infinite range ($\kappa \in [0, \infty]$).
iii) Special cases (a flat surface, for example) are easier to represent and identify using angles than using the curvature.

In order to reflect the curvature of the object in different directions, DCH extracts information from a set of N radial bands of sensory data placed symmetrically around the target point. This allows DCH to encode the local curvature of the surface in the vicinity of the target point, following a series of different orientations (Figure 5.3).

In practice, the computation process of DCH measures the curvature of the object as the

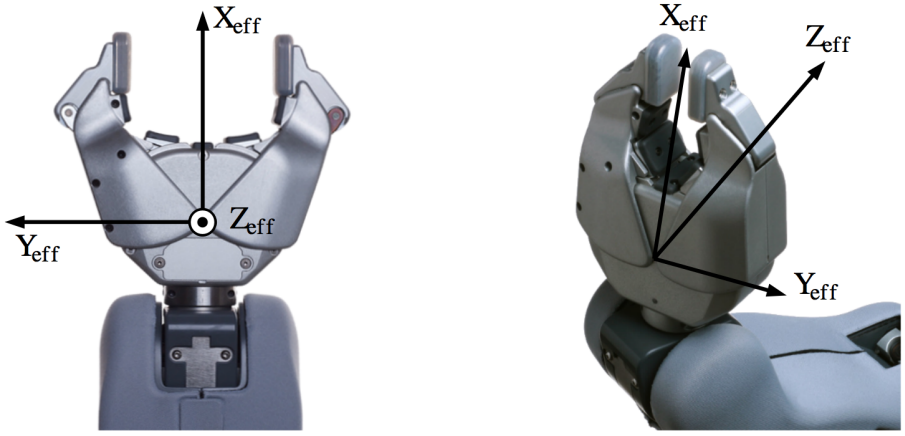


Figure 5.1: *Coordinate system of the effector.*

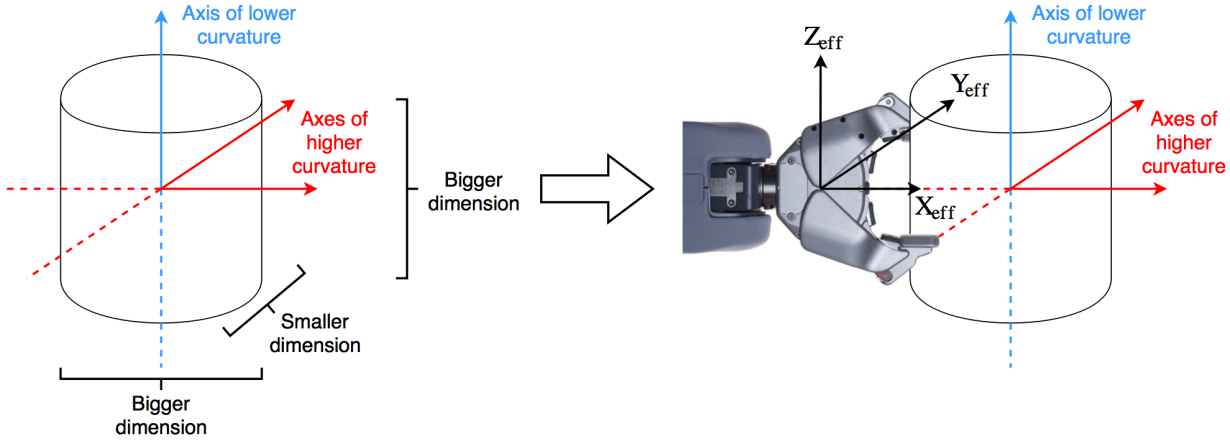


Figure 5.2: *Alignment of the grasp according to the axis of lower curvature. This strategy mocks the decision process of humans, where objects are grasped by their smaller dimension.*

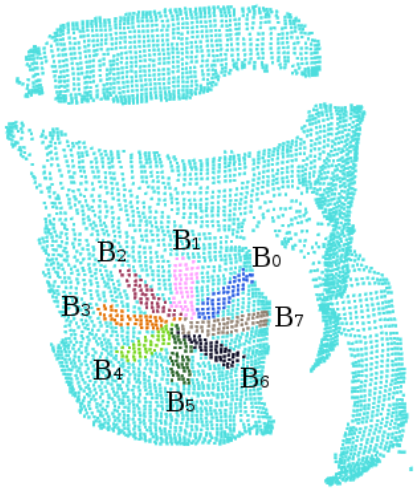


Figure 5.3: *Set of 8 bands highlighted on 3D sensory data. Each band is used to provide information about the curvature of the object along a different direction.*

angle between the target point and the points of each band, consequently describing how the curvature changes around the target point and going along the surface in different directions.

5.2 Computation

As previously mentioned, this definition assumes that the sensory data is structured as a 3D point-cloud. Nevertheless, all the following definitions can be easily applied to different sensory data representations.

The computation process is divided into two consecutive steps:

Step I. **Bands Extraction**, whose goal is to generate several sets of 3D points, each one defining a band of sensory data. Every band will then be used to get a representation of the evolution of the curvature in a particular orientation.

Step II. **Descriptor Computation**, whose goal is to compute the descriptor vector itself, encoding the information about the evolution of the surface normal.

Bands Extraction

The goal of this step is to extract N sets of 3D points, each one defining one band of sensory data in the vicinity of the target point $\vec{p}_*$ (Figure 5.3). For this purpose, a sphere of radius r is defined centered at the target point, which generates the neighborhood set Φ. The radius of this sphere determines the extent of the encoded curvature information.

In order to determine which points are part of each band, a plane tangent to the surface at the target point is defined using Equation 5.1, where $\vec{p}_*$ is the target point and $\vec{n}_*$ its surface normal.

$$f(\vec{x}): \quad \vec{n}_* \cdot (\vec{x} - \vec{p}_*) = 0 \tag{5.1}$$

A local 2D coordinate system $O_{\mathcal{L}}$ is defined inside such a plane, with its origin at $\vec{p}_*$ and following an arbitrary orientation β (this orientation will be later useful for the grasping task). Note that, by definition, $\vec{p}_*$ will be inside the plane $f(\vec{x})$.

Next, every point $\vec{p}_i \in \Phi$ is projected on $f(\vec{x})$ and translated to the local coordinate system $O_{\mathcal{L}}$. This yields an *image* set of points $\vec{q}_i \in \Phi_{IMG}$, completely contained in $f(\vec{x})$.

To define the orientation of each band a set D of N unitary vectors is generated. Each $\vec{\delta}_k \in D$ is defined inside $f(\vec{x})$, starting at $\vec{p}_*$. The set of N vectors is arranged symmetrically around $\vec{p}_*$, sorted counterclockwise, starting with the first vector, $\vec{\delta}_1$, parallel to the axis $O_{\mathcal{L}}\, X_{\mathcal{L}}$. Consequently, every $\vec{\delta}_k$ will be separated from the following vector $\vec{\delta}_{k+1}$ by an angle $\gamma = 2\pi/N$ (Figure 5.4).

The definition of each band is given by the equation of a line, presented in Equation 5.2.

$$\ell_k(t) = \vec{p}_* + t \cdot \vec{\delta}_k \tag{5.2}$$

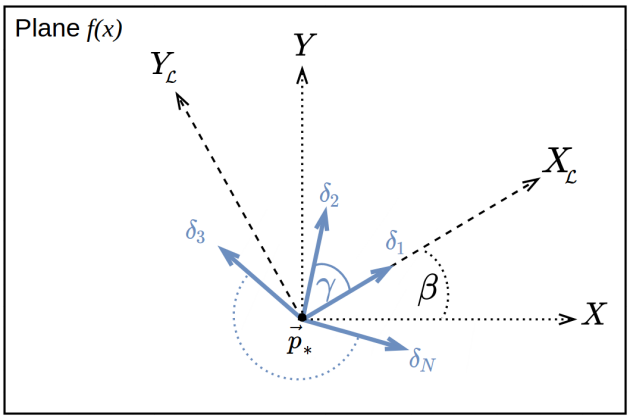


Figure 5.4: *The set of vectors $\vec{\delta}_k \in D$ are arranged symmetrically around $\vec{p}_*$. Each $\vec{\delta}_k$ act as the director vector of a band. The axes $X_{\mathcal{L}}$ and $Y_{\mathcal{L}}$ define the local coordinate system $O_{\mathcal{L}}$.*

All the points that are close enough to any line $\ell_k(t)$ will be part of the corresponding band B_k. For this purpose, the perpendicular distance between each point $\vec{q}_{\text{i}} \in \Phi_{IMG}$ to each line $\ell_k(t),\ k \in \{1, \ldots, N\}$ is measured. The points with a distance smaller than a threshold w are associated with the respective band. It is important to notice that this definition allows the presence of a point in more than one band, depending on the width of the band. This is the case of the points that are closer to the target point.

Finally, each band is defined as a subset of points $B_k \subseteq \Phi$ such that the image point $\vec{q}_{\text{i}}$ of each $\vec{p}_{\text{i}}$ is at a distance equal or less than w.

$$B_k = \{\vec{p}_{\text{i}} \in \Phi \ \mid \ \vec{q}_{\text{i}} \in \Phi_{IMG} \ \wedge \ dist\,(\vec{q}_{\text{i}},\ \ell_k(t)) \ \leq \ w\} \tag{5.3}$$

Since the set of bands $\{B_1, B_2, \ldots, B_N\}$ was extracted from Φ, then the length of each band will be at most r and its width will be $2 \cdot w$.

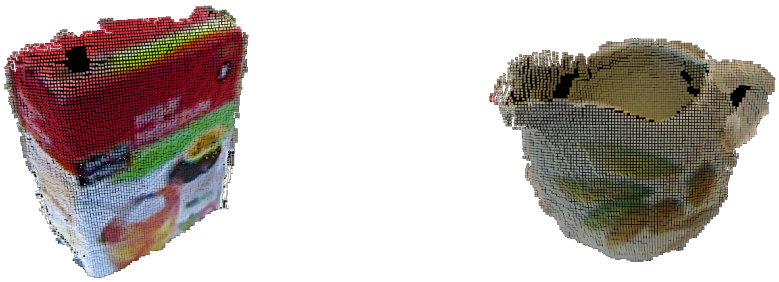
Descriptor Computation

The goal of this step is to encode the way in which the surface normal changes along each of the N bands defined for DCH. This is done by generating one histogram per band, using the angle between the surface normals as the target variable.

In detail, the angle between the surface normal at the target point, $\vec{n}_*$, and the surface normal at every point present in each band $\vec{n}_{\text{i}} \mid \vec{p}_{\text{i}} \in B_k$, is computed as shown in Equation 5.4. The results of such computations are grouped per band and then a histogram H_k is produced for each B_k.

$$\alpha_{\text{i}} = arctan\left(\frac{\|\vec{n}_* \times \vec{n}_{\text{i}}\|}{\vec{n}_* \cdot \vec{n}_{\text{i}}}\right)^{-1} \tag{5.4}$$

A histogram is used to encode the evolution of α_{i} because it allows the descriptor to reduce the effect of noise and errors in the estimation of the normals, as well as in the estimation of the angle. Since every band extends in a different direction, to use one histogram per band provides the descriptor with a way to encode information about the evolution of the surface



(a) *Three visible sides of a food box.* **(b)** *Visible surface of a pitcher.*

Figure 5.5: *Visible surfaces in point-clouds captured using a single sensing device.*[1]

normal in several directions. Is important to distinguish that such a description does not encode the spatial evolution of the curvature along the band, but rather the distribution of α_i. Therefore, each histogram only encodes the curvature features on a particular direction, given by the data present in each band, but not the spatial distribution of the curvature features along each band.

On the other hand, the use of several bands, symmetrically arranged, provides an overview of the surrounding surface. For such reasons this descriptor is named Directed Curvature Histograms (DCH).

Finally, the DCH descriptor vector, $\vec{\mathrm{d}}_{DCH}$, is formed by the concatenation of the histograms associated with each band (Equation 5.5).

$$\vec{\mathrm{d}}_{DCH} = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ \vdots \\ H_N \end{bmatrix} = \begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ \vdots \\ h_{N1} \\ h_{N2} \\ \vdots \end{bmatrix} = \begin{bmatrix} v_{11} \\ v_{12} \\ \vdots \\ \vdots \\ v_{N1} \\ v_{N2} \\ \vdots \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} band\ 1 \\ \vdots \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} band\ N \end{matrix} \tag{5.5}$$

Note that the resolution used for the histograms (that is, the size of each bin) will affect the size of the final descriptor. Smaller bins provide a fine-grained representation of the curvature of the object, but generate longer histograms, which will translate into a final descriptor of higher dimensionality.

Most of the time, the sensory data stored in a point-cloud is captured using a single sensing device, therefore the sensory data is captured from a single perspective. This implies that the surfaces contained in a point-cloud will show between one and three sides of an object (Figure 5.5), and the back of it will never be visible. This also implies that is unlikely for a point-cloud to contain surfaces showing a difference in the orientation of the normal vector bigger than $\pi/2$. For this reason, from now on the angular range represented by each histogram in DCH will be restricted to $[-\pi/2, \pi/2]$.

[1] Point-clouds from the RGB-D Object Dataset by Lai et al. [56].

Additionally, for the remaining part of this work the size of the bins used in the computation of DCH will be fixed in $\pi/18$ (10°), making the number of bins in each histogram equal to 18. This size is defined after testing DCH, in an attempt to balance dimensionality and descriptive power. Note that such a selection is suitable for the description requirements of the application approached by this work, but it is not the only choice. Therefore, a different application may use a different bin size, resulting in a different number of bins in each histogram, and therefore a different descriptor size.

5.2.1 Computation Parameters

The computation process of DCH, presented in Section 5.2, defines a computation function f_{DCH}, which takes a point-cloud C, and computes the descriptor vector $\vec{\mathrm{d}}_{DCH}$ associated to a given target point $\vec{p}_*$. According to the described process, there are 4 parameters required for the computation of the descriptor: i) the number of bands, N; ii) the size of the vicinity r, that is, the radius of the sphere used to extract the neighborhood of the target point; iii) the width of each band, w; and iv) the orientation of the first band, β. In consequence, f_{DCH} is defined as shown in Equation 5.6.

$$f_{DCH}\left(C,\ \vec{p}_*,\ N,\ r,\ w,\ \beta\right) \ =\ \vec{\mathrm{d}}_{DCH} \tag{5.6}$$

In general terms, the value of each of these parameters can be defined using the context provided by the problem, as follows:

- **Number of Bands** N: a higher number of bands allows DCH to encode information about the surface in a higher number of directions, but it increases the size of the final descriptor. Since each histogram has 18 bins, every extra band will increase the size of the final descriptor by 18. Is important to note that when N is too high, the difference between each band becomes minimal (the overlap between bands tends to be high), which increases the size of the descriptor without increasing the encoded information.
- **Size of the Vicinity** r: this parameter is related to the curvature of the graspable zone. Since DCH reflects the curvature in the vicinity of a point, which will help identify which zones are suitable to perform a grasp, this parameter should reflect the maximum aperture of the effector; hence the maximum distance between points compared to estimate the angle between normals.
- **Width of the Bands** w: the width of the bands is directly dependent on the conditions of the problem. Depending on the relative location of the 3D sensor and the target object, the density of points will vary, which will be reflected in the number of points in each band. If the bands are too thin, few points will be in each of them, making the contents of the histograms not representative of the underlying surface. On the other hand, if the bands are too wide, the overlap between them will be high, which reduces the directionality of the encoded information.
- **Orientation of the First Band** β: in general this parameter can be arbitrarily defined depending on the problem. It is important to note that, to be capable of comparing instances of DCH, this parameter should to be the same between instances.

It is arguable if some of these parameters should be defined in a way different to the one suggested in this Section. Most of the time an arbitrary definition would be acceptable for parameters N, w and β. For example, a higher number of bands could be used for objects with irregular geometries, in order to produce a characterization of the surface capable of accounting for such irregularities. It is important to keep in mind how each of these parameters affects the behavior of the descriptor (dimensionality, descriptive power, among others).

This Section also suggests to define r according to the features of the robot, in particular according to the maximum aperture of the effector. A different idea may be to define this parameter according to the features of the target object (height, width, etc.), which might lead to use high-level features as indicators used for the definition of grasping strategies. However, such an approach does not take into account that independently of the dimensions of the object, the grasping abilities of a robot are limited by the maximum aperture of its grip. Therefore, even when a proper grasping strategy is defined, it still will have to pass through a validation step, where the the actual feasibility of the grasp will have to be evaluated according to the maximum aperture of the grip. On the other hand, defining r according to the features of the robot allows to use DCH to describe the object directly taking into account such limitations.

5.2.2 Computational Complexity

The theoretical complexity of the DCH computation process can be analyzed in terms of the parameters presented in Section 5.2.1. For this analysis, n will be defined as the number of points in the input cloud $n = |C|$, and n_v as the number of points in the vicinity of the target point $\vec{p}_*$, which directly depends on r, that is $n_v(r)$. Next, the complexity of the computation process can be studied using the previously presented division into two steps as follows:

- **Bands Extraction:** this step can be further divided into four sub-steps.

  i) **Vicinity extraction:** this is done using a KD-Tree [57], in order to perform fast range queries. Therefore the computational costs involved include building the tree, with a complexity of $\mathcal{O}(n \log n)$; and performing a range search to extract the points in the vicinity of the target point, with a complexity of $\mathcal{O}\left(n^{2/3} + n_v(r)\right)$ for three-dimensional data. Therefore, the complexity of this sub-step is as presented in Equation 5.7.

  $$\mathcal{O}(n \log n) + \mathcal{O}\left(n^{2/3} + n_v(r)\right) \tag{5.7}$$

  In the worst case $n_v(r) = n$, which would be equivalent to compute DCH using a vicinity size r big enough to include the whole cloud C in it. In such a case, the second term of Equation 5.7 can be bounded as shown in Equation 5.8.

  $$\mathcal{O}\left(n^{2/3} + n_v(r)\right) \leq \mathcal{O}\left(n^{2/3} + n\right) = \mathcal{O}(n) \tag{5.8}$$

  Also, note that $\mathcal{O}(n) < \mathcal{O}(n \log n)$. Therefore in the worst case, the time complexity of this sub-step will be $\mathcal{O}(n \log n)$.

ii) **Tangent plane definition:** this definition is constant regardless of the computation parameters, hence the complexity of this sub-step is $\mathcal{O}(1)$.

iii) **Bands direction definition:** the number of directions computed to define each band depends directly on the number of bands; in consequence, this sub-step has a complexity of $\mathcal{O}(N)$.

iv) **Data extraction:** the definition of which points are part of each band is done by traversing every point in the vicinity and checking if it is close enough to any of the bands, therefore the complexity is $\mathcal{O}(n_v(r)N)$. As previously mentioned, in the worst case $n_v(r) = n$, hence the worst case complexity will be $\mathcal{O}(nN)$.

Since the number of bands is always much smaller than the number of points, $N \ll n \ \wedge \ N \ll n_v(r)$ (n is usually on the order of 5,000 to 10,000 points and $n_v(r)$ on the order of 500 to 700 points), then only the terms coming from sub-steps i) and iv) will be relevant for the complexity analysis. Given that $\mathcal{O}(nN) < \mathcal{O}(n \log n)$, the time complexity of this step will be $\mathcal{O}(n \log n)$.

- **Descriptor Computation:** in this step, the change in orientation of the normal vector is computed for every point in each band. Let m_{i} be the number of points present in the i-th band; then the complexity of this step for each band will be $\mathcal{O}(m_{\mathrm{i}})$. If m is defined as $m = \max\{m_0, m_1, \ldots, m_N\}$, then it follows that $\mathcal{O}(m_{\mathrm{i}}) \leq \mathcal{O}(m)$. Finally, the total complexity of this step will be $\mathcal{O}(mN)$.

The complexity of the complete computation process will be the sum of the complexity of both steps $\mathcal{O}(n \log n) + \mathcal{O}(mN)$.

The worst case scenario here would be to compute DCH using a vicinity size r big enough to include the whole cloud C in it, plus using a band width w large enough to contain half of the vicinity in each band (since each band start at the center of the vicinity, one band at most could cover half of it). Under such circumstances $m = n/2$, therefore the complexity can be bounded as presented in Equation 5.9.

$$\mathcal{O}(n \log n) + \mathcal{O}(mN) \leq \mathcal{O}(n \log n) + \mathcal{O}\left(\frac{n}{2}N\right) = \mathcal{O}(n \log n) \tag{5.9}$$

Finally, the worst case time complexity for the computation of DCH is given by $\mathcal{O}(n \log n)$.

5.3 Coordinate System Selection

As mentioned in Section 5.2, the orientation of the coordinate system used for the computation of the descriptor is arbitrarily defined through the parameter β. The advantage in the existence of such a feature resides in the fact that it can be exploited according to the problem being approached.

The formulation of DCH allows users to define the most convenient reference frame for the application they are working on, ranging from an ad-hoc orientation to a general reference

frame. The only requirement for a general formulation, which can be used across different instances of the descriptor and different experiments, is that the reference frame has to be repeatable and consistent, so different instances DCH can be compared.

One possible general formulation can be achieved by simply using a fixed orientation, defined in a global coordinate system. A different option is to use a repeatable local reference frame. For example, Tombari et al. [46] proposed the definition of a local reference frame using the principal directions computed through an Eigenvalue Decomposition (EVD) or a Singular Value Decomposition (SVD) of the covariance matrix of the point coordinates, plus a heuristic procedure for the disambiguation of the sign of the reference frame vectors.

In broad terms, the advantage of using a general formulation is that it provides the descriptor with rotation and translation invariance.

5.3.1 Coordinate System for Robotic Grasping

The proposed descriptor can be used to solve the robotic grasping problem (Chapter 3), by following a data-driven approach. For this problem, the best choice for the coordinate system which is used to define the orientation of the descriptor is to select it to match the orientation of the coordinate system of the effector, that is, match the orientation of axis $X_{\mathcal{L}}Y_{\mathcal{L}}$ (Figure 5.4) with axis $Z_{eff}Y_{eff}$. When the orientation of both reference frames is matched, then the orientation of band B_0 will be the same as the orientation of the effector.

This formulation allows the descriptor to encode the curvature in the surface as observed from the coordinate system of the effector. This implicitly encodes the orientation of the effector in the descriptor vector, which can be later used to perform a supervised learning process, focused on identifying the observed curvature of the object when a successful/unsuccessful grasp is performed.

5.4 Examples

Figure 5.6 presents 3 examples showing the results of the computation of DCH on different surfaces. Each example shows the location in a point-cloud where DCH was computed (using 8 bands) and the histogram associated with each band. It is interesting to note that DCH effectively reflects the underlying surface in each example. In the first example, the descriptor is computed on a plane (Figure 5.6a). The histograms associated to such an example (Figure 5.6b) show that there is no change in the angle of the normal vector in any direction, reflecting the nature of the plane.

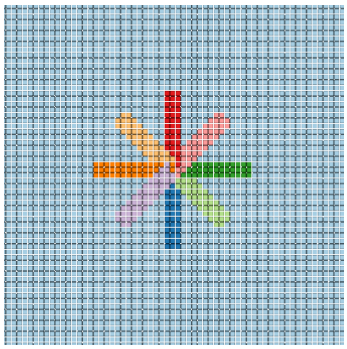
In the second example, the descriptor is computed on the surface of a sphere (Figure 5.6c). In this case, the histograms of the descriptor (Figure 5.6d) show the same behavior in the orientation of the normal vector in every direction, reflecting the symmetrical nature of the sphere.

In the third example, DCH is computed on the surface of a cylinder (Figure 5.6e). The histograms of the descriptor in this case (Figure 5.6f) reveal a mix of the features observed in the plane and the sphere. Vertical bands show histograms reflecting no change in the orientation of the normal vector, while horizontal bands reflect changes in the orientation of the normal which are similar to the behavior observed for the sphere. This example shows the ability of DCH to encode the curvature of the surface in different directions.

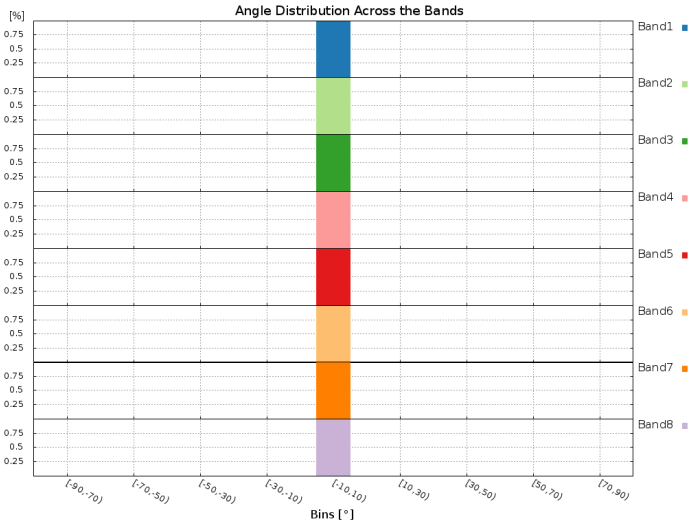
Finally, Figure 5.7 presents the results of computing DCH using different angles for the orientation of the first band.

5.5 Summary

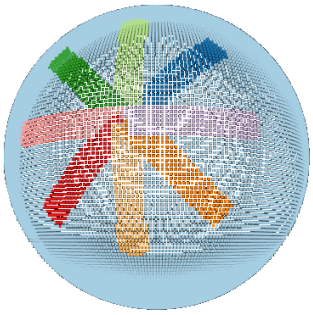
This chapter presented a novel contribution to the 3D local description problem. The proposed method, Directed Curvature Histograms (DCH), presents a description technique of an object based on the evolution of its curvature in different directions. The examples presented show the ability of DCH to characterize different surfaces, reflecting changes in the curvature along each band of sensory data.



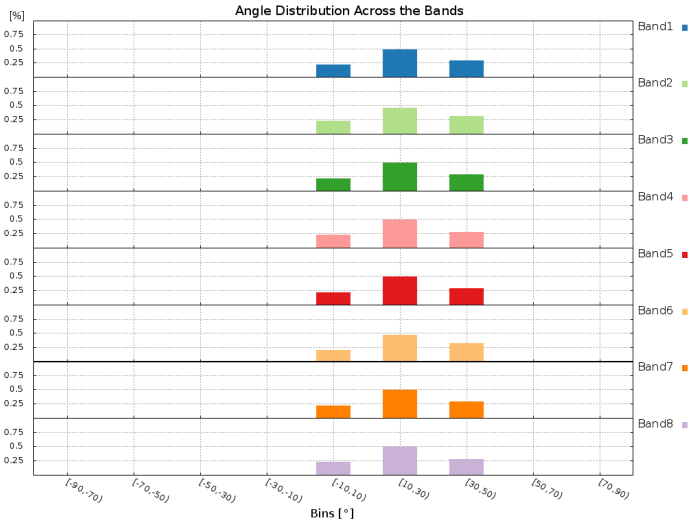
(a) *DCH computed on a plane.*



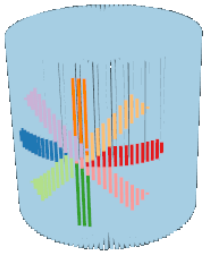
(b) *Histograms associated with each band.*



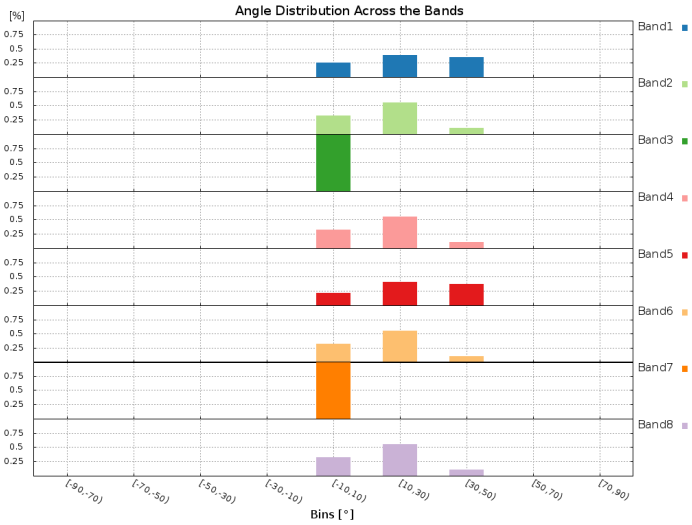
(c) *DCH computed on a sphere.*



(d) *Histograms associated with each band.*

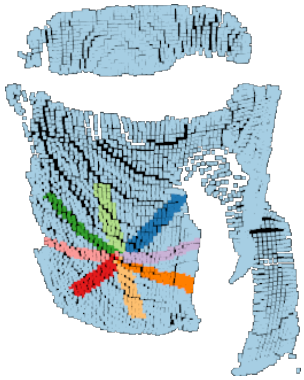


(e) *DCH computed on a cylinder.*

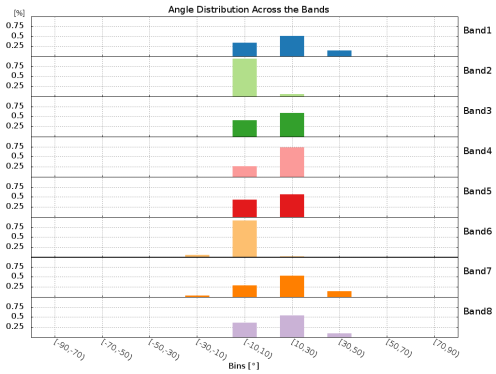


(f) *Histograms associated with each band.*

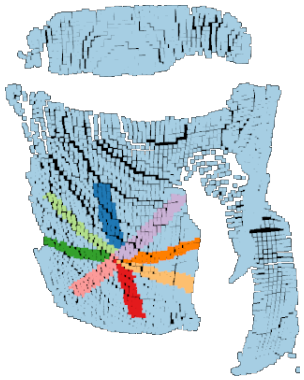
Figure 5.6: *Computation of DCH on different point-clouds surfaces.*



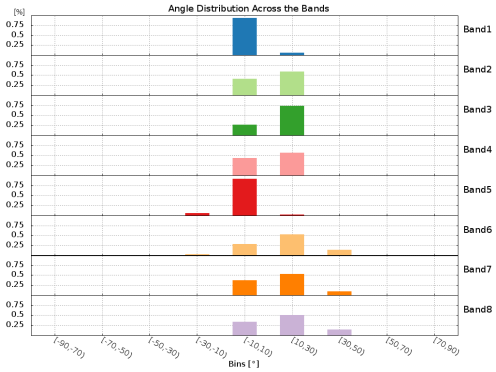
(a) *Bands location for computation angle 0°.*



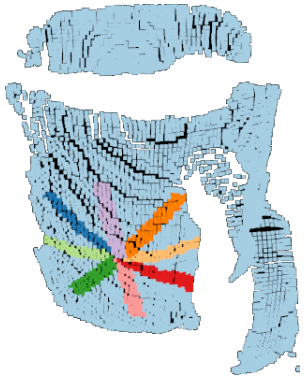
(b) *Histograms for computation angle 0°.*



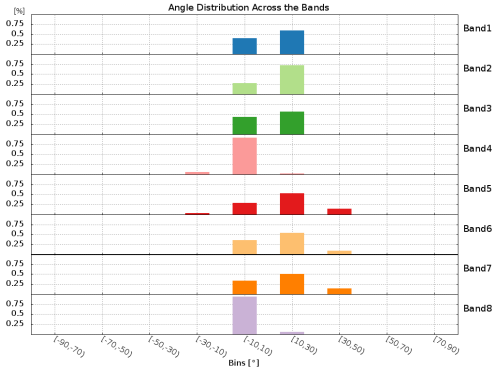
(c) *Bands location for computation angle 45°.*



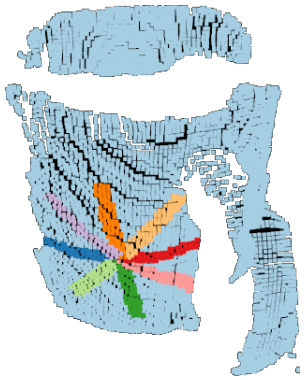
(d) *Histograms for computation angle 45°.*



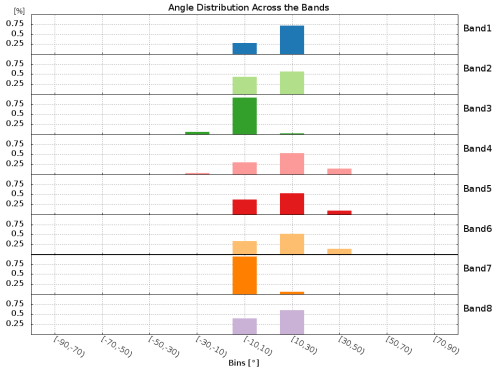
(e) *Bands location for computation angle 90°.*



(f) *Histograms for computation angle 90°.*



(g) *Bands location for computation angle 135°.*



(h) *Histograms for computation angle 135°.*

Figure 5.7: *Computation of DCH using different angles.*

Chapter 6

Proposed Grasp Synthesis Method

The grasp synthesis method proposed in this thesis can be categorized as a data-driven approach (Section 3.2). The proposed method takes advantage of the observation made by Feix et al. [1] (described in Section 5.1), which points out that humans tend to grasp objects by their smallest dimension. This is done by using the sensory data description performed by DCH, together with machine learning techniques, to develop a method which identifies suitable grasping zones in an object, as well as proper parameterizations for the effector, all this based on previous grasping experience.

As stated in Section 2.3, the PR2 robot was selected for the development of this method, given its capabilities to perform grasping tasks. Accordingly, the following descriptions assume the availability of the sensors present in a PR2 robot and the use of a gripper as effector. However, the proposed method can be easily used with a different robotic platform, as long as such the robot is equipped with a 3D sensor providing depth images organized as point-clouds, and a robotic effector that can be used as a gripper.

Additionally, all the descriptions use the coordinate system defined for the effector as shown in Figure 5.1.

6.1 Premise and Design

This grasp synthesis method relies on two base premises: i) the curvature information, obtained from previous grasping experiences, can be exploited to identify the zones in an object which are better suited for grasping; and ii) the curvature around a given point can be used to identify a proper parameterization for the effector that would allow grasping an object. These premises are based on the observation that humans repeat a grasp because it is common to find different objects having zones with a similar shape; and on the observation that humans grasp elements by their shortest dimension, as previously stated.

The proposed method assumes that the curvature information can be used to learn to identify suitable grasping zones and parameterizations for the effector, which would allow a

robot to perform a successful grasp (Hypothesis IV and Hypothesis V). According to such assumptions, the proposed method can be categorized as a grasp-of-familiar-objects approach (Section 3.2.2).

The existing methods following a grasp-of-familiar-objects approach typically focus on shape recognition and matching to define a grasping point, after which is used an additional step to define a proper parameterization. Unlike those methods, the proposed approach focuses on simultaneously learning to identify suitable grasping points, as well as suitable parameterizations to grasp an arbitrary object, taking advantage of the 3D sensory information. Also, the proposed method directly attempts to mock (through a learning process) the notions and the decision process performed by humans as reflected in the conclusions reached by Feix et al. [1]. In this case, such conclusions influenced the design of the grasp synthesis method, which focuses on using the changes in curvature as a meaningful way to identify the proper parameterizations for grasping.

For this method, a suitable parameterization of the effector will be understood as a rotation around axis X_{eff} (Figure 5.1), such that the vertical axis Z_{eff} is aligned with the axis of the lower curvature in the target object (Figure 5.2).

6.1.1 Design

Following the presented premise, the proposed grasp synthesis method is designed to take advantage of the description done by DCH, together with unsupervised and supervised learning algorithms, in order to learn the features of a successful grasp. To ease the understanding of the proposed method, it is divided into 4 stages, according to the goal of each stage (Figure 6.1): i) feature space reduction; ii) grasp candidates generation, iii) learning; and iv) knowledge exploitation.

6.1.2 DCH Parameters

According to Section 5.2.1, the parameters that need to be defined to compute DCH are: i) the number of bands, N; ii) the size of the vicinity r; iii) the width of each band, w; and iv) the orientation of the first band of the descriptor, β. The context of the problem (the robotic grasping problem) has to be taken into account to define values for such parameters. Based on such context, these parameters were chosen as follows:

- **Number of Bands** N: eight were selected because such a number represents a good compromise between the final size of the descriptor and the number of directions represented by it. Since each histogram has 18 bins (as noted in Section 5.2), the final size of the descriptor used for the development of this work is $|DCH| = 8 \cdot 18 = 144$.
- **Size of the Vicinity** r: since the maximum aperture of the gripper of the PR2 robot is 7 cm, the size of the vicinity was selected to be 3 cm, leaving some room to account for computation errors in the definition of the vicinity.

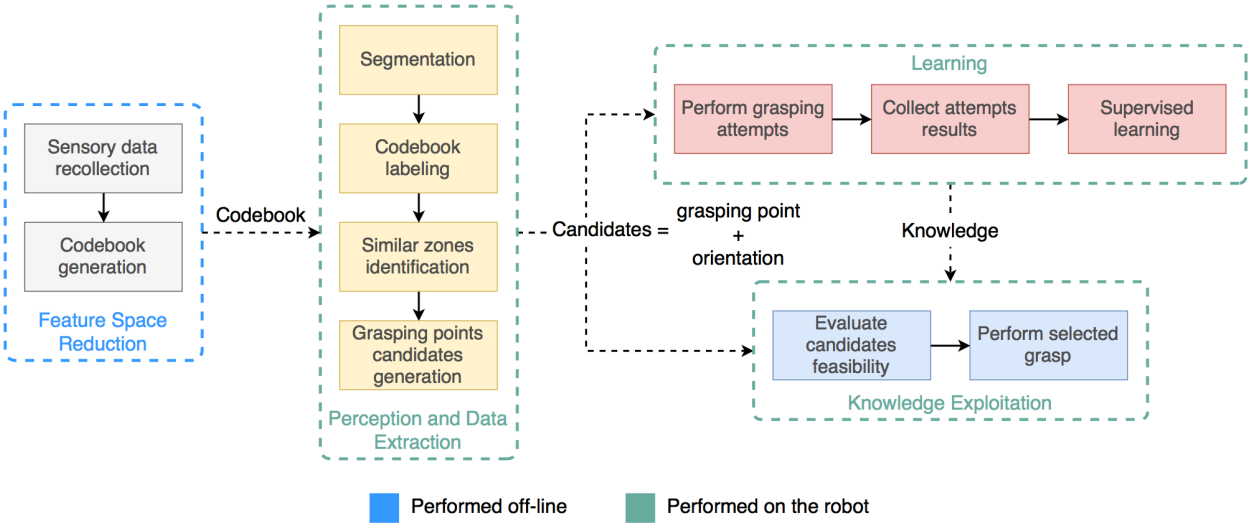


Figure 6.1: *Design of the proposed grasp synthesis method.*

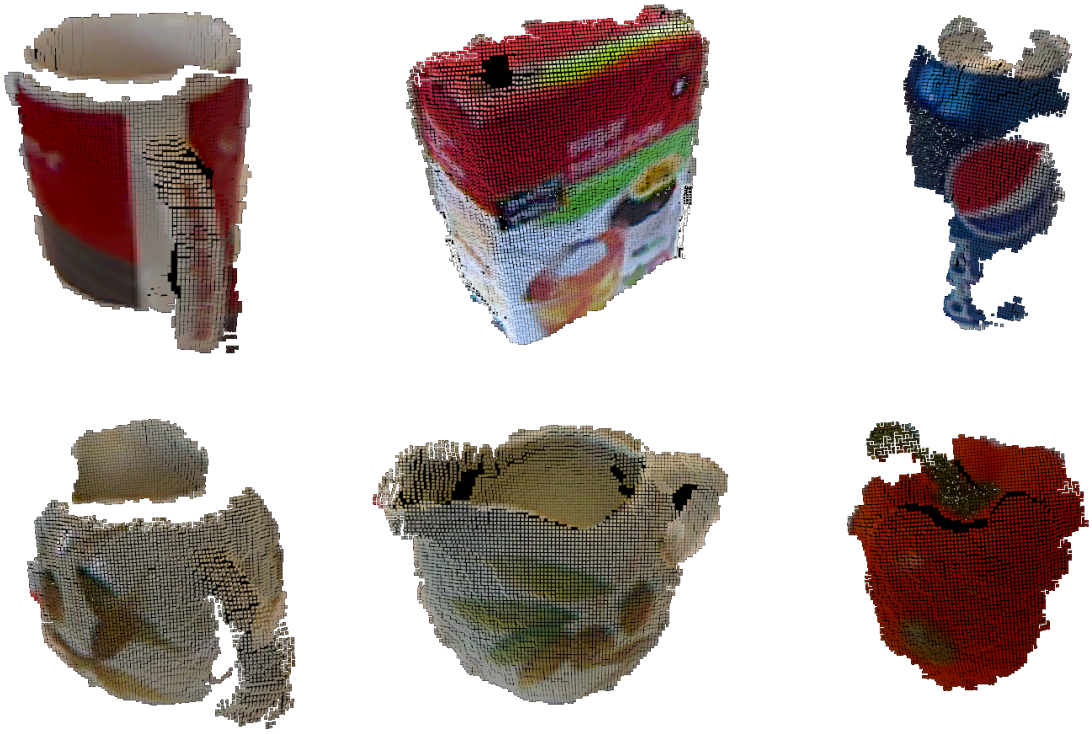


Figure 6.2: *Example of the point-clouds present in the RGB-D Object Dataset.*[1]

- **Width of the Bands** w: given the conditions of the setup used for testing and for the experiments, the width of the bands was selected as 0.5 cm.
- **Orientation of the First Band,** β: during the first stage of the proposed method (Figure 6.1), β was selected as 0 because the compared descriptor vectors, $\vec{d}_{DCH}$, were computed using different point-clouds. Since the sensory data has a constant coordinate system, keeping the same β ensures the comparability of the computed instances of

[1] Point-clouds from the RGB-D Object Dataset by Lai et al. [56].

DCH. On the other hand, for the remaining stages the value of β was selected as equal to the orientation of the effector. The goal of this is to allow DCH to encode the curvature of the surface, according to the coordinate system of the effector.

6.2 Stages of the Proposed Method

The proposed grasp synthesis method was implemented following the design presented in Figure 6.1. The following section describes the details of such implementation.

6.2.1 Stage 1: Feature Space Reduction

In this stage, the feature space, implicitly defined by the use of the DCH as the description method, is reduced following the approach of the BoVW model. The result is a codebook which represents said feature space.

This is the only stage performed off-line, meaning that it is carried out prior to any grasping attempt and it can be performed outside the robot (not using the computer mounted on the robot). This is because, in order to reduce the feature space, a representative sample of DCH instances has to be used as input for the BoVW model. However, such a set of DCH instances cannot be obtained simultaneously with a single robot. In consequence, a set of point-clouds, coming from the *RGB-D Object Dataset* [56], is used to perform this stage.

The RGB-D Object Dataset is composed by point-clouds of 300 objects, captured using a Microsoft Kinect as 3D sensor. This dataset contains multiple partial views of different objects (Figure 6.2).

In order to generate a representative sample of DCH instances, a subset of 60 point-clouds is selected from the RGB-D Object Dataset. Then the function f_{DCH} is evaluated on every point of each point-cloud from such a subset, procedure commonly referred to as *dense evaluation*. This produces a set of descriptor vectors, $\vec{\mathrm{d}}_{DCH}$, with a number of elements equal to the total of points present in the subset of 60 clouds.

Since each point-cloud coming from the RGB-D Object Dataset has between 4,000 to 8,000 points, the total number of DCH instances produced by this procedure is in the order of 400,000. In practice, the computation of a codebook using such amount of data cannot be done in a reasonable time using (less than one day). There are two factors that explain this issue:

i) The dense evaluation process involves the computation of DCH as many times as the number of points in the point-cloud evaluated, m. Since the computation of DCH has a complexity of $\mathcal{O}(n \log n)$, then complexity of the dense evaluation would be $\mathcal{O}(m \cdot n \log n)$, where typically $m > n$. Finally, if this process is repeated in q point-clouds, then the complexity of this step will be $\mathcal{O}(q \cdot m \cdot n \log n)$.

ii) The computational complexity of each iteration of the standard K-Means algorithm is

$\mathcal{O}(n \cdot k \cdot d)$, where n is the number of points processed, k is the number of clusters, d is the dimension of each point ($d = 18 \cdot N$). Therefore, if the algorithm requires i iterations to find the k desired clusters, then the complexity of the whole computation would be $\mathcal{O}(n \cdot k \cdot d \cdot i)$.

The combination of both factors makes the reduction of the feature space a computationally expensive task. In particular, the computational cost of such a process is highly affected by K-Means.

To address this issue, an initial reduction of the number of descriptors is performed. Such reduction is done by simply applying the clustering algorithm K-Means to the set of descriptors generated from each cloud, to produce a set of representative vectors per cloud. In this case, the size of this preliminary representative set was selected to be 200 vectors per cloud. This reduces the total number of descriptors to process to $60 \cdot 200 = 12,000$.

The resulting set is used as input to the BoVW model, to generate a codebook of size K. As in most clustering problems, a big value for K (a big codebook) produce a large set of representative vectors, with several vectors showing high levels of similarity between them. When applied to the identification of similar zones in a point-cloud, a big codebook tends to generate a large number of small blobs of similar features. On the other hand, a small value for K (a small codebook) produce a smaller number of representative vector, which tends to be composed by very different vectors. When a smaller codebook is applied to the identification of similar zones in a point-cloud, it tends to produce fewer larger blobs, which group areas in the surface of the object.

In this case, and after evaluating the problem, the size of the codebook chosen was $K = 7$, that is, the computed codebook has 7 vectors representative of the DCH feature space. This value was selected in an attempt to find balance between the ability of the reduced feature space to represent different groups of features, and to avoid and generation of an excessive number of small blobs of similar features on the surface of the object.

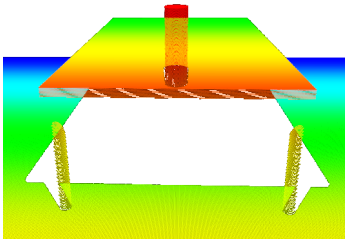
Also, note that in this case the initial reduction of the number of descriptors was done using K-Means given its simplicity and straight forward applicability. However, a different clustering algorithm might be used to either privilege the prevalence of a particular group of features after the clustering stage, or to produce a codebook better suited to the specific characteristics of the feature space.

6.2.2 Stage 2: Grasp Candidates Generation

This is the first stage actually performed by the robot. Its goal is to produce a set of grasp candidates, that is, a set of points on the surface of the target object plus the corresponding orientations for the effector of the robot, which can be used to perform grasps on the target object.

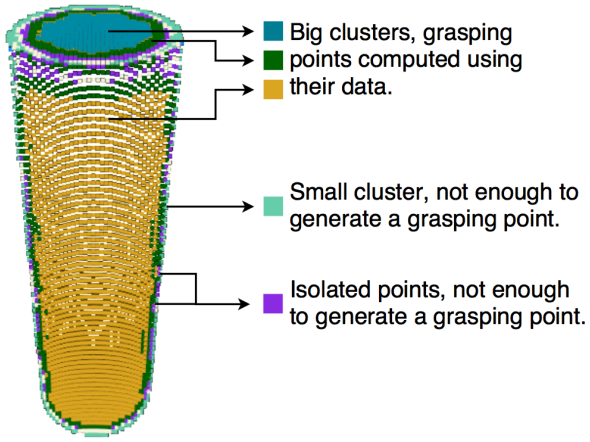
This stage starts with the segmentation of a point-cloud C of size S, captured by the 3D sensor mounted in the head of the robot (Figure 6.3b). Since the focus of this work is

(a) *Example of an experimental setup.*



(b) *Captured point-cloud. The color reflects the depth of the environment.*

(c) *Segmented point-cloud.*



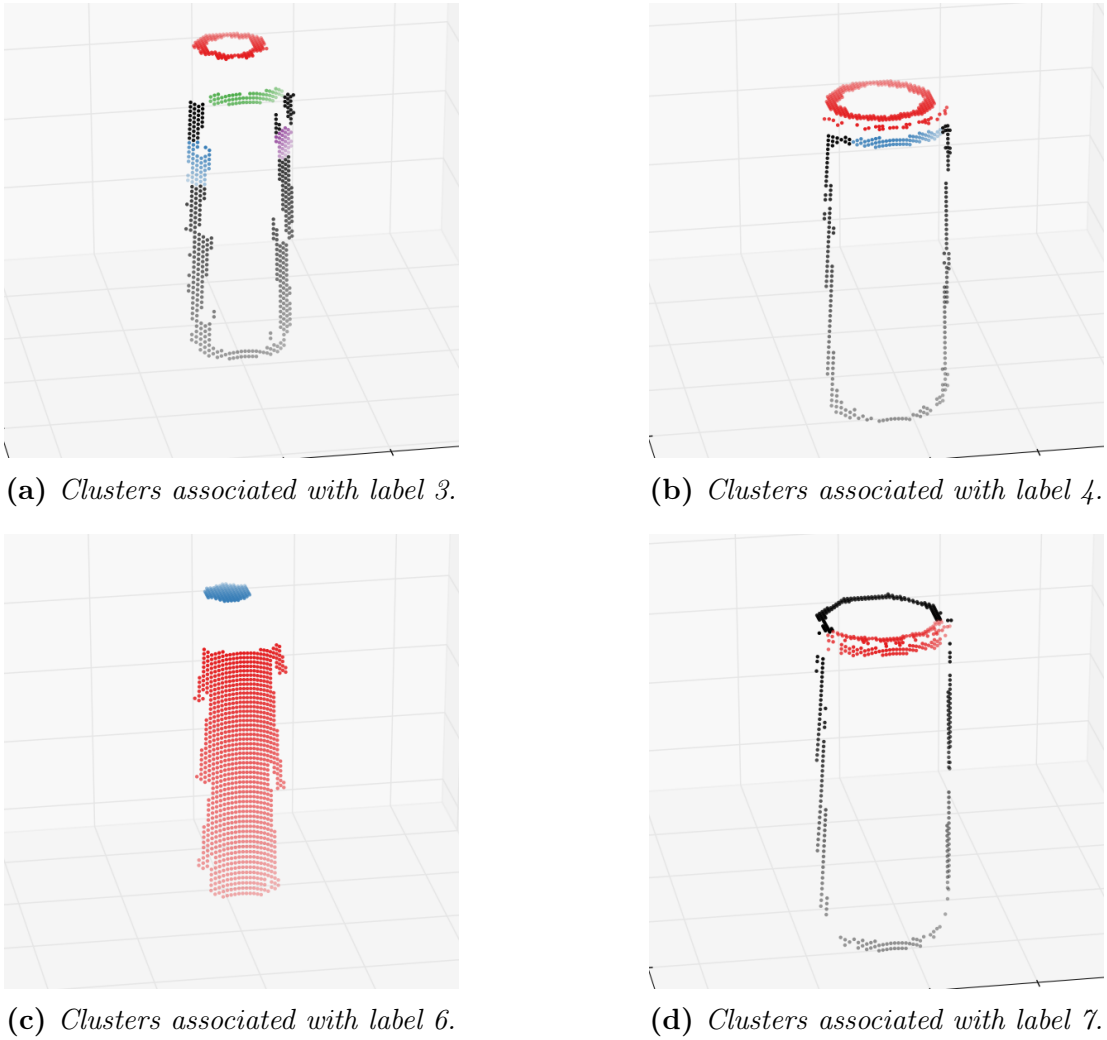
(d) *Labeled point-cloud. Each color represents a different group of points which may or may not produce a grasping point.*

Figure 6.3: *Segmentation of a cloud captured by the robot.*

on the development of the grasp synthesis method, the setup for the grasping attempts is a simplified version of a real grasping environment, which only includes a table and the target object (described in Chapter 7). An example of such a setup can be seen in Figure 6.3a. For this reason, the segmentation of the captured cloud is performed by simply identifying the location of the table under the object and then clipping the point-cloud to leave only the target object (Figure 6.3c), which generates a cloud C_{seg} of size $S_{seg} \leq S$.

Next, a dense evaluation of f_{DCH} is performed over the segmented cloud, which creates a set Ω_{DCH}, formed by S_{seg} descriptor vectors. Such a set is labeled using the codebook produced in Stage 1, that is, for every element in $\vec{d} \in \Omega_{DCH}$ the closest element in the codebook is selected, and then its ID is used as a label for $\vec{d}$. This generates a labeled set of points Ω_{label}. Note that since the codebook size is $K \ll S_{seg}$, the labeled set will have large groups of points sharing the same label, as shown in Figure 6.3d. These groups define zones of similar features, according to the description done by DCH.

The grasp candidates are generated using the zones of similar features, based on the mass center of the labeled points plus a given orientation. However, most of the time the mass



(a) *Clusters associated with label 3.*

(b) *Clusters associated with label 4.*

(c) *Clusters associated with label 6.*

(d) *Clusters associated with label 7.*

Figure 6.4: *Results of the DBSCAN algorithm on the point-cloud divided by label. Each color represents a different cluster found by the algorithm.*

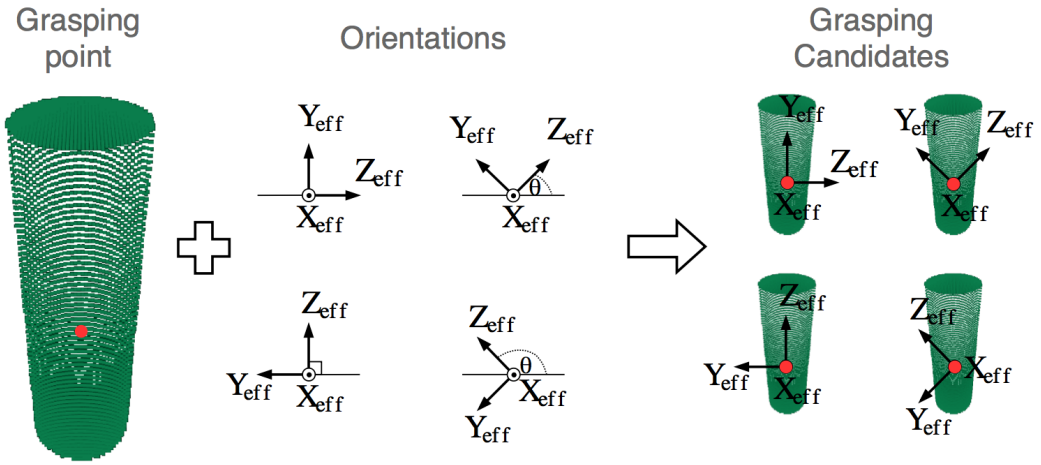


Figure 6.5: *The grasp candidates are generating by placing the coordinate system of the effector at the grasping point, with different orientations, for example 4 different orientations.*

center of these zones cannot be directly computed since there are several issues that may affect the correct computation, such as:

- The points sharing the same label are not always adjacent.
- The points sharing the same label may be part of different spatial clusters.
- The existence of small groups of points sharing some labels.
- The presence of zones with a low density of points sharing the same label as zones of high density.

In order to properly generate the grasp candidates, Ω_{label} is divided into K subsets, according to the label of each point. Each subset is defined by Equation 6.1.

$$\Omega_{\mathrm{i}} = \{p \in \Omega_{label} \mid label(p) = \mathrm{i}\},\ \mathrm{i} \in \{1, \ldots, K\} \tag{6.1}$$

Note that by following this definition, then $\Omega_{label} = \Omega_1 \cup \Omega_2 \cup \ldots \Omega_K$. After dividing Ω_{label}, the density based clustering algorithm DBSCAN (Section 2.1.5) is applied to each Ω_{i}. The goal here is to produce a set of clusters of contiguous points sharing the same label, while simultaneously points located in zones of low density are discarded. Figure 6.4 shows the clusters found on the point-cloud presented in Figure 6.3d, when DBSCAN is applied to the sub-clouds produced by dividing the original cloud according to the label of each point.

This process generates multiple clusters per Ω_{i}. These are analyzed, discarding the clusters with few points and leaving M_{i} clusters per Ω_{i}. The total number of produced clusters is given by Equation 6.2.

$$M = \sum_{\mathrm{i}=1}^{K} M_{\mathrm{i}} \tag{6.2}$$

The computation of the mass center of each cluster produces a set $P = \{p_1, \ldots, p_M\}$ of grasping points. Then, a set of arbitrary angles $\Theta = \{\theta_1, \ldots, \theta_A\}$ is used to finally generate a set G of grasp candidates, by forming pairs (point, angle) with all the possible combinations between P and Θ (Figure 6.5).

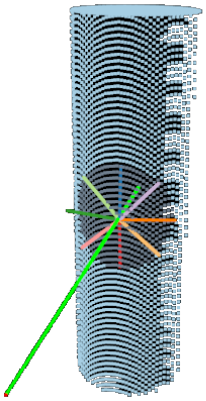
$$G = P \times \Theta = \{g_1, \ldots, g_{|G|}\} = \{(p_1, \theta_1), (p_1, \theta_2), \ldots, (p_M, \theta_{A-1}), (p_M, \theta_A)\} \tag{6.3}$$

It follows that the number of grasp candidates produced is $|G| = M \cdot A$. The number and value of the angles in Θ is defined as part of the parameters of the experimental setup, described in Chapter 7. Figure 6.6 shows an example of the grasp candidates generated from one of the clusters found using DBSCAN.

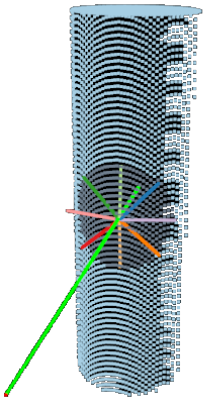
6.2.3 Stage 3: Learning

The goal of this stage is to use the set G of grasp candidates generated in Stage 2 to perform a series of grasping attempts, focused on collecting data for learning. Then, the collected results of each attempt are used to carry out a supervised learning process, focused on learning a relation between the result of the grasp attempt and the grasp candidate.

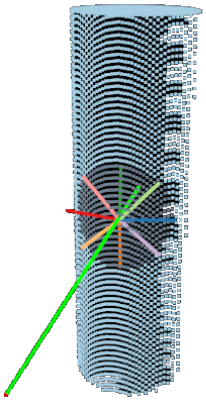
This stage is can be subdivided into two parts: i) Data Capture, and ii) Learning.



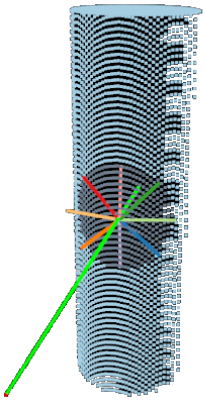
(a) *Candidate produced using orientation 0.*



(b) *Candidate produced using orientation $\pi/4$.*



(c) *Candidate produced using orientation $\pi/2$.*



(d) *Candidate produced using orientation $3\pi/4$.*

Figure 6.6: *Example of grasp candidates produced by the described method. The colored lines show the axes of each band, the green line depicts the axis used by the effector to approach the object.*

Data Capture

This part starts with the computation of one descriptor vector $\vec{\mathrm{d}}_{DCH}$ for each grasp candidate $g_{\mathrm{i}} = (p_j, \theta_k) \in G$ using the parameters defined in Section 6.1.2, and a point-cloud C captured by the robot. The computation is performed using the grasping point and angle provided by the grasp candidate.

$$f_{DCH}\,(C,\ p_j,\ N,\ r,\ w,\ \theta_k)\ =\ \vec{\mathrm{d}}_{DCH} \tag{6.4}$$

This ensures that the orientation used to calculate $\vec{\mathrm{d}}_{DCH}$ (orientation of the first band) remains equal to the orientation of the coordinate system of the effector, as stated in Section 6.1.2. Then, a grasping attempt is performed for every candidate present in G. Each attempt focuses on grasping the target object by positioning the center of the effector on the grasping point p_j, while setting the orientation of the effector to θ_k. Three variables are recorded during each attempt:

- The grasp candidate used, $g_{\text{i}} = (p_j, \theta_k)$.
- The descriptor vector $\vec{\text{d}}_{DCH}$.
- The result of the attempt, that is, if the object was successfully grasped or not, which is represented as a binary variable (1 for success, 0 for failure).

Learning

Finally, the data recorded in all the grasping attempts is used to perform a supervised learning process. Here, a classifier is trained using the set of recorded $\vec{\text{d}}_{DCH}$ as input variable and the result of each attempt as the target variable. This allows the learning process to produce a function H, which represents the learned relation between the curvature observed from the coordinate system of the effector, encoded in $\vec{\text{d}}_{DCH}$, and the result of the grasping attempt. Note that the learned relation relates the curvature of the object, as observed from the coordinate system of the effector, because the computation process of $\vec{\text{d}}_{DCH}$ was performed using θ_k as orientation, and therefore the descriptor itself encodes the orientation of the effector.

Ultimately, the learned relation can be used as a function to predict the outcome of future grasp attempts.

$$H\left(\vec{\text{d}}_{DCH}\right) = \begin{cases} 1, & \text{predicted success} \\ 0, & \text{predicted failure} \end{cases} \tag{6.5}$$

It is important to note that the relation H provides the ability to predict the result of a given grasp candidate g_{i}, but it does not allow the robot to predict the grasp candidate itself. For this reason, the proposed grasp synthesis method necessarily has to be a two-step procedure, that is, perception + learning/exploitation.

During the experimental evaluation of this method a SVM classifier (Section 2.1.3) was trained and used as a concrete instance of the function H. Such a method was selected for two reasons: i) it is capable of performing multi-class classification, and ii) it provides a way to define a score to assess the feasibility of each prediction performed by the classifier. For this particular classifying technique, such a score can be defined by using the distance of the classified sample to the classifying hyperplane defined by SVM. Despite this selection, a different classifier could have also been selected.

6.2.4 Stage 4: Knowledge Exploitation

The goal of this stage is to exploit the knowledge learned in Stage 3 to shape a system capable of autonomously deciding a suitable grasping point and orientation for the effector, according to the captured sensory data (as a point-cloud).

This step starts by receiving the set of grasp candidates G, generated in Stage 2. Then, the descriptor vector $\vec{\text{d}}_{DCH}$ associated to each grasp candidate $g_{\text{i}} \in G$ is computed using Equation 6.4. Such a descriptor is used to assess the feasibility of each candidate to be used

to perform a successful grasping attempt. This is done by using the learned function H (Equation 6.5).

Depending on the type of classifier trained during Stage 3, the learned function H may also be able to estimate a score associated to the feasibility of the evaluated grasping candidate. In such cases, the candidates are sorted according to the score provided by H, then, the best feasible candidate deemed to be feasible is used to perform the grasp of the target object.

On the contrary, if the trained classifier does not provide a learned function H with a method to estimate a score associated with each prediction, then all the feasible candidates are tested on the target object. This is done until a successful grasping attempt is performed. Although this alternative does not provide a single candidate to test, as might be expected from a method to generate grasping strategies, it reduces the set of grasp candidates to a small set, preventing the robot from performing a longer exploratory step.

6.3 Summary

This chapter presented the proposed grasp synthesis method, which can be classified as a data-driven approach. The main idea behind this proposal stems from the fact that humans tend to repeat grasp strategies and usually grasp an object by its smallest dimension. The proposed method uses the experience collected on previous grasp attempts to decide the best grasping strategy for new objects. This method employs DCH to extract features from the observed surfaces and generate a characterization, which is then combined with a supervised learning process to transfer the experience from previous grasp attempts.

Chapter 7

Experimental Setup and Evaluation

Since this work focuses on the development of a data-driven grasp synthesis method, which in turn involves a process of supervised learning (as described in Chapter 6), then a natural requirement for its development is to have experimental data with information of grasping attempts. The existence of such data is required for 2 purposes: i) to perform the supervised learning process described as part of the proposed grasp synthesis method; and ii) to assess the ability of the learned relation function H (Equation 6.5) to successfully predict a suitable grasping strategy for a given target object.

Thus, in order to produce the required experimental data, diverse experiments were performed in a simulated environment, using the Gazebo simulator and a simulated instance of the PR2 robot (detailed in Section 2.2.2 and Section 2.3, respectively).

This chapter describes the process performed to generate and collect the experimental data, and also the assessment process of the proposed grasp synthesis method.

7.1 Experimental Data Generation

7.1.1 Setup

In real life, when a robot is faced with a grasping problem, it has to deal not only with the generation of a proper grasping strategy but also with perceptual, geometric and motor problems, among others. As mentioned in Section 2.1.1, a real life grasping problem usually involves one or more steps of sensory data processing and perception, where the target object is extracted from the surrounding environment (process known as segmentation) and the relevant information is collected from the sensory data (process known as feature extraction).

Since the goal of this work is to develop a grasp synthesis method, all the experiments were performed using a simplified version of a real grasping problem. In this case, such simplification is done only in terms of the number of objects present in the experimental

environment. The goal of this is to reduce the complexity of the sensory data collected by the robot, which in turn reduces the complexity of the segmentation problem.

With this purpose, the setup used for all the experiments included only 3 objects (Figure 7.1):

i) The PR2 robot.
ii) The target object.
iii) A table to hold the target object at an appropriate height.

The goal of this setup is to simplify the segmentation process at a point where it can be performed by just simply trimming the sensory data captured by the robot (a point-cloud) to remove the floor and the table, leaving only the points which correspond to the sensory representation of the target object. Nevertheless, if an adequate perceptual method is provided, the proposed grasp synthesis method could be directly used in more complex environments, without making any type of simplification. Since such a problem is not the focus this thesis, it will not be further discussed.

7.1.2 Data Generation Procedure

Broadly speaking, the generation and collection of experimental data involve the execution of stages 1, 2 and 3 of the proposed grasp synthesis method (Figure 6.1). Nevertheless, the following descriptions assume that Stage 1 (Feature Space Reduction) was previously completed, since it can be done without using a simulated environment, as described in Section 6.2.1. Therefore, this process can be limited to the execution of Stage 2 and the first part of Stage 3 (Data Capture), since both require the simulated environment and robot to be performed.

This process starts in Stage 2 of the proposed grasp synthesis method, with the capture of a point-cloud and the generation of grasp candidates, and then continues with the execution of the first part of Stage 3, where multiple grasp attempts are carried out. The experimental data produced by this process corresponds to the data collected from each grasp attempt. As stated in Section 6.2.3, such data is:

- The grasp candidate used, $g_{\text{i}} = (p_j, \theta_k)$.
- The descriptor vector $\vec{\text{d}}_{DCH}$ associated with g_{i}.
- The result of the grasp attempt.

As described in Section 6.2.2, a set of angles for the orientation of the effector is required to generate the grasp candidates to perform the grasping attempts. In this case, 2 different sets of angles were used for this, with the purpose of producing 2 different sets of experimental data (Figure 7.3). The first set contained 4 angles, $\Theta_1 = \{0, \pi/4, \pi/2, 3\pi/4\}$, while the second set contained 5 angles, $\Theta_2 = \{0, \pi/5, 2\pi/5, 3\pi/5, 4\pi/5\}$.

Note that the angles in both sets are limited to the range $[0, \pi]$. This is because the effector of the PR2 robot is symmetric for rotations around the X_{eff} axis (Figure 5.1), as a

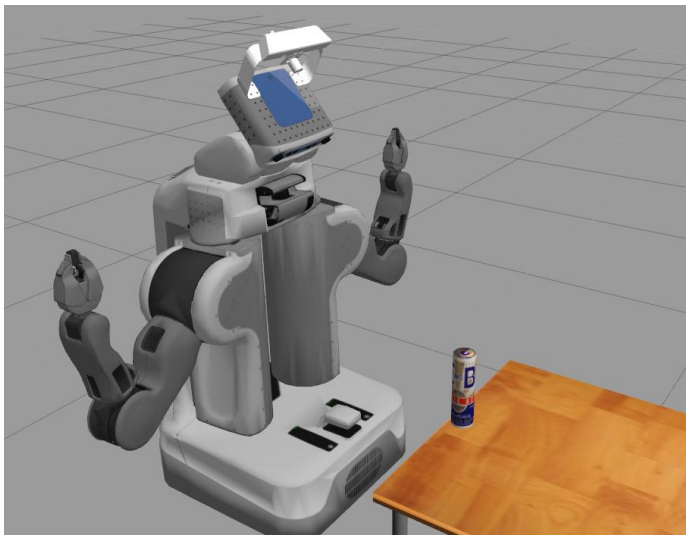
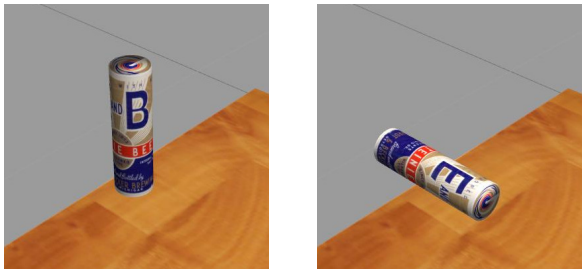
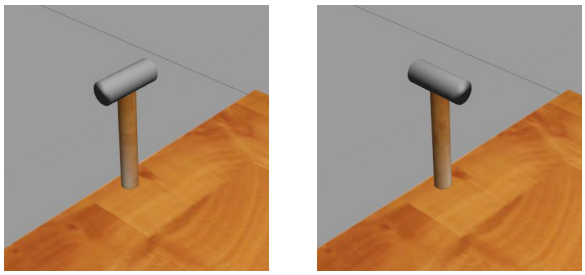


Figure 7.1: *Example setup used to perform all the experiments.*



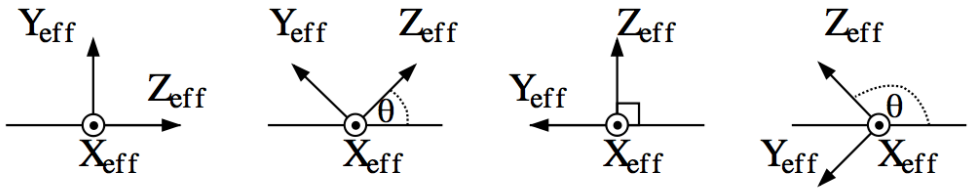
(a) *A can, arranged in two different positions.*



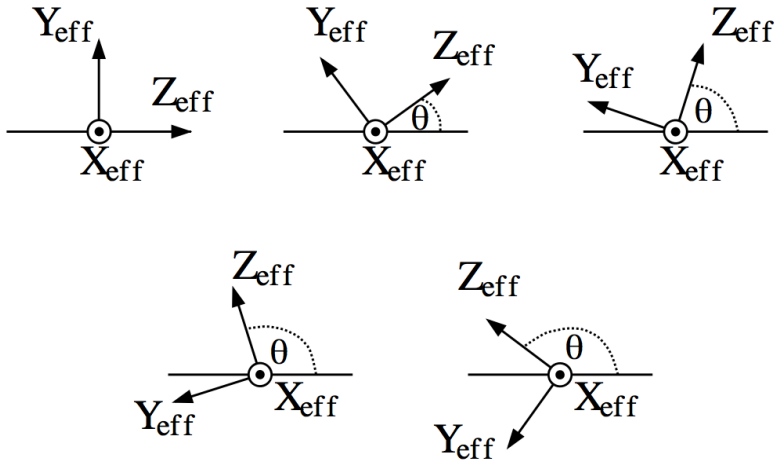
(b) *A hammer, arranged in two different positions.*

(c) *A cordless drill, arranged in three different positions.*

Figure 7.2: *Objects in different positions, used for experimental data generation.*



(a) *Set Θ_1, used for experimental data generation.*



(b) *Set Θ_2, used for experimental data generation.*

Figure 7.3: *Sets of angles used for the experimental data generation.*

consequence, the range $[0, \pi]$ produces the same angles as the range $[\pi, 2\pi]$.

Additionally, with the purpose of producing experimental data showing enough diversity, this process was performed using the described experimental setup, but with 3 different target objects. Each object was arranged in several positions (as shown in Figure 7.2), producing a total of 7 different experimental setups:

- A can, in 2 different positions.
- A hammer, in 2 different positions.
- A cordless drill, in 3 different positions.

This way, 2 sets of experimental data were collected, using these 7 experimental setups together with the 2 sets of angles, Θ_1 and Θ_2, previously described. A total of 231 grasp attempts were performed in the 7 experimental setups, which corresponds to the number of blobs defining similar zones in the experimental setups. The results of each attempt were recorded, identifying 3 outcomes:

i) Grasp attempt successful (object grasped).
ii) Grasp attempt failed (object not grasped).
iii) Grasp attempt unfeasible (the robot is incapable of performing the grasp).

Table 7.1: *Summary of the generated data.*

Angles	Successful	Failed	Unfeasible	Total
4	33	29	34	96
5	31	45	59	135
Total	64	74	93	231

Table 7.2: *Details of the generated data.*

	Angles	Successful	Failed	Unfeasible	Total
Can	4	5	14	1	20
	5	4	15	6	25
Drill	4	8	5	31	44
	5	9	13	48	70
Hammer	4	20	10	2	32
	5	18	17	5	40
Total	-	64	74	93	231

Unfeasible grasps exists due to the nature of the grasp synthesis method (data-driven), which does not take into account the physical limitations of either the robot or the setup.

Table 7.1 presents a summary of the results of the grasp attempts, while Table 7.2 lists the results grouped by target object.

7.2 Learning Process and Assessment

The experimental data, generated through the procedure previously described was used to carry out the second part of Stage 3 of the proposed grasp synthesis method (Section 6.2.3). This part corresponds to a supervised learning process, where a classifier is trained to learn a relation H between the descriptor vector associated with each grasp candidate, $\vec{d}_{DCH}$, and the result of each grasp attempt.

7.2.1 Learning

The training procedure was performed using only the data collected from the grasp attempts marked as "successful" and "failed". The attempts marked as "unfeasible" were discarded, since they were not executed and therefore do not have valid information for the learning process. The remaining data was divided into two subsets: i) a subset T_0, for training and validation, and ii) a subset A_0, for assessment of the trained classifier.

The subset T_0 was formed by data collected from 2 of the 3 objects available in the experimental setup, more specifically from the setups containing a can (Figure 7.2a) and a cordless drill (Figure 7.2c) as target objects. Similarly, the subset A_0 was formed by data collected from the remaining third object present in the experimental setup, that is, the setup containing a hammer (Figure 7.2b) as the target object. The goal is to perform the learning process using a set of target objects completely different from the ones used for the assessment process, in order to evaluate if the grasping experience obtained from one object can be applied to learn how to grasp a different object.

Finally, to perform the training process the set T_0 was divided into 2 subsets, T_1 and T_2, each one containing the experimental data collected from the grasping attempts performed using the sets of angles Θ_1 and Θ_2, respectively.

$$\begin{aligned} T_1 &= \{t = grasp\,(g_{\text{i}}) \in T_0 \mid g_{\text{i}} = (p_j, \theta_k) \ \wedge \ \theta_k \in \Theta_1\} \\ T_2 &= \{t = grasp\,(g_{\text{i}}) \in T_0 \mid g_{\text{i}} = (p_j, \theta_k) \ \wedge \ \theta_k \in \Theta_2\} \\ T_0 &= T_1 \ \cup \ T_2 \end{aligned} \tag{7.1}$$

Then, a set of 3 classifiers, $\Gamma = \{H_0, H_1, H_2\}$, were trained using the datasets T_0, T_1 and T_2 respectively. The goal of this is to generate a set of classifiers trained using different angles for the orientation of the effector, in order to evaluate how such a variable affects the performance of the learned relation.

Classifier Training

As stated in Section 6.2.3, the classifier used for the experimental evaluation was SVM. The particular incarnation of the classifier used was a C-SVM with a RBF kernel. The training procedure for all the classifiers was performed following a K-Fold cross-validation method (Section 2.1.7) with $K = 10$ and using the "auto-training" functionality provided by OpenCV [16]. Such a functionality trains the selected type of SVM automatically by choosing the optimal values for parameters C and γ. The values of the parameters are optimized by sweeping a range of possible values, those are said to be optimal when the error of the cross-validation process is minimal.

7.2.2 Assessment

The assessment process focused on 2 points:

i) To evaluate the ability of the classifier to predict the outcome of a grasp candidate.
ii) To evaluate the ability of the classifier to identify feasible angles for the orientation of the effector, even when such angles have not been part of the training process.

Clearly, the goal of the first point is simply to determine if the trained classifier can accurately predict the outcome of a grasp candidate, that is, if it will produce a successful grasp or not. Such an ability is key for the exploitation of the knowledge acquired.

The goal of the second point is to determine if a classifier can be used to define ranges of angles for the orientation of the effector, which may produce successful grasps. The focus is to assess the capability of the classifier to define such ranges from isolated points. Note that the benefit of determining such ranges resides in the fact that the set of orientations used in the training step may differ from the ones used during the exploitation step. Such a scenario could occur either because the grasp synthesis method might benefit from using a larger number of possible orientations, or because the training step might be too expensive to be performed with a large set of orientations, consequently using fewer angles than expected for regular operation.

In order to assess the 2 mentioned points, the evaluation set A_0 was divided into two parts, A_1 and A_2, following the same logic used to divide the training set T_0 (Equation 7.1).

$$\begin{aligned} A_1 &= \{a = grasp\,(g_{\mathrm{i}}) \in A_0 \mid g_{\mathrm{i}} = (p_j, \theta_k) \ \wedge \ \theta_k \in \Theta_1\} \\ A_2 &= \{a = grasp\,(g_{\mathrm{i}}) \in A_0 \mid g_{\mathrm{i}} = (p_j, \theta_k) \ \wedge \ \theta_k \in \Theta_2\} \\ A_0 &= A_1 \ \cup \ A_2 \end{aligned} \tag{7.2}$$

Then, 5 different experiments were performed using such datasets:

Exp. I H_0 evaluated using data from the set A_0.
Exp. II H_1 evaluated using data from the set A_1.
Exp. III H_2 evaluated using data from the set A_2.
Exp. IV H_1 evaluated using data from the set A_2.
Exp. V H_2 evaluated using data from the set A_1.

With the purpose of comparing the effectiveness of DCH to aid in resolving the grasping problem, these experiments were also carried out with some 3D local descriptors existing in the literature, that is, a set of classifiers were trained using different descriptors and then evaluated using the logic previously presented. The descriptors considered for this comparison were Spin Images, SHOT and FPFH.

7.3 Synthetic and Real-life Sensory Data

As stated before, the experimental data required to evaluate the developed method was generated using a simulated environment, provided by the Gazebo simulator. However, despite being a realistic simulator which runs the actual software of the simulated robots, Gazebo necessarily has to synthetically produce sensory data, in order to complete the simulation process.

Note that the use of Gazebo is likely to suppress or remove any complexity that may be the result of the presence of noise and/or perturbations in the sensory data, also hiding any sensitivity of DCH to such effects. This issue is especially delicate for the surface normals, which are a keystone element for the computation of DCH, and whose computation is particularly sensitive to noise.

Generally speaking, the methods used to estimate the normal vector to a surface involve fitting several planes tangent to the surface of the object at different points in order to obtain the normal direction, and then disambiguate the exact orientation of the normal vector using a particular algorithm or criterion.

In the case of DCH, the normal estimation was performed using the method proposed by Rusu [58], which is based on fitting a tangent plane by means of a Least Squares estimation, and then disambiguating the orientation using Principal Component Analysis. As previously explained, this method makes DCH susceptible to noise. However, to provide the descriptor with robustness against noise and perturbations a filtering step was added prior to the computation of the normals. Such a step was carried out by performing a Gaussian filtering on the captured point-cloud, in order to produce a smoothing effect which reduces the influence of noise in the position of the points, and in turn, reduces the variability in the orientation of the estimated normals. This filtering is done by performing a convolution of the captured point-cloud with a Gaussian kernel.

$$G(\vec{x}, \sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^N} \mathrm{e}^{\frac{\|\vec{x}\|^2}{2\sigma^2}} \tag{7.3}$$

The effect of the filtering step is easily noticeable in the normals computed on a point-cloud. Figure 7.4 shows a comparison between the results of computing the surface normals on different clouds.

Figure 7.4a shows a raw point-cloud, obtained from the RGB-D Object Dataset (described in Section 6.2.1) and Figure 7.4c shows a filtered point-cloud, produced by applying the filtering step on the point-cloud presented in Figure 7.4a. Figures 7.4b and 7.4d show the computed normals on each cloud, respectively. Notice the positive effect on the orientation of the normal vectors, produced by the filtering step.

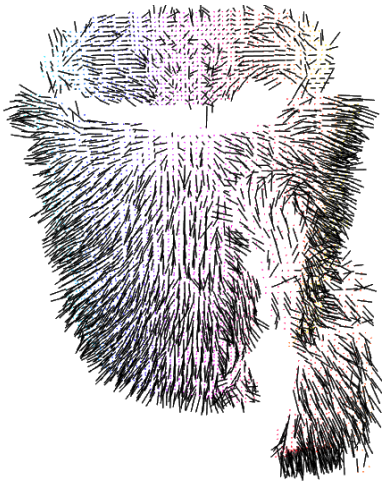
Additionally, Figure 7.4e shows a synthetic point-cloud depicting a cylinder, while Figure 7.4f presents the respective normals computed using that cloud. Observe the similarities in the computed normals between Figure 7.4d and Figure 7.4f, as well as the differences with respect to the normals computed on the raw point-cloud (Figure 7.4b)

7.4 Summary

This chapter presented the details of the experimental setup used to produce the data required to perform the supervised learning process, which is part of the proposed grasp synthesis method. Next this chapter presented a description of the training and assessment procedures carried out to produce and evaluate the classifier used to select suitable grasping candidates.

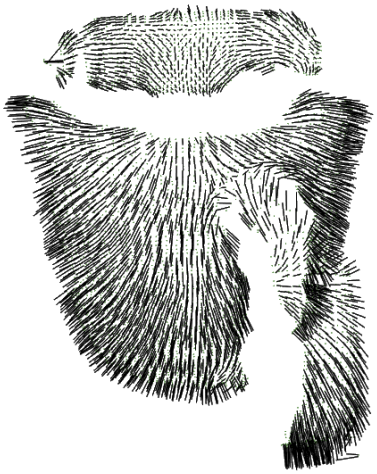
In addition, this chapter presented a discussion on the difference between real and synthetic data, as well as a simple method to reduce the effects of noise and perturbations present in real sensory data. Such a method is used in order to achieve a comparable behavior for the estimations of the surface normals between real and synthetic data.

(a) *Raw point-cloud.*

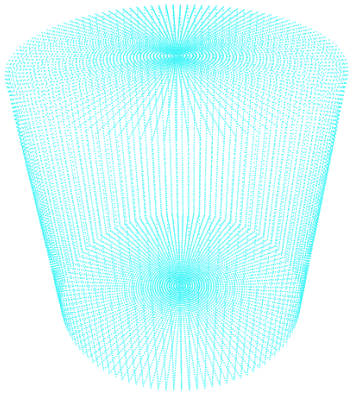


(b) *Normals computed on the raw cloud.*

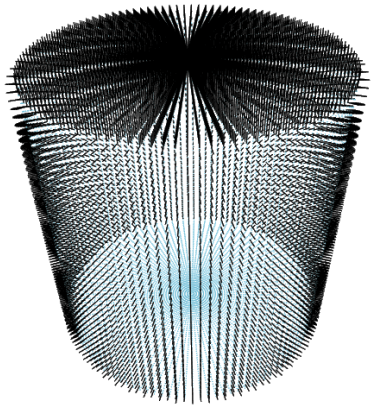
(c) *Filtered point-cloud*



(d) *Normals computed on the filtered cloud.*



(e) *Synthetic point-cloud*



(f) *Normals computed on the synthetic point-cloud.*

Figure 7.4: *Computed normals on raw and filtered point-clouds.*

Chapter 8

Experiments and Results

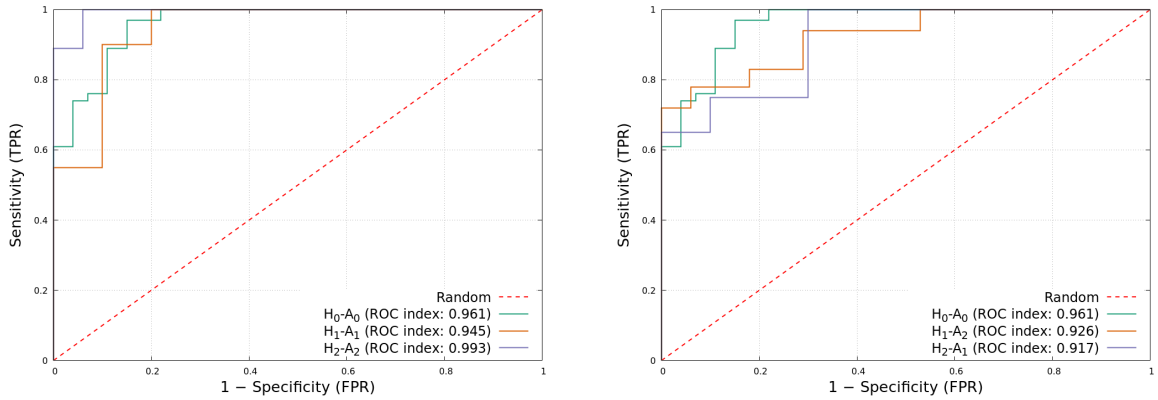
The ability of the proposed grasp synthesis method to produce successful grasps was evaluated by several experiments, which were performed following the procedures and using the experimental setup described in Chapter 7. These experiments were focused on assessing the ability of the trained classifier to predict the outcome of a grasp candidate, its sensitivity to changes in the parameters of DCH, and the ability of the proposed method to produce successful grasps.

8.1 Predictive Performance

8.1.1 Classification Performance Using DCH

The results of the performance evaluation of the classifier, trained using DCH descriptors, are presented in Figure 8.1. Here two ROC curves (Section 2.1.8) show the ability of the classifier to produce correct predictions on the outcome of a grasp candidate. In detail, Figure 8.1a shows the classification performance assessed using a dataset with the same number of orientations for the effector that the dataset used for training, which corresponds to Experiment II and Experiment III (presented in Section 7.2.2). Figure 8.1b shows the classification performance assessed using a dataset with a different number of orientations for the effector that the dataset used for training, that is, Experiment IV and Experiment V. Both figures also include, for comparison purposes, a ROC curve showing the classification performance assessed using datasets which contained all the tested orientations for the effector, as well as for the training process as for the evaluation process.

In both cases, the behavior observed in the ROC curve displays a good classification performance, therefore, the classifiers are able to correctly predict the outcome of a grasp candidate in general terms. As expected, the performance observed in the evaluation carried out using datasets with the same number of orientations, was better than the performance observed in the evaluation carried out with a dataset with a different number of orientations.



(a) *Classification performance with matching orientations for training and evaluation.*

(b) *Classification performance using different orientations for training and evaluation.*

Figure 8.1: *Classification performance using DCH descriptor.*

Regarding the experiments using a different number of angles for the orientation of the effector (Figure 8.1b), the good performance observed indicates that the classifier is capable of correctly predicting the outcome of a grasp candidate, even when an orientation not used in the training step is used.

In real life, the orientation of the effector which may produce a successful grasp is not an isolated point, but a range of valid values. This range could be small or large, depending on the grasp candidate and the object. The performance showed by the classifier during Experiment IV and Experiment V is an indicator of the ability of the classifier to identify such a range of valid orientations.

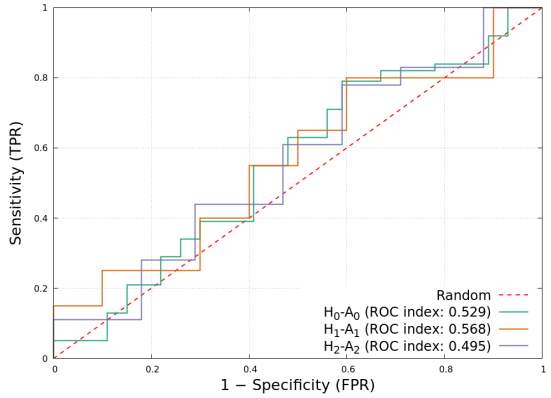
The classifier trained and evaluated using all the tested orientations for the effector (Experiment I) also displayed a good performance. In comparison to the results of the classifiers trained with a matching number of orientations (Figure 8.1), the performance of this classifier is between better than the one trained using 5 orientations and slightly worse than the one trained using 4.

8.1.2 Classification Performance Using Other Descriptors

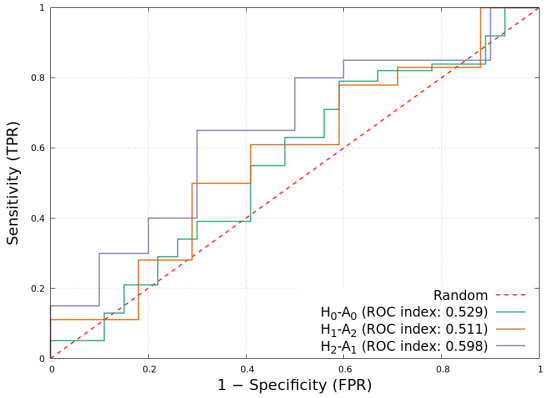
To compare the results obtained by the classifiers trained using DCH, the same experiments were carried out using different descriptors, as stated in Section 7.2.2.

In the same way that the results were presented in the previous section, Figures 8.2, 8.3 and 8.4 display a set of ROC curves showing the classification performance reached using Spin Images, SHOT and FPFH, respectively.

The classification performance observed using these descriptors was notoriously worse than the one observed for DCH. This can be explained by the fact that all the tested descriptors presented rotational invariance, which made the description incapable of encoding the differences between the rotations. In comparison, the classifier trained using Spin Images showed

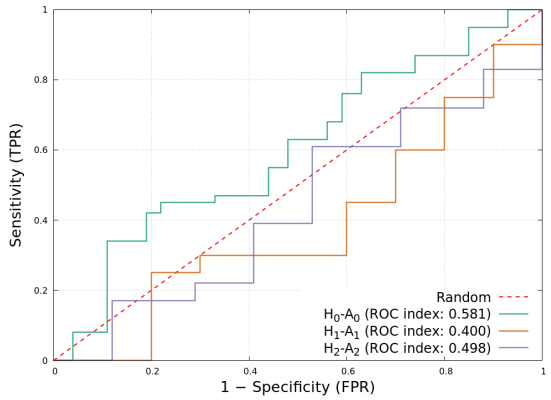


(a) *Classification performance with matching orientations for training and evaluation.*

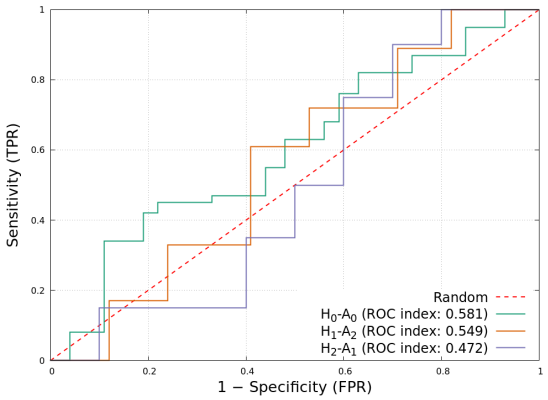


(b) *Classification performance using different orientations for training and evaluation.*

Figure 8.2: *Classification performance using SHOT descriptor.*

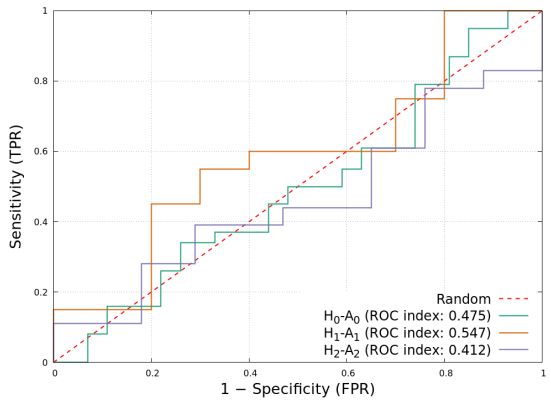


(a) *Classification performance with matching orientations for training and evaluation.*

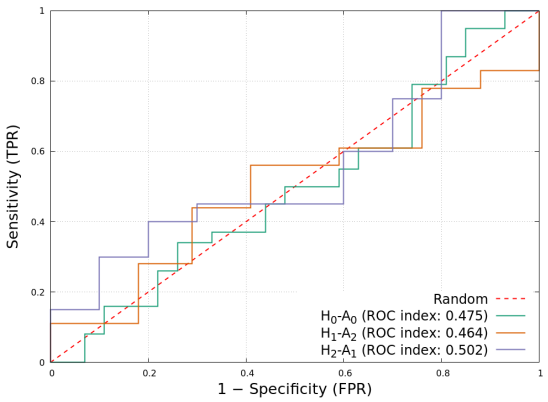


(b) *Classification performance using different orientations for training and evaluation.*

Figure 8.3: *Classification performance using Spin Images descriptor.*



(a) *Classification performance with matching orientations for training and evaluation.*



(b) *Classification performance using different orientations for training and evaluation.*

Figure 8.4: *Classification performance using FPFH descriptor.*

the lowest performance, while the one trained using SHOT showed slightly better results.

The particularly bad performance obtained with Spin Images can be explained by taking into account that this descriptor projects the vicinity of the target onto a local coordinate system. If the sensory data contained in such vicinity do not include enough information or do not have the required extent, its projection onto the local coordinate system might generate a spin image with too few features to provide a useful descriptor. This problem can be approached by increasing the size of the vicinity of the target point, which in turn will increase the amount of data used to compute the descriptor. However, this is likely to have a negative effect on the computation time of the descriptor.

8.1.3 Grasping Performance Using DCH

The same set of experiments assessing the classification performance were carried out in a simulated environment in order to evaluate the grasping performance of the developed grasp synthesis method. Figure 8.6 shows the results of this evaluation, detailing the performance observed using the same and a different number of orientations for the effector as the ones used in the training dataset.

For these experiments, two additional objects (Figure 8.5) were added to the experimental setups used for evaluation. These objects were not used in any previous training step.

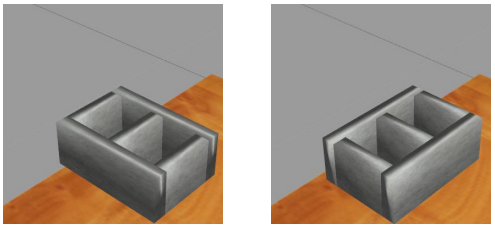
In this case, the ROC curves show a good grasping performance for both experiments. However, there is an obvious drop in performance in comparison to the classification experiments presented in Section 8.1.1.

After reviewing these results, it can be noticed that most of the failed grasp attempts are the product of issues in the alignment of the effector. These problems can be explained by the joint action of two conditions:

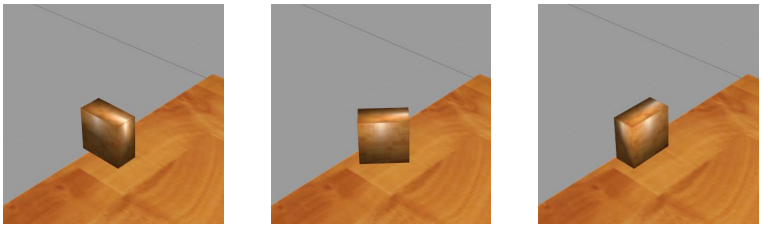
i) Gazebo simulates the behavior of the PR2 robot by running the same software used by the robot in real life. Such software makes intensive use of polling and asynchronous communication to monitor the state of sensors and devices, as well as to control the execution of actions. This makes the control system flexible and robust to communication problems. However, at the same time, this makes each execution sensitive to the frequency of the status polling and to delays in the communication. As a consequence, different executions of the same command are likely to produce slightly different outcomes. For example, a command issuing the movement of an arm to a particular position on two different executions will produce two final positions which are similar, but not the same.

ii) The training process intrinsically takes into account the uncertainties derived from the sensor polling mechanism and sensory data. However, the drop in performance could be an indicator of the inability of the training to produce a better generalization to new objects and/or to the variability of the sensory data.

    This effect may be boosted by the number of orientations used during the training step. When a small number of orientations is used, there is more pressure on the

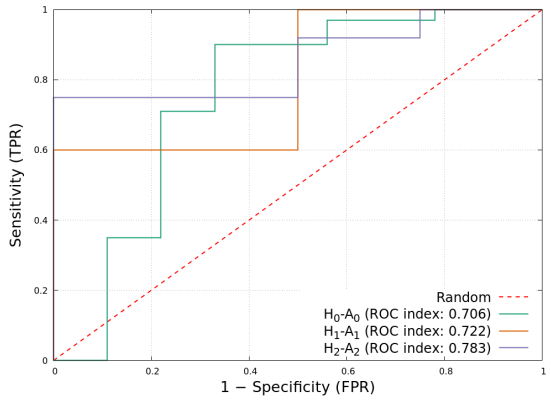


(a) *A cinder block, arranged in two different positions.*

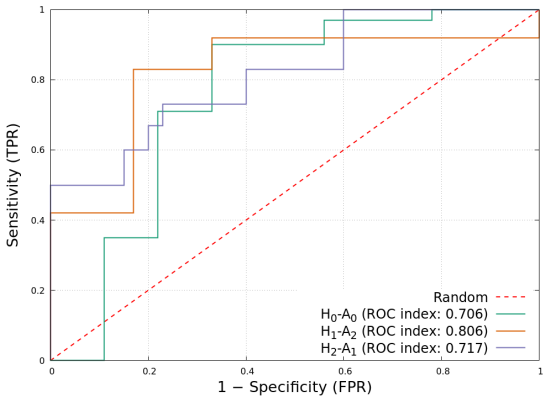


(b) *A wooden block, arranged in three different positions.*

Figure 8.5: *Additional experimental objects.*

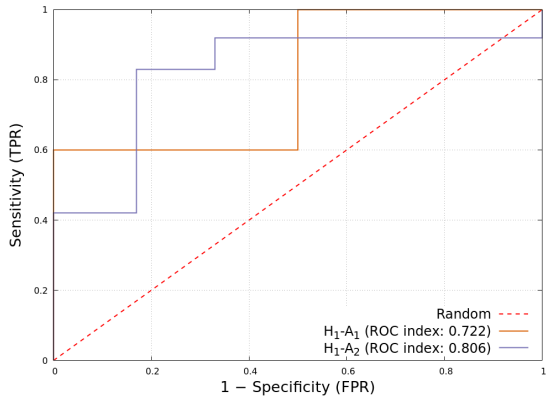


(a) *Grasping performance with matching orientations for training and evaluation.*

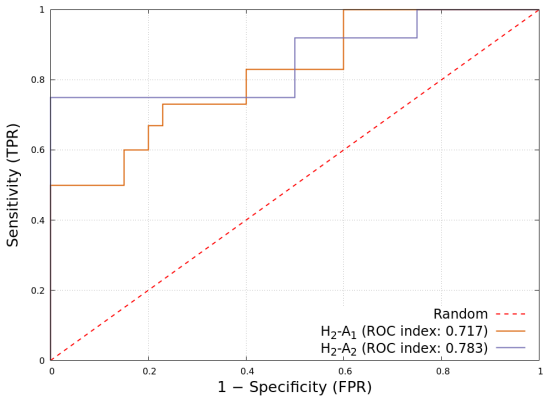


(b) *Grasping performance with different orientations for training and evaluation.*

Figure 8.6: *Grasping performance using DCH.*



(a) *Performance with 4 orientations.*



(b) *Performance with 5 orientations.*

Figure 8.7: *Grasping performance using DCH, by number of orientations.*

classifier when it has to predict the outcome for new orientations. On the other hand, when a larger number of orientations is used for training more samples of the space of possible orientations are considered, therefore producing a classifier capable of learning a better definition of the orientations which generate successful grasps. This remark can be partially observed in Figure 8.7, where the results of the classifier trained using 5 orientations (Figure 8.7b) show a more stable a behavior than the results produced by the classifier trained using 4 orientations (Figure 8.7a).

It is interesting to note that, despite the drop in performance observed, the classifier is still capable of displaying good prediction abilities when a different number of orientations are used in the training and evaluation process (Figure 8.6b).

8.2 DCH Sensitivity Analysis

In order to evaluate the impact of a change in the parameters of DCH on the performance of the developed grasp synthesis method, a sensitivity analysis was performed, including 2 of the 4 parameters described in Section 5.2.1. The parameters included were the number of bands present in the descriptor, N, and the width of each band, w. The remaining parameters were excluded from this analysis for two reasons:

i) The size of the vicinity, r, is directly related to the maximum opening of the effector. Therefore, its value is defined depending on the features of the robot, and in consequence, it is not a parameter which would be actively modified.

ii) The angle of the first band, β, was excluded because its value is fixed during the training process and equal to the orientation of the effector, as described in Section 6.1.2.

With the purpose of avoiding the need to include an extra variable and to simplify the analysis, this was carried out considering only the classifier trained using 4 orientations for the effector.

For each analysis, the value of the parameters were the ones discussed in Section 6.1.2 (with the exception of the parameter under study), that is:

- Number of bands $N = 8$.
- Size of the vicinity $r = 3$ [cm].
- Width of the bands $w = 0.5$ [cm].
- Angle of the first band, β, was fixed and equal to the orientation of the effector.

An additional analysis of the sensitivity to size of the bin used in the definition of the histograms of DCH was also included.

8.2.1 Sensitivity to the Number of Bands

To assess the sensitivity of the classification performance to changes in the number of bands in DCH, the training step of the classifier was repeated several times. Each repetition consisted in training the classifier using instances of DCH computed with different parameters, that is, to repeat the process described in Section 7.1.2.

A total of 11 experiments were carried out, using the following values for N:

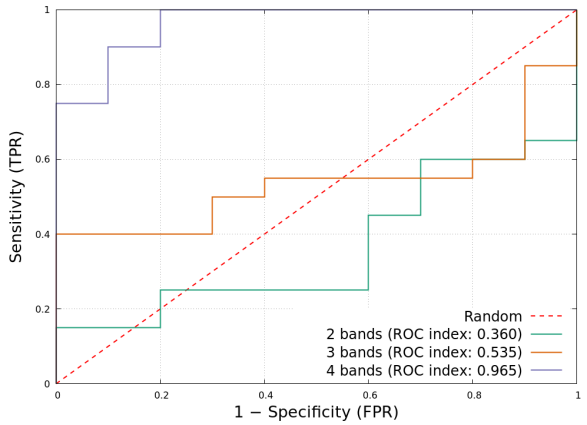
i) $N = 2$
ii) $N = 3$
iii) $N = 4$
iv) $N = 5$
v) $N = 6$
vi) $N = 7$
vii) $N = 8$
viii) $N = 15$
ix) $N = 25$
x) $N = 50$
xi) $N = 100$

The results of the sensitivity analysis on the number of bands of the descriptor, N, are shown in Figure 8.8.

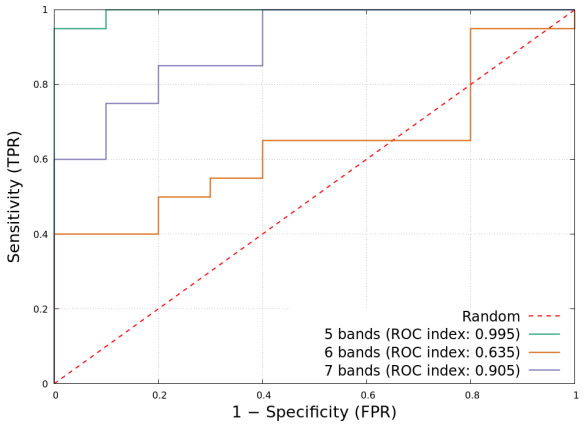
In general terms, the results show that the performance of the trained classifier improves as the number of bands increases, showing an asymptotic-like behavior. The addition of the initial bands strongly increases the performance, while the latter additions show a limited effect.

Figure 8.8a shows a dramatic increase in performance as the number of bands rises from 2 to 4. This is explained by the fact that a version of DCH with only 2 bands will have them placed at angles 0 and π, which means that the descriptor will be able to only extract information along a single *line* going forward and backward. This configuration clearly lacks enough descriptive power to encode the information surrounding the target point, which explains the bad performance. The results observed for 3 and 4 bands confirm such an explanation. In particular, the performance observed with 4 bands is far better than with 2 or 3 bands, which is explained by the fact that 4 bands would be placed every $\pi/2$ radians, easily covering the 4 main axes around the target point and, consequently, greatly increasing the descriptive power of DCH.

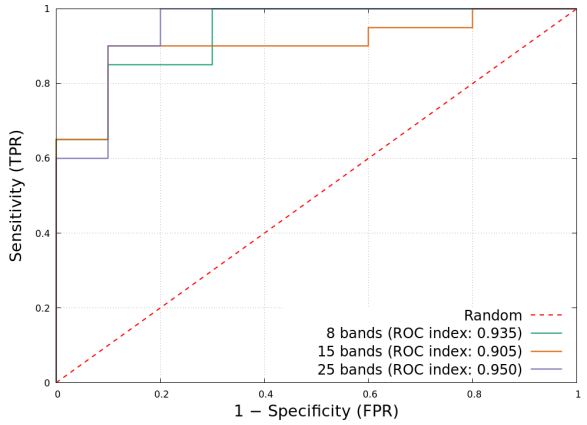
For higher number of bands (Figure 8.8c and Figure 8.8d) the performance shown does not strongly change. This is caused by two simultaneous conditions: i) the increasing coverage of the vicinity data, and ii) the increasing overlap of the bands in the descriptor. As the number of bands increases, the part of the vicinity which is *covered* by bands increases, and therefore, the part of the vicinity which is used for the computation of DCH. Simultaneously, as the number of bands increases, the overlap between bands also increases. This, in turn, increases the amount of data which is used more than once for the computation of DCH (each point is present in several bands simultaneously). In consequence, as the number of bands increases, the description process reaches a point where it does not add any more new information, and therefore, the performance neither highly increases nor highly decreases.



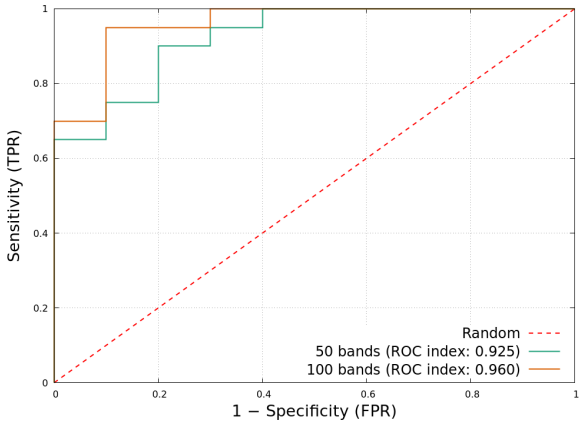
(a) *Performance using DCH with 2, 3 and 4 bands.*



(b) *Performance using DCH with 5, 6 and 7 bands.*



(c) *Performance using DCH with 8, 15 and 25 bands.*



(d) *Performance using DCH with 50 and 100 bands.*

Figure 8.8: *Performance evolution according to the number of bands used in DCH.*

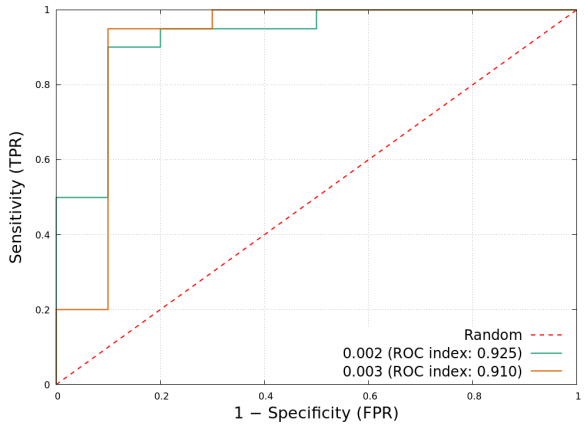
8.2.2 Sensitivity to the Width of the Bands

Similarly to the previous section, to evaluate the sensitivity of the classification performance to changes in the width of the bands of DCH, the training step of the classifier was repeated several times. During each repetition, the classifier was trained using instances of DCH computed with different parameters for the width of the bands, w.

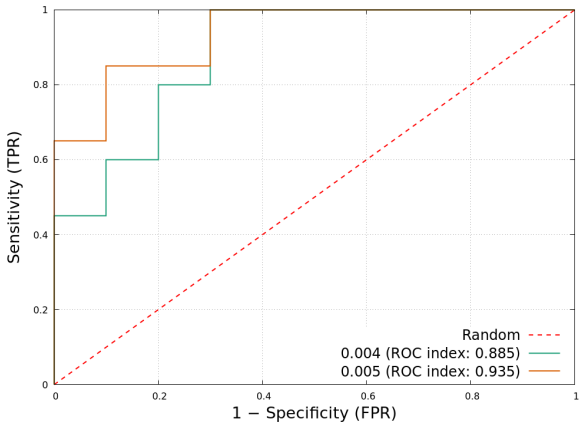
In this case, given the conditions of the experiment which determine the density of points of the point-cloud (distance of the sensor to the object, orientation of the sensor, etc.), the range of values for the width covered by this analysis was defined around the parameters of regular operation of the system (Section 6.1.2). A total of 8 experiments were carried out using values ranging from $w = 0.2$ [cm] to $w = 2$ [cm]. These are considered small and big values, given the fact that the size of the vicinity was defined as $r = 3$ [cm]. The values included in this analysis were:

i) $w = 0.2$ [cm]
ii) $w = 0.3$ [cm]
iii) $w = 0.4$ [cm]
iv) $w = 0.5$ [cm]
v) $w = 0.75$ [cm]
vi) $w = 1.0$ [cm]
vii) $w = 1.5$ [cm]
viii) $w = 2.0$ [cm]

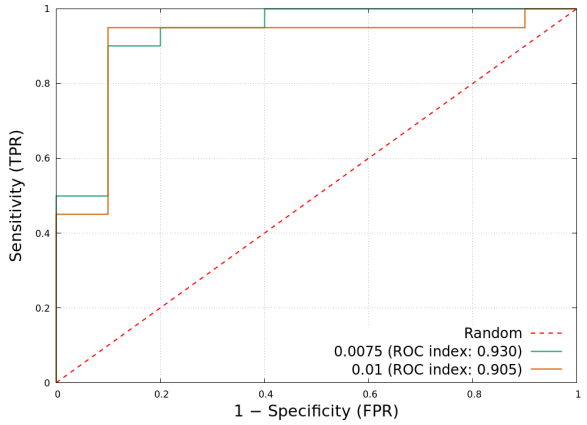
Figure 8.9 presents the results of the sensitivity analysis performed of the width of the bands, w.



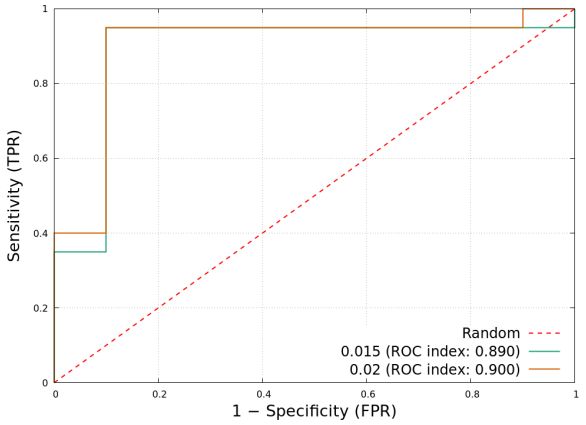
(a) *Performance using DCH with bands of width $w = 0.2$ [cm] and $w = 0.3$ [cm].*



(b) *Performance using DCH with bands of width $w = 0.4$ [cm] and $w = 0.5$ [cm].*



(c) *Performance using DCH with bands of width $w = 0.75$ [cm] and $w = 1$ [cm].*



(d) *Performance using DCH with bands of width $w = 1.5$ [cm] and $w = 2$ [cm].*

Figure 8.9: *Performance evolution according to the width of the bands of DCH.*

Generally speaking, the results of this analysis show that as the width of each band increase, the performance of the classifier tends to improve until an optimal value is reached, after which the performance slightly drops. Figure 8.9c and Figure 8.9d reveal that when the width increases too much, it actually worsens the results. This observation can be explained by 2 effects.

First, if the width of the band is small, the number of points present in each band will be small and there will be little information encoded in each of them. This makes the descriptor

susceptible to noise and perturbations because each band contains too few points available to compute the histograms of curvature, making them not representative of the real curvature of the surface. As the band increases its size, the number of points contained in each one increases, which makes the computation of the histograms more accurate.

The second effect is observed when the width of the band increases too much. When this happens, each band no longer displays a *directed* nature in the data contained by it. Despite that more data implies more descriptive power, the lack of directionality reduces the performance because contiguous bands become similar, making it harder for the learning algorithm to effectively identify the differences between multiple rotations of the descriptor.

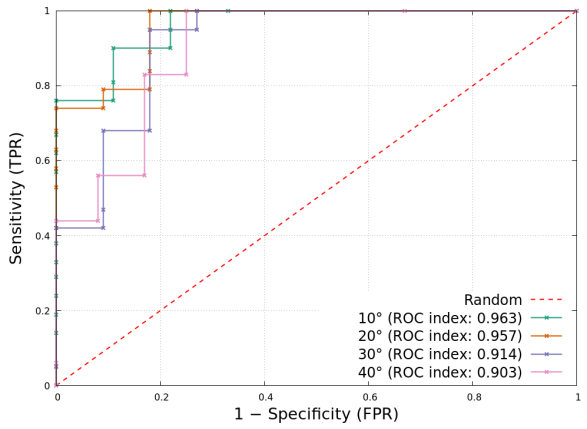
8.2.3 Sensitivity to the Size of the Bins

In the same way that the two previous sensitivity analysis, this was performed repeating the training step several times, using different sizes for the bins of the histograms during each one.

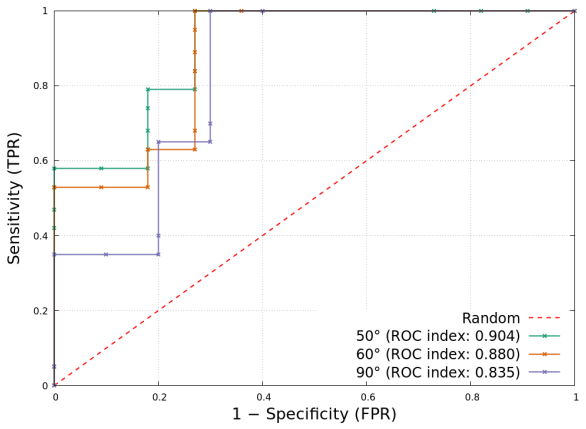
As stated in Section 5.2, the angular range represented by each histogram is restricted to $[-\pi/2, \pi/2]$. This same range was covered by this analysis, starting with a bin size equal to $\pi/18$ (10°), which corresponds to the bin size used in all the experiments; to a bin size equal to $\pi/2$ (90°), which is equivalent to have a single bin per histogram. The values included in this analysis were:

i) bin size = $\pi/18$ (10°).
ii) bin size = $\pi/9$ (20°).
iii) bin size = $\pi/6$ (30°).
iv) bin size = $2\pi/9$ (40°).
v) bin size = $5\pi/18$ (50°).
vi) bin size = $\pi/3$ (60°).
vii) bin size = $\pi/2$ (90°).

Figure 8.10 presents the results of this analysis.



(a) *Performance using DCH with bins of sizes* 10°, 20°, 30° *and* 40°.



(b) *Performance using DCH with bins of sizes* 50°, 60° *and* 90°.

Figure 8.10: *Performance evolution according to the size of the bins of DCH.*

The results show a clear tendency pointing that as the size of the bins increase, the performance of the proposed method decreases. In general, this observation can be explained by the fact that an increase in the size of the bins makes DCH more robust to noise, but less descriptive. Therefore, a sustained increase in the size of the bins can lead to a very robust, yet very poor descriptor, negatively affecting the classification performance.

Figure 8.10a shows that a small increase in the size of the bins does not have a mayor impact in the performance of the grasping method, as can be observed in the ROC curves for 10° and 20°. However, as the size of the bin increases towards bigger values, the performance significantly drops (Figure 8.10b).

Is interesting to note that, despite that the size of the bins have a marked effect on the performance, it is never as strong as the effect observed in the sensitivity analysis for the number of bands.

8.3 Summary

This chapter presented the results obtained in the experiments carried out with the purpose of evaluating the ability of the developed grasp synthesis method to produce successful grasps. This was done through 2 experiments: i) an evaluation of the ability of the trained classifier to correctly predict the outcome of a grasp candidate, and ii) an evaluation of the grasping performance of a simulated robot using the proposed grasp synthesis method. The results showed that the proposed method was capable of correctly predicting the outcome for different grasp candidates, behavior which was later confirmed by the grasping performance observed in the simulations.

This chapter also presented the results of a sensitivity analysis which related the predicting ability of the trained classifier with the computation parameters of DCH.

Chapter 9

Conclusions

This chapter finishes this thesis by presenting the conclusions reached through the study of the presented results. With such a purpose, this chapter starts by making a brief overview of the work done. Then, a review of the goals stated for this thesis is presented, discussing the fulfillment of each. Next, a discussion of the findings and limitations of this research is presented. This chapter concludes with a set of ideas and proposals for any future work aiming to continue with this research.

9.1 Conclusions

As the use and integration of robotics in diverse aspects of day-to-day life continues, robots are more often than not required to operate autonomously and to be able to freely perform in any human environment. In order to be a truly autonomous machine, a robot must be able to perform multiple types of tasks, ranging from the perception and recognition of objects, places, and people, to navigation and displacement. However, most of the time it is almost compulsory to be able to interact with the objects present in any environment, eventually modifying them, depending on the needs of each task. As a consequence, the ability to grasp an arbitrary object plays an important role in the future of the development of robotics.

This thesis proposes a novel 3D local descriptor and a grasp synthesis method, both of which address the problem of producing strategies to grasp an object, known as the *robotic grasping problem*.

The proposed 3D local descriptor, Directed Curvature Histograms (DCH), uses the curvature of an object along different directions in order to describe the vicinity of the target point. Despite there being several approaches in the literature which use the curvature as a description feature (for example PFH, FPFH), these methods do not recognize the direction of the observed changes as a meaningful feature. DCH proposes a novel and simple strategy to incorporate such information.

The proposed grasp synthesis method presents a data-driven approach, which uses the

curvature description by DCH to identify the zones in an object which are suitable for grasping. Unlike the methods existing in the literature, which mainly focuses on characterizing the sensory data in order to characterize the shape of graspable zones, this directly attempts to exploit the empirical information which shows that humans tend to grasp the objects using their smallest dimension, employing DCH for such purposes.

Proposed Goals

The main goal of this thesis was to develop a data-driven grasp synthesis method using machine learning techniques to produce a set of grasps, according to the curvature observed along different directions.

A set of specific objectives, presented in Section 1.2.2, were proposed as seven concrete actions focusing on supporting the accomplishment of the main goal. The first and second objectives were the design and implementation of a 3D local descriptor which would be capable of encoding the curvature of a surface along different directions. These objectives were addressed and fulfilled through the development of the proposed 3D local descriptor, Directed Curvature Histograms (DCH).

The third objective was the generation of a reduced representation of the feature space implicitly defined by DCH. This was accomplished by following the approach of the BoVW model, to compute a codebook which was later used in the perceptual step to identify similar zones and produce a set of possible grasping points.

The fourth and fifth objectives were the collection of a set of experimental data (successful/unsuccessful grasp attempts) and the use of such data to carry out a supervised learning process, in order to produce a classifier capable of identifying the characteristics of the suitable grasping zones and the proper parameterizations for the effector. These objectives form the core of proposed grasp synthesis method. They were fulfilled by the process of experimental data collection and learning, described in Section 6.2.3.

The sixth and seventh objectives were the evaluation of the ability of the developed method to successfully identify suitable grasping zones and parameterizations for the effector, by means of using a simulated environment. These were accomplished through the assessment process, described in Chapter 7, which produced the results presented in Chapter 8.

Finally, through the accomplishment of all the proposed specific objectives, which lead to the developed and presented grasp synthesis method, the main goal of this work can be considered fulfilled.

Results and Findings

The results of the performance evaluation of the trained classifier showed that, in general terms, it was capable of correctly predicting the outcome of a grasp candidate, even for objects which were not used during the training phase. Therefore, the grasp synthesis method was

capable of predicting and selecting the proper grasp candidate to perform successful grasps, behavior which was observed in the results collected during the simulation experiments. In conclusion, the developed grasp synthesis method works and is capable of providing a robot with the ability to grasp arbitrary objects, even when these have not been previously encountered. These results also show that the information provided by DCH is useful to identify suitable grasping zones, as well as suitable parameterizations for the effector. In contrast, the descriptors existing in the literature were not helpful for the resolution of the grasping problem using the developed grasp synthesis method, showing poor performance. This particular result is mainly due to the rotation invariance, exhibited by all the tested descriptors.

The experiments also showed that the trained classifier was capable of generalizing the orientation of the effector, allowing the system to generate successful grasps even for orientations not included in the training step. This represents an advantage since it allows the robot to explore the space of orientations using more points than the ones used during the training, which in exchange can result in the generation of grasps using a finer angle resolution for the orientation of the effector.

Limitations

As in any work, there are some limitations in the work presented in this document.

One such limitation comes from the fact that the grasp synthesis method presented was designed to operate using effectors which are simple and act as a two-fingered gripper. This is because the method finds the orientation which properly aligns the opening between the fingers of the gripper, in order to grasp an object. If such an opening is not clearly defined, like it would be in the case of a three-fingered effector, it is not clear if the developed method would be capable of learning the proper grasping strategy, particularly, the proper orientation for the effector. Despite this limitation, grippers are a very common type of effector, hence this method can still be used in a wide range of setups.

A different limitation of the presented grasp synthesis approach stems from the fact that the computation of DCH requires the definition of a vicinity around the target point. If the size of the vicinity is too small, the amount of sensory information inside it might be too small, therefore making the curvature encoded by DCH unreliable or even useless. On the other hand, if the size of the vicinity is too big, the localness of the descriptor tends to get lost, which affects the identification of similar zones. Note that for the presented approach the size of the vicinity was fixed and its value was determined according to the physical characteristics of the robot, however, for a different situation this parameter might be changed.

Regarding the experimental setups used to train and evaluate the classifier, it can be pointed out that they represent a limitation since the presented results were obtained using a simulator, therefore using synthetic data. It is important to note the fact that Gazebo is a realistic simulator, hence the sensory data produced by it is in no sense simplistic or geometrically simpler than the sensory data that can be collected in a real setup. Therefore,

the main limitation stemming from the use of a simulated environment is the fact that the sensory data is affected neither by noise nor by any kind of perturbation observed in sensory data captured with a real sensor. Nevertheless, the existence of perturbations in sensory data was not ignored during the development of DCH. It is key to highlight that the computation process of the codebook used in the definition of grasping points was performed using real-life sensory data from the RGB-D Object Dataset, as stated in Section 6.2.1. As a consequence, the computation of DCH included a step for handling noise and perturbations, described in Section 7.3, leaving the proposed method prepared to operate with data from real sensors.

In a similar way, the fact that the setups used in the experiments were a simplified version of a real grasping scenario (because they contained only one object) can also be pointed out as a limitation of the proposed method. However as mentioned in Section 7.1.1, the only motivation for this was to simplify the resolution of the *scene segmentation* problem. Such a problem is by itself a completely different research topic, and it was not solved in a general way because it was not part of the goals of this work. Therefore regarding the complexity of the scene, the only requirement to use the proposed grasping method in a real-life scenario is to count with an algorithm capable of segmenting single objects in a complex scene. In the literature it is possible to find several algorithms suitable for this purpose, for example the approach proposed by Ückermann et al. [59], or by Rao et al. [60].

Finally, regarding the classifier used for predictions, the number of objects used in the training and evaluation processes could be pointed to as a limitation of this work. As described in Section 7.1.2, such processes were done using 3 objects in several different poses, generating 7 different experimental setups, which in turn produced a total of 138 grasp attempts. Here is important to stress the fact that, despite using a reduced number of objects for experimental evaluation, the complexity of each experiment was not different from the complexity of any real-life grasp situation. Thus, the only limitation might be the ability of the trained classifier to correctly predict the outcome of a grasp candidate for a wide scope of situations and objects; however, this does not restrict or invalidate the applicability of the developed method. An extended evaluation using a bigger set of objects should be considered for any following work.

9.2 Future Work

The most direct tasks that could be done in order to continue the development of this work is to approach the limitations described in the previous section. In particular, to test the developed grasp synthesis method using a real robot is one of the most interesting tasks to undertake. Generally speaking, this could be carried out following 2 possible approaches:

i) The first option is to validate the applicability of the developed method to real-life situations, without performing any additional training step. This would allow finding if the method is useful for a real robot, as well as if the training performed in a simulated environment can be extrapolated to real-life grasping scenarios, with all the advantages that such a result could mean (for example, a less time-consuming training stage).

ii) The second option corresponds to validating the applicability of the developed method

by performing a full real-life training and testing.

On the other hand, if it is decided to keep the simulator-based development, a thorough evaluation of the trained classifier should be considered for future work.

Finally, a possible development stemming from this work could be to extend the usage of the method used for the generation of spatial clusters, currently applied to generate grasp candidate points. Such method could be applied to produce a blob-based description, representing an object as a graph of spatially connected blobs. Such information can be used in combination with recognition algorithms to identify each part of the object, producing a semantic description of it.

Acronyms

3DSC	3D Shape Context.
BoVW	Bag-of-Visual-Words.
BoW	Bag-of-Words.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
DCH	Directed Curvature Histograms.
DoF	degrees of freedom.
FPFH	Fast Point Feature Histograms.
LRF	Local Reference Frame.
OSRF	Open Source Robotics Foundation.
PCL	Point Cloud Library.
PFH	Point Feature Histograms.
RA	Reference Axis.
ROS	Robot Operating System.
SHOT	Signatures of Histograms of Orientations.
SI	Spin Images.
SVM	Support Vector Machine.
USC	Unique Shape Context.

Glossary

Codebook	In the context of the BoVW model, a Codebook is a set of representative vectors of a feature space. Sometimes it is also referred to as the *vocabulary* of the represented feature space.
Dense Evaluation	Process of evaluating a local descriptor on every point of a cloud, or every pixel of an image.
Effector	Name given to a robotic hand, gripper or tool in the most general case, used by a robot to perform manipulation tasks. Sometimes it is also referred to as *end-effector*.
Gazebo Simulator	3D dynamic robotic simulator, developed by The Open Source Robotics Foundation.
Grasp Candidate	Combination of a grasping point candidate, placed on the surface of the target object, plus an orientation for the effector of the robot.
Grasp synthesis	Name given to the problem of determining a parametrization of a robotic hand (or effector), which allows it to grasp an object.
Grasping Point	Point used to produce a grasp candidate by adding it an orientation for the effector of the robot.
Point-cloud	Three dimensional sensory type, characterized by the representation of the sensory data a set of 3D points.
PR2 Robot	Humanoid manipulation robotic platform developed by Willow Garage [19], starting in 2010.

Bibliography

[1] Thomas Feix, Ian M. Bullock, and Aaron M. Dollar. "Analysis of human grasping behavior: Object characteristics and grasp type." In: *IEEE Transactions on Haptics* 7.3 (2014), pp. 311–323.

[2] Beatriz León, Antonio Morales, and Joaquin Sancho-Bru. "Robot Grasping Foundations." In: *Cognitive Systems Monographs*. Vol. 19. 2014, pp. 15–31.

[3] Rodrigo Schulz, Pablo Guerrero, and Benjamin Bustos. "Directed Curvature Histograms for Robotic Grasping." In: *Eurographics Workshop on 3D Object Retrieval*. Ed. by Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov. The Eurographics Association, 2017.

[4] Vladimir Vapnik and Alexander Lerner. "Pattern Recognition Using Generalized Portrait Method." In: *Journal of Automation and Remote Control* 24.6 (1963), pp. 774–780.

[5] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks." In: *Machine Learning* 20.3 (1995), pp. 273–297.

[6] Gokmen Zararsiz, Ferhan Elmali, and Ahmet Ozturk. "Bagging Support Vector Machines for Leukemia Classification." In: 2012.

[7] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. 1st. Springer Publishing Company, Incorporated, 2008.

[8] James B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations." In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1.233 (1967), pp. 281–297.

[9] David Arthur and Sergei Vassilvitskii. "K-Means++: the Advantages of Careful Seeding." In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* 8 (2007), pp. 1027–1025.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[11] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), pp. 603–619.

[12] Frederic Jurie and Bill Triggs. "Creating efficient codebooks for visual recognition." In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1.* Vol. 1. 2005, 604–610 Vol. 1.

[13] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)." In: *Proceedings - IEEE International Conference on Robotics and Automation.* IEEE, May 2011, pp. 1–4.

[14] Morgan Quigley et al. "ROS: an open-source Robot Operating System." In: *Icra* 3.Figure 1 (2009), p. 5.

[15] Open Source Robotic Foundation. *Open Source Robotic Foundation.* `https://www.osrfoundation.org/`. Accessed in February 15, 2017.

[16] Gary Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[17] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator." In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* 3 (2004), pp. 2149–2154.

[18] Willow Garage. *The PR2 Robot.* `http://www.willowgarage.com/pages/pr2/overview`. 2010 (accessed in February 14, 2017).

[19] Willow Garage. *Willow Garage.* `http://www.willowgarage.com/pages/about-us`. Accessed in February 15, 2017.

[20] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. "Data-Driven Grasp Synthesis: A Survey." In: *IEEE Transactions on Robotics* 30.2 (Apr. 2013), pp. 1–21.

[21] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. "An overview of 3D object grasp synthesis algorithms." In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 326–336.

[22] Van-Duc Nguyen. "Constructing force-closure grasps in 3D." In: *1987 IEEE International Conference on Robotics and Automation.* Vol. 4. Institute of Electrical and Electronics Engineers, 1987, pp. 1368–1373.

[23] Karun B. Shimoga. "Robot Grasp Synthesis Algorithms: A Survey." In: *The International Journal of Robotics Research* 15.3 (June 1996), pp. 230–266.

[24] Antonio Morales, Tamim Asfour, Pedram Azad, Steffen Knoop, and Rüdiger Dillmann. "Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands." In: *IEEE International Conference on Intelligent Robots and Systems.* IEEE, Oct. 2006, pp. 5663–5668.

[25] Andrew Miller and Peter Allen. "Graspit: A versatile simulator for robotic grasping." In: *IEEE Robotics and Automation Magazine* 11.4 (2004), pp. 110–122.

[26] Andrew Miller, Steffen Knoopt, Henrik Christensen, and Peter Allen. "Automatic grasp planning using shape primitives." In: *International Conference on Robotics and Automation* 2 (2003), pp. 1824–1829.

[27] Kai Huebner and Danica Kragic. "Selection of robot pre-grasps using box-based shape approximation." In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2008), pp. 1765–1770.

[28] Matei Ciocarlie and Peter Allen. "Hand Posture Subspaces for Dexterous Robotic Grasping." In: *The International Journal of Robotics Research* 28 (2009), pp. 851–867.

[29] Renaud Detry et al. "Learning object-specific grasp affordance densities." In: *2009 IEEE 8th International Conference on Development and Learning, ICDL 2009*. Vol. 2. 1. IEEE, 2009, pp. 1–7.

[30] Oliver Kroemer, Renaud Detry, Justus Piater, and Jan Peters. "Combining active learning and reactive control for robot grasping." In: *Robotics and Autonomous Systems* 58.9 (Sept. 2010), pp. 1105–1116.

[31] Freek Stulp et al. "Learning motion primitive goals for robust manipulation." In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 28. 6. IEEE, Sept. 2011, pp. 325–331.

[32] Renaud Detry, Dirk Kraft, Anders Glent Buch, Norbert Krüger, and Justus Piater. "Refining grasp affordance models by experience." In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, May 2010, pp. 2287–2293.

[33] Sahar El-Khoury and Anis Sahbani. "Handling objects by their handles." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on grasp and task learning by imitation* (2008), pp. 58–64.

[34] Ying Li and Nancy S. Pollard. "A shape matching algorithm for synthesizing humanlike enveloping grasps." In: *Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots*. Vol. 2005. IEEE, 2005, pp. 442–449.

[35] Antonio Morales, Eris Chinellato, Andrew H. Fagg, and Angel P. del Pobil. "Using Experience for Assessing Grasp Reliability." In: *International Journal of Humanoid Robotics* 01.04 (Dec. 2004), pp. 671–691.

[36] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. "Learning object affordances: From sensory - Motor coordination to imitation." In: *IEEE Transactions on Robotics* 24.1 (Feb. 2008), pp. 15–26.

[37] Ulrich Hillenbrand and Maximo A. Roa. "Transferring functional grasps through contact warping and local replanning." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 2963–2970.

[38] Hao Dang and Peter K. Allen. "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task." In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 1311–1317.

[39] Claire Dune, Eric Marchand, Christophe Collowet, and Christophe Leroux. "Active rough shape estimation of unknown objects." In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, Sept. 2008, pp. 3622–3627.

[40] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. "General 3D modelling of novel objects from a single view." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2010, pp. 3700–3705.

[41] Dirk Kraft et al. "Birth of the Object: Detection of Objectness and Extraction of Object Shape Through Object–Action Complexes." In: *International Journal of Humanoid Robotics* 05.02 (June 2008), pp. 247–265.

[42] Mila Popović, Gert Kootstra, Jimmy Alison Jørgensen, Danica Kragic, and Norbert Krüger. "Grasping unknown objects using an early cognitive vision system for general scene understanding." In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2011, pp. 987–994.

[43] Antonio Morales, Pedro J. Sanz, Angel P. del Pobil, and Andrew H. Fagg. "Vision-based three-finger grasp synthesis constrained by hand geometry." In: *Robotics and Autonomous Systems* 54.6 (June 2006), pp. 496–512.

[44] Mario Richtsfeld and Markus Vincze. "Grasping of Unknown Objects from a Table Top." In: *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments* (2008).

[45] Zhengyou Zhang. "Microsoft Kinect sensor and its effect." In: *IEEE Multimedia* 19.2 (2012), pp. 4–10.

[46] Federico Tombari, Samuele Salti, and Luigi Di Stefano. "Unique signatures of histograms for local surface description." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6313 LNCS.PART 3 (2010), pp. 356–369.

[47] Chin Seng Chua and Ray Jarvis. "Point Signatures: A New Representation for 3D Object Recognition." In: *International Journal of Computer Vision* 25.1 (1997), pp. 63–85.

[48] Yiyong Sun and Mongi A. Abidi. "Surface matching by 3D point's fingerprint." In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 2. July. IEEE Comput. Soc, 2001, 263–269 vol.2.

[49] John Novatnack and Ko Nishino. "Scale-Dependent / Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images." In: *Eccv* 5304 (2008), pp. 440–453.

[50] Ae Johnson. "Spin-images: a representation for 3-D surface matching." In: *Carnegie Mellon University* CMU-RI-TR-97-47 (1997), p. 138.

[51] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. "Recognizing Objects in Range Data Using Regional Point Descriptors." In: *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III* 3023 (2004), pp. 224–237.

[52] Federico Tombari et al. "Unique Shape Context for 3D Data Description." In: *Proceedings of the ACM workshop on 3D object retrieval - 3DOR '10*. New York, New York, USA: ACM Press, 2010, pp. 57–62.

[53] Serge Belongie and Jitendra Malik. "Matching with shape contexts." In: *Proceedings - IEEE Workshop on Content-Based Access of Image and Video Libraries, CBAIVL 2000*. Vol. 00. c. IEEE, 2000, pp. 20–26.

[54] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. "Persistent point feature histograms for 3D point clouds." In: *Intelligent Autonomous Systems 10, IAS 2008* (2008), pp. 119–128.

[55] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration." In: *IEEE International Conference on Robotics and Automation* (2009), pp. 3212–3217.

[56] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. "A large-scale hierarchical multi-view RGB-D object dataset." In: *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 1817–1824.

[57] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching." In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[58] Radu Bogdan Rusu. "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments." In: *KI - Künstliche Intelligenz*. Springer Tracts in Advanced Robotics 24.4 (Nov. 2010), pp. 345–348.

[59] Andre Ückermann, Christof Elbrechter, Robert Haschke, and Helge Ritter. "3D scene segmentation for autonomous robot grasping." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1734–1740.

[60] Deepak Rao et al. "Grasping novel objects with depth segmentation." In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 2578–2585.

Appendix A

Clustering Algorithms

A.1 K-Means Pseudo Code Implementation

Algorithm 1: Pseudo code implementation of K-Means algorithm.

input : $D = \{\mathrm{d}_0, \mathrm{d}_1, \ldots, \mathrm{d}_N\}$: data to be clustered
k: number of clusters to be found
M: maximum number of iterations to evaluate
output : $C = \{c_0, c_1, \ldots, c_k\}$: set of found centroids
$L = \{l_1, l_2, \ldots, l_N\}$: labels for each $\mathrm{d_i} \in D$

foreach $c_j \in C$ **do**
  $c_j \leftarrow \mathrm{d_i}$ `// randomly select` k `initial centroids`
end
while *number of iterations* $< M$ **do**
  foreach $\mathrm{d_i} \in D$ **do**
    $l_\mathrm{i} \leftarrow argmin_C(\mathrm{d_i})$ `// label each point according to its closest` c_j
  end
  foreach $c_j \in C$ **do**
    $c_j \leftarrow mean(\forall\, \mathrm{d_i} \in D \mid l_\mathrm{i} = c_j)$ `// update the position of each` c_j
  end
  if *position of each* c_j *have not changed* **then**
    stop
  end
end

A.2 DBSCAN Pseudo Code Implementation

Algorithm 2: Pseudo code implementation of DBSCAN algorithm.

input : $D = \{\mathrm{d}_1, \mathrm{d}_2, \ldots, \mathrm{d}_N\}$: data to be clustered
ε: radius of the vicinity of a point
m: minimum number of points required to be considered a core point
output : $C = \{C_0, C_1, \ldots, C_k\}$: set of clusters found

```
1  foreach d_i ∈ D do
2      if d_i is already processed then
3          continue to next element
4      end
5      V_iε = {d_j ∈ D | ‖d_j − d_i‖ < ε} // vicinity of d_i
6      if ‖V_iε‖ < m then
7          mark point as noise
8      else
9          define C_k as the next cluster
10         C_k = C_k ∪ {d_i}
11         foreach d_j ∈ V_iε do
               // try to add more points to the cluster
12             if d_j has not been visited then
13                 mark d_j as visited
14                 V_jε = {d_q ∈ D | ‖d_q − d_j‖ < ε} // vicinity of d_j
15                 if ‖V_jε‖ ≥ m then
16                     V_iε = V_iε ∪ V_jε
17                 end
18             end
19             if d_j is not member of any cluster then
20                 C_k = C_k ∪ {d_j}
21             end
22         end
23     end
24 end
```