LEARNING DEEP REPRESENTATIONS FOR ROBOTICS APPLICATIONS

by

ÜMİT RUŞEN AKTAŞ

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
May 2018

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

Abstract

In this thesis, two hierarchical learning representations are explored in computer vision tasks. First, a novel graph theoretic method for statistical shape analysis, called Compositional Hierarchy of Parts (CHOP), was proposed. The method utilises line-based features as its building blocks for the representation of shapes. A deep, multi-layer vocabulary is learned by recursively compressing this initial representation. The key contribution of this work is to formulate layer-wise learning as a frequent sub-graph discovery problem, solved using the Minimum Description Length (MDL) principle. The experiments show that CHOP employs part shareability and data compression features, and yields state-of-the-art shape retrieval performance on 3 benchmark datasets. In the second part of the thesis, a hybrid generative-evaluative method was used to solve the dexterous grasping problem. This approach combines a learned dexterous grasp generation model with two novel evaluative models based on Convolutional Neural Networks (CNNs). The data-efficient generative method learns from a human demonstrator. The evaluative models are trained in simulation, using the grasps proposed by the generative approach and the depth images of the objects from a single view. On a real grasp dataset of 49 scenes with previously unseen objects, the proposed hybrid architecture outperforms the purely generative method, with a grasp success rate of 77.7% to 57.1%. The thesis concludes by comparing the two families of deep architectures, compositional hierarchies and DNNs, providing insights on their strengths and weaknesses.

to mum and dad

Acknowledgements

A heartfelt thanks to mum and dad for supporting me at all times. This thesis was written standing on the shoulders of these two giants. Special thanks to mum's special frozen cherries and dad's cool stories.

To my sister Berfin, for sharing a special bond and inspiring me in life. Her strong support and pep talks were instrumental in me achieving this goal.

To my dear Raluca, for accompanying and supporting me through the ups and downs of this journey. Her joy, warm heart and compassion have been my beacons of light.

To Jeremy, for showing the way in the complex labyrinth and helping to carve a path for my studies through his willpower and lead. He was invaluable, and it was a great pleasure to work with him.

To Ales, for helping kick-start my studies, the needed motivation and countless ideas.

To Fatoş Hoca, for her support, and inspiring me to study for a PhD.

To my peers and friends Manolis, Maxime, Fani, Lenka and Vladislav; as well as Tomas, Bram, Jiangshan, Mohab and Auke for sharing the burden and helpful discussions.

To my friends and colleagues Mete and Mirela, for great collaboration and friendship.

To Chao, Marek, and Claudio, whose contributions were crucial in my research.

To Justus for the numerous discussions and reviewing my thesis. My thanks to Mohan for reviewing this thesis and his helpful tips. And to Dave, for chairing my viva.

To Ela and Peter, for being the voices of reason along the way.

To Filiz Hanım, for helping me cope with life.

To my friend Onur, who instilled his unique combination of food, basketball and chillout taste in me, for the positive distractions.

To my long-term friend Buğra, with whom I also happened to have one of the most exciting professional chapters of my life.

To the many more who I have come across along the way, including my teachers, professors, family, friends and colleagues. To those I have forgotten to mention.

Thank you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Object recognition at the class level has been a fundamental problem in computer vision for the past 50 years. The total number of object classes (i.e. human defined) has been estimated to be around 30,000 [17]. Accounting for background clutter, occlusion, illumination, camera viewpoint and scale variables, the number of images algorithms have to deal with is almost infinite. One goal of computer vision has been to recognise objects, in increasingly complex conditions, within an acceptable duration. This is a task primates with complex visual systems are extremely good at. The biological visual systems of humans and monkeys have evolved to deal with the seemingly infinite combinations of object classes, poses, scales and illumination conditions in the order of miliseconds [181, 50].

The object recognition problem, broadly, contains two sub-problems: object classification and object detection. The classification of an object (or image) can be stated as a question: “What types of objects are there in image X?”. Object class recognition is about finding the type of an object (e.g. human), rather than recognising a specific instance (person A). Approaching this problem requires dealing with inter-class differences (e.g. cars having different colours and designs), varying object poses and scales, illumination conditions, clutter, etc. A brute-force solution may take all of these factors of variation into account, e.g. by creating a template for every possible combination of variables. Such an approach makes the problem arguably intractable [219]. However, modern deep hierarchies have efficiently solved this problem using vast amounts of labeled data, even surpassing human performance in some respects [79]. The ImageNet dataset [38] is widely regarded as a benchmark for object recognition. The ImageNet Large Scale Visual Recognition Challenges (ILSVRC 2010-2017) are based on training a 1000-class subset of ImageNet, containing millions of training images, for the object recognition task.

The detection problem is based on the same set up, but it asks a follow-up question: "Where are the objects in the image X?". In addition to finding the object classes, the locations of the discovered objects may also be of interest. The coordinates of an object's location, its parts (e.g. left/right arm, head of a person), the coordinates and size of the enclosing bounding box, or segmentation of the pixels occupied by the object provide information at different levels of granularity about the detected object. Pascal VOC Challenges [48] have been extensively used in benchmarking object detection algorithms, along with the ImageNet Dataset [38], and more recently, the Microsoft Co-Co Dataset [130]. Additionally, this problem resonates with the research questions in the robotics community. In order to manipulate or navigate around an object, a robot not only needs to identify its class, but also has to detect its location, scale, and pose (orientation).

Methods that address these problems can be grouped according to three different ways. First is the *representational power*, which relates to the capacity to model the variety of objects and their parts that can be recognised. Second is the training methodology, which can be *unsupervised* or *supervised*, or a combination of both. The *inference* mechanism defined for recognition is the third property, and concerns devising a procedure that uses the learned representations to determine the presence of an object in an image. As illustrated in the next chapter, deep, hierarchical learning methods have emerged as high-capacity learners that can perform unsupervised or supervised learning with efficient inference mechanisms.

Two relevant learning frameworks are introduced in this thesis. The first one, a statistical shape analysis method given in Chapter 3, focuses on shape retrieval and classification tasks, which are closely related to the object recognition problem. The shapes in question mostly consist of black-and-white images and have relatively little clutter. The second method of interest, a generative-evaluative framework for dexterous grasping, explained in Chapter 5, aims to solve the object detection and pose estimation tasks in order to grasp objects. These two core parts are not directly connected to each other. While they both use hierarchical architectures to model visual data, the underlying frameworks differ in detail. The reader is advised to treat them as two distinct parts, though certain similarities will be pointed out where necessary. The final goal of this thesis is to enable the robot to grasp objects using a human-like hand, which typically requires the precise localisation of the objects in a scene.

Section 1.1 is dedicated to the properties and advantages of deep and hierarchical methods, which constitute the core algorithms presented in this thesis. A summary of the contributions made in this thesis is given in Section 1.2. Finally, Section 1.3 contains the thesis outline.

1.1 Motivation

This thesis introduces two deep and hierarchical feature learning approaches, used for object recognition and detection tasks. There are two reasons why these methods were favoured over hand-crafted features: inspiration from complex biological vision systems, and the overwhelming empirical evidence highlighting the advantages of deep methods. These reasons are explained below.

1.1.1 Biological Reasons

Early hierarchical models drew their inspiration from biological vision systems [63, 115, 174]. It is known that in the primate brain, object recognition happens in a pathway spanning from the primary virtual cortex, V1, to the inferotemporal cortex, IT, [221]. The last layer, IT, is where the visual data processing finishes and recognition is finalised [137, 221, 139]. Recognition involves multiple layers of signal processing [51] in mostly a feed-forward manner [160]. In this deep network of neurons, the lower layers are responsible for the detection of simple features, such as line segments or colour [109], and more complex cells operate on the outputs of these layers. Some core properties of the biological systems, such as translation invariance [102, 137] in higher layers and sharing of parts [109], have contributed ideas to deep architectures. These topics are further explored in Section 2.1.

1.1.2 Empirical Evidence

Hand-crafted features have proven to be very popular in recognition tasks. Various methods, such as: Scale-Invariant Feature Transform (SIFT) [138], Speeded Up Robust Features (SURF) [10], Histogram of Gradients (HoG) [36], Shape Context (SC) [12], Local Binary Patterns (LBP) [155], among others, were extensively used in classification of objects and shapes. These methods are based on representing a local patch or keypoint using a descriptor, and using these descriptors as features in a discriminative task. Often, the extracted features are used to train a classifier, such as a Support Vector Machine (SVM) [34]. Feature-based methods, termed *flat* approaches, have proven to be insufficient in modeling complex object classes. Hierarchical methods [63, 57, 103, 247, 245, 174, 2], on the other hand, can easily represent the complex relationships between an object's parts and varying appearance features. The key advantage of hierarchical compositional methods is the ability to start with relatively simple features and learn statistical, multi-layer and deformable part models. The complex features in these methods are richer than flat approaches, since they may be deformed to represent the patterns and variations in the data in ways feature-based methods can not. The reader is advised to refer to Section 2.4 for further discussion.

Most methods which currently perform well on object recognition and detection benchmarks are variants of Deep Neural Networks (DNNs) [13]. The origins of these methods can be traced to the Perceptron [180], and later Neocognitron [63], which was directly influenced by biological vision systems. The Backpropagation algorithm [182] enabled the efficient training of DNNs. However, the research on neural networks stagnated for more than 20 years due to the lack of hardware to train DNNs with high capacity. Since Krizhevsky et al.'s breakthrough network AlexNet in 2012 [107], DNNs have become popular. They have also become deeper, from the 8-layer AlexNet [107] to the 152-layer ResNet [79], which surpassed human performance in the ILSVRC-2015 challenge. DNNs have convincingly outperformed other approaches on most vision challenges [13], adding to the empirical evidence indicating the advantages of deep architectures. A more detailed review of deep methods is given in Section 2.5.

1.2 Summary and Contributions

This thesis presents two deep architectures: a compositional shape hierarchy, and a neural network. The first method is a statistical shape learning framework, used for shape retrieval and classification. The system is based on line-based features at its lowest level, and learns a compact hierarchical vocabulary of object models and parts. The contributions of our first work can be summarised in three points:

1. A novel graph-theoretic method, termed Compositional Hierarchy of Parts (CHOP), is proposed to learn a compositional hierarchy for representing object shape and parts. Unlike most compositional methods, CHOP provides a principled way to approach the statistical shape analysis problem. Tools based on graph theory, such as subgraph isomorphism, graph compression and graph matching, are used to perform learning and inference of shape models.

2. Two information-theoretic tools are employed during training in order to learn the statistical relationships defining object parts and shape vocabularies. The first one is Minimum Conditional Entropy Clustering (MCEC), which is used to analyse the pairwise spatial relationships between two parts [124]. The second tool is the Minimum Description Length (MDL) principle, which selects a compact set of shapes to construct the shape vocabulary.

3. The proposed method is a hybrid generative-descriptive model for hierarchical shape representation. The learned parts are generative, since part instances can be generated by sampling from a part's parse tree. MDL-driven learning ensures that the learned parts are *descriptive* of the shapes that are frequently encountered during training.

The proposed approach was tested on shape retrieval and classification tasks using 2D shapes, yielding state-of-the-art results on tested benchmark datasets in 2014. Compared with object recognition targeted by DNNs [107, 211, 79] trained on large-scale image datasets [38], statistical shape learning remains relatively under-studied by deep methods. In the tested benchmark datasets, given in Sections 3.4 and 3.5, the number of training images is typically much smaller (20-1000) than ImageNet ($>$1M), presenting an interesting challenge for data-efficient learning.

The second part of the thesis focuses on a robotic manipulation task: dexterous grasping of novel objects using a single-view of the object from a depth camera. The problem of grasping is approached in a generative-evaluative framework. A number of plausible candidate grasps are sampled using a generative model. This generative model is trained using an efficient method that learns from a handful of human-demonstrated grasps (and is not a contribution of this thesis). These grasps are then ranked by two proposed evaluative models, which are data-intensively trained deep networks. Designing a hybrid learning architecture has two clear advantages. First, generative models select grasps using a good prior, although the generated grasps are ranked by a likelihood function which does not necessarily correlate with the grasp success. The evaluative model replaces these metrics as a method for ranking promising grasps. Second, the evaluative model benefits from the high quality grasps that a generative model brings. A generative model follows a pattern when sampling grasps, as opposed to random generation. This leads to a structured grasp configuration space that can be parameterised and learned by a deep network. This work makes the following contributions:

1. This thesis presents the first dexterous grasping method which combines a *learned*

generative model with a *learned* evaluative model for dexterous grasping of novel objects.

2. A highly-realistic simulated data collection setup for performing grasps was developed, based on the MuJoCo physics simulator. A Kinect simulator [19] was modified and tuned for the Carmine 1.09 depth sensor for obtaining simulated depth images.

3. A new dataset of 3D object models, consisting of 295 objects from 20 classes was collected. All objects are graspable using a dexterous hand, and have realistic scales. The object models are accompanied by their approximate convex decompositions, calculated using the V-HACD algorithm [146].

4. Two millions grasps were performed in simulation, and each grasp was saved with its corresponding simulated depth image, hand parameters, and results in the form of grasp success and stability. The collected dataset can be used for predicting grasp success probability, along with grasp stability (how the object moves and rotates in the hand).

The code and the collected data will be released online under an open-source license.

1.3 Thesis Outline

This thesis consists of two core parts. The goal of the thesis is to introduce both methods, and draw the connections between. Chapters 2 and 3 focus on hierarchical shape learning. Next, Chapters 4 and 5 introduce the work on dexterous grasping of novel objects using deep networks. Finally, 6 concludes the thesis by discussing the results and comparing both problems and approaches.

In Chapter 2, a detailed literature review about representation learning for object recognition and classification is given. The chapter starts with the biological arguments for choosing hierarchical learning techniques over feature-based methods. Next, an evolutionary history of the literature from feature-based methods to hybrid multi-layer techniques, and finally, deep representations is given. The deep methods of interest in this thesis, compositional hierarchies and deep networks, are reviewed in this chapter.

The proposed compositional framework, CHOP, is presented in Chapter 3. First, the method is summarised by introducing the learning and inference mechanisms. Explanation of the building blocks that constitute the CHOP framework follow this summary. The experimental section includes the results of the method on the shape retrieval task in 6 benchmark datasets. Part sharing and sub-linear vocabulary growth features are illustrated via analysis by varying object sizes, classes and views. Further results are provided in the shape classification task, showcasing the method's ability to model invariances to rotation, scale and background noise.

Next, we switch to the robotics domain. Chapter 4 contains an overview of the existing work in grasping. The proposed architecture in the second part of the thesis contains a deep network, and this chapter contains relevant research in the context of grasping. The geometric methods, which dominated early grasping literature, are followed up with more recent data-driven techniques which have shown great success in grasping. The chapter concludes with the most relevant research to our work: the methods that use deep networks for dexterous grasping.

A generative-evaluative approach for dexterous grasping of novel objects from a single-view image is proposed in Chapter 5. The innovative aspect of this work comes from combining a data-efficient generative model with a data-intensive evaluative model based on a deep network. Next, the simulated data collection setup, which was used to collect the data to train the deep network, is explained. This chapter also contains the experimental analysis for simulation and the real world, which clearly shows that the proposed architecture improves grasp success rate significantly.

Finally, the conclusion of the thesis is given in Chapter 6. This chapter contains a summary, a discussion of the hierarchical approaches and future directions for research.

Part I

Hierarchical Shape Representations

CHAPTER 2

RELATED WORK: 2D IMAGE REPRESENTATIONS

This chapter gives an overview of existing approaches on object class recognition. Below, a summary of the development of computer vision algorithms for 2D image understanding is given, as well as successful examples through the years. First, we set out the motivation for hierarchical approaches and their biological origins in Section 2.1. Next, "flat" methods which dominated early work on object recognition are introduced in Section 2.2. Section 2.3 lists transitional algorithms that combine features into *parts* to represent objects. Section 2.4 is dedicated to compositional hierarchical systems. Finally, Section 2.5 contains deep neural networks.

2.1 Biological Motivation for Hierarchical Learning

The hierarchical multi-layer structure of modern computer vision techniques is supported by research about biological vision systems. The ventral visual pathway is considered to be responsible for object recognition [221], spanning an area from the primary virtual cortex, V1, to the inferotemporal cortex, IT. IT is thought to be central for the recognition task [137, 221, 139]. Some well-known and widely accepted elements of the primate vision systems are given below:

- The vision processing pipeline spans 8-10 layers [51],

- Information channels such as colour, shape, motion, texture, 3D information are processed separately, though being highly inter-connected [109],
- Layers can exploit the *shareability* of elements, since they are built on top of each other [109],
- Receptive field sizes and translation invariances increases towards higher layers [137, 102, 160].
- Object recognition is a predominantly feed-forward process, although feedback does exist [160].

Considering the speed of recognition in the primate brain, these findings are not surprising. The lower layers of the visual hierarchy consist of simple features such as colour, line segments, texture and motion elements. Higher layers become more invariant to the exact location of the stimuli [137, 102]. The learned representations yield a degree of translation and rotational invariance even in the case of limited viewpoints encountered during training [174]. The best scheme that explains such invariances is the MAX operation performed in the "complex" cells (cells which operate on the output of sensory neurons) [174]. The MAX operation takes the maximum of the signals in a receptive field, effectively providing feature specificity and invariance to the signal's exact location and to background clutter.

The shareability of lower-layer features in the visual hierarchy is a key reason why hierarchical methods are efficient learning representations. For example, T-junctions and corners can be found in many distinct shapes, hence only one copy of each needs to be kept in the vocabulary (per view). Higher layers get more specialised as the receptive fields get bigger and shape complexity increases, resulting in a higher number of elements. Shareable features help limit the otherwise combinatorial explosion of parts in the vocabulary. An empirical analysis of the relationship between shareability, vocabulary size and run-time performance is given in the next chapter.

The elements listed above exist in most hierarchical learning methods, and their examples can be found in Sections 2.4 and 2.5. The next section is dedicated to the literature on non-hierarchical methods, which were extensively used in object recognition.

2.2 Flat Approaches

Feature-based methods have led the way in object recognition/detection up until the last decade [138, 229, 36]. In this section, we consider feature-based methods and their learned counterparts, single layer networks [32], under the name of "flat approaches". Such methods are based on *designing* or *learning* a set of features that *together* describe an object. Because each feature focuses on a small part of the object, the high-level structural information is ignored, leading to a less powerful representation. Usually, learned/designed methods are coupled with Support Vector Machines (SVM) [34] or other classifiers to perform the recognition task. It should be noted that most of the methods in this section can be classified as "unsupervised" techniques, where images' category labels do not contribute to the feature design/learning process.

Feature-based methods are not able to represent an object using a single feature mask/filter. The appearance variability at the object level is prohibitive for an object-level representation. On the other hand, hierarchical approaches can be designed to model whole objects at their top layers, as explained in Section 2.4. They can recursively compress the image data to obtain more compact representations. This allows them to have abstract classes such as "car", which can be decomposed into sub-structures such as "door", "tyre", "trunk", etc. These sub-structures can be further recursively decomposed, until the basic building blocks (e.g. "line segment"s) are obtained. Similarly, deep learning methods in Section 2.5 are able to learn abstract features which cover a large receptive field, possibly enclosing the whole object. The complexity of the representations in both hierarchical compositional techniques and deep neural networks are higher than feature-based methods, and results in better performances [13].

The feature sets of the techniques in this section are often required to be over-complete, as is in the case of Viola-Jones face detector [229] or Haar-like features [127]. Keypoint detection methods such as SIFT [138] and Harris Corner Detector [77] represent objects with a number of keypoints, and the keypoint positions and descriptors can be used for matching objects. Feature learning approaches such as auto-encoders [167, 228] and k-means [49] strive to learn features which collectively reconstruct the data, and the performance increases as the learned dictionaries contain more redundancy [32]. It can be inferred that as over-completeness and redundancy in a representation increases, the performance rises as well.

Earlier work on object/shape recognition involved using either hand-crafted descriptors [138, 10, 36] or learned features [32]. Hand-crafted features such as SIFT [138] are based on detecting interest points in an image, and obtaining a local descriptor for each point. The interest points (landmarks) in an image *collectively* represent an object, and recognition is usually performed by matching the constellation of points in pairs of compared images. For instance, SIFT keypoint detector is a scale and rotation invariant technique to detect extremity points (blobs, corners, etc) in an image [138]. SIFT descriptor is a 128-dimensional histogram which defines the area around the relevant keypoint, and is used for matching keypoints. A query object and a sample object can be compared by finding the best transformation to match the keypoint sets belonging to the two objects, using a method such as [61]. SIFT, and its speeded-up version SURF [10], have been extensively used in recognition and detection tasks until the last decade. However, the deep learning methods explained in Section 2.5 have recently outperformed them in nearly all object recognition/detection tasks [13].

A dictionary of features can be *learned* to represent an image or an object. A straightforward approach into feature learning is to specify a patch size and learn a set of features of the set size to perform a task. For example, the k-means [49] algorithm can be used to cluster (label) a number of patches randomly collected from the images in the training set. This is analogous to creating a dictionary of visual words to explain the data. A variation of the dictionary approach is the work of van Gemert et al [223], where *soft labels* are used to label visual features. An analysis of single-layer networks [32] by Coates et al. shows that learning the features using k-means clustering is often as effective as single-layer networks such as sparse auto-encoders (SAEs) [167], sparse Restricted Boltzmann Machines (RBMs) [116] and Gaussian Mixture Models (GMMs) [172, 32].

Variations of Deep Neural Networks (DNNs) can be used for feature learning. In this thesis, the review of the literature containing DNN-based recognition methods is divided into two. The rest of this section focuses on DNN architectures specifically used for learning patch-based image features. Class-level recognition approaches are explained in Section 2.5.

Auto-encoders (AE) [167, 228] are neural networks that encode/decode the data to learn the underlying structures. Typically, an auto-encoder framework consists of an encoder network f and a decoder network g, and the network weights are learned by reconstructing input x by going through the encode/decode operation $g(f(x))$. By using encoder/decoder networks of limited capacity and regularisation techniques, auto-encoders can avoid over-fitting to specific examples in the training set. Sparse auto-encoder (SAE) is an interesting variation of the AEs. Sparsity is a desired property which helps preventing overfitting to the training set. The sparsity constraint can either be applied by penalising the hidden unit biases (to reduce the number of active hidden nodes), or directly penalising the activations in the output of the encoder network $f(x)$ [14]. Another variation of AEs is the denoising auto-encoder (DAE), which aims to denoise an artificially corrupted input rather than reconstructing it. SAE has to learn the identity function within restricted framework, while DAE must capture the underlying input distribution and ignore the added noise. Both architectures have been useful in performing recognition tasks, despite being unsupervised methods [14].

There are various other feature learning methods such as Principle Component Analysis (PCA) [159, 85], Restricted Boltzmann Machines (RBMs) [82], Deep Boltzmann Machines [186], manifold learning algorithms (e.g. t-SNE [222]), and many more [14, 32].

The methods explained in this section have paved the way to *deeper* methods which are currently successful in computer vision benchmarks, as evidenced by the the top performers in [13]. The next section is dedicated to part-based methods, which form compositions of learned/designed features, some of which are listed in this section. The deep learning methods, extending the learning-based algorithms in this section, are given in Section 2.5.

2.3 Part-Based Methods

The methods in Section 2.2 fall short of modeling whole shapes or objects, because their receptive fields are smaller than the objects, and the appearance variation at the object level is prohibitive. This section focuses on methods combining these features in shallow architectures, i.e. having 3 or fewer layers. Part-based methods use the structural information in objects (spatial arrangement of features) to a limited extent. The papers listed here can be considered to precede modern deep architectures.

Bag-of-Features (Bag-of-Visual-Words, BoF) is a methodology which originates from a similar model used in text classification, the Bag-of-Words (BoW) model. In the Bag-of-Words approach, a text document is interpreted using statistics such as the number of occurrences of each word [5]. In other words, each document is considered as a bag of words or phrases, regardless of their order. Bag-of-Features is a very similar technique applied in image retrieval problems [30]. Each image is represented as a bag of features, and the statistics of these features define the characteristics of the image.

The Bag-of-Features method ignores the spatial arrangements of the features in the image, losing precious spatial information: People expect to see the nose under the eyes, car tyres are below the car, etc. On the positive side, methods that ignore locations of features are often scalable to big datasets and are faster to evaluate [30, 203, 37]. Dean et al. [37] developed a scalable object detection framework by replacing the dot product operator in feature convolution with locally-sensitive hashing. The replacement of convolution with hashing speeds up the feature extraction process by at least four orders of magnitude. The mean average precision of the method over the full dataset is around 0.16 on 100.000 classes. Chum et al. [30] apply the BoF model to an image retrieval problem over a large dataset consisting of more than 1M images. They refine the results by assessing the top retrieved images based on the spatial distribution of features on the query image and retrieved images. The spatial verification step improves the performance of the algorithm substantially, adding further evidence to the importance of learning spatial structures.

Other works that use the BoF technique include *spatially-aware* approaches, which exploit complex visual words that represent the configurations of features. In [23], Bronstein and Bronstein augment the BoF model by taking the spatial relationships between features into account. [23] is a two-level hierarchy, where the first level consists of simple features, and the second level is comprised of common *geometric expressions* of these features. In their later work [25], Bronstein et al. applied *spatially-sensitive* BoF to the shape retrieval problem in a large-scale shape dataset consisting of articulated 3D models. They argue that combining 3D features into complex geometric expressions improves the performance of the retrieval algorithm. In the proposed method presented in the next chapter, we made the same observation. In our case, allowing the algorithm to build higher levels in the hierarchy, rather than limiting it to two levels, results in richer representations and more complete object models.

Generalised Hough transform (GHT), proposed by Ballard in 1981 [9], is another part-based recognition technique. It is a modification of the Hough Transform (HT) [86], which was originally created to match simple shapes such as lines and circles. Ballard enabled the use of Hough transform for arbitrarily defined shapes. Variations of GHT have been used for object detection in the literature [118, 144, 156, 117]. Gall et al. [64] introduced Hough forests (see also [214, 170]), which are random forests that perform the generalised Hough transform efficently. In [64], as is the norm in Hough voting, small shape patches (features) vote for a shape or object's location. This star-shaped model has a drawback: the method performs well in the case of solid objects, but fails on articulated/deformable objects. Guo et al. proposed a more robust voting scheme in [72], where parts' contibutions depend on their uniqueness and positional variance, but neither technique is able to model articulations. The hierarchical methods presented in the next section are well-suited to deal with such appearance variations, particularly by the help of OR-nodes [57, 247].

Part-based methods can also incorporate supervised learning. Object categories can be modeled by parts with deformable spatial configurations, and this idea has been used in various papers [47, 54, 62, 191]. Felzenszwalb et al. [53] learned *discriminative* deformable part models which outperformed the best results in 2006 PASCAL person detection challenge [36]. They trained latent SVM models where part positions with respect to the bounding box are treated as latent variables, hence the learned category models are the direct outcome of the discriminative training. Each coarse category filter, which spans the entire bounding box of an object, is accompanied by 6 deformable parts (heuristically selected). The part filters are based on Histogram of Gradients (HoG) features [36]. In an alternative approach, Savalle et al. [188] replaced the HoG features with Convolutional Neural Network (CNN) features, which are obtained from AlexNet [107]. Further works on bridging the connection between Deformable Part Models (DPMs) and CNNs include the paper by Girshick et al [66], where a procedure was devised to convert DPMs to CNNs. More recently, Dai et al. [35] used ideas (e.g. deformable convolution and Region of Interest pooling) from DPMs to improve the transformation modeling capabilities of CNNs.

2.4 Compositional Hierarchies

In the previous section, part-based methods with no clear hierarchical structure were examined. Here, the aim is to cover hierarchical, part-based methods which model objects, *usually* have 3 or more layers, and have complex part-to-part interactions (e.g. to model articulations) [246]. A hierarchical structure leads to increased model complexity and invariance properties, which are in line with the biological motivations set forth in Section 2.1. Part-based hierarchical architectures have been studied in object category representation [41], object detection [245, 103], human body tracking [154], animal detection [29] and face image reconstruction [236] problems. A review of hierarchical compositional models was published in [246].

Hierarchical feature learning approaches present an alternative over the flat approaches that often use hand-crafted/simple features. A hierarchical compositional learning method can automatically learn structural features by combining basic building blocks [57], whereas a flat method can only work with a single layer of features to perform recognition [43]. As put forth in Section 2.1, a recent analysis of the Primate Visual System by Kruger et al. [109] showed that the brain processes signals obtained from the cells in the eyes in a hierarchical manner, in the order of 8-10 levels. Each level in the hierarchy responds to a different class of complexity of features, in *receptive fields* of growing sizes. Many hierarchical vision algorithms have been developed that imitate this type of behaviour [246]. In order to have a tractable framework, compositional hierarchies balance a trade-off between the representation's complexity and the memory footprint/run-time of the algorithm, by optimising a cost function. In this aspect, they exhibit similarities with deep-learning algorithms that aim to learn a model of the data based on a task-specific criterion [14]. Similarly, the multi-level vocabulary learned in a hierarchical compositional architecture can be designed to solve object categorisation [196], recognition [20] and detection [52] problems, or be adaptively task-driven [4]. More generalised representations focus on object reconstruction and parsing, which means they are constructed to be as rich as necessary for their tasks, e.g. generating the data [103, 59].

While they do not fully exploit the hierarchical structure, 2-level learning models are able to learn combinations of simple features as well. Yu et al. [238] presented a 2-level sparse coding method that takes into account the spatial arrangement of low-level features. Similarly, the work of Jamieson et al. [92] is a 2-level part hierarchy. The method has a graph-based representation, in which nodes are the parts, and edges correspond to spatial relationships. Neither work has a trivial extension into a multi-layer hierarchy required to learn complex object models. The amount of variability that can be encoded in a part in a 2-layer hierarchy is bound by the small receptive fields, hence the learned representations have limited power.

Riesenhuber and Poggio proposed HMAX [174], a 5-layer hierarchical recognition model designed to mimic the functionality of the pathway from V1 to the posterior inferotemporal (PIT) layer in the primate visual cortex. The original version did not incorporate any training, and the basic building blocks at the bottom layer are line segments. Top-layer nodes of the hierarchy are view-specific, task dependent elements. HMAX can recognise translation and scale-invariant features. Later, Serre et al. extended HMAX [194] to include feature learning at the intermediate and top levels of the network. Further extensions of HMAX include the work of Hu et al. [88], where sparsity in activations was introduced, and binarised HMAX [240], which is significantly faster. It should be noted that HMAX was created based on Fukushima's Neocognitron [63], a multi-layer recognition framework with increasing translation invariances towards upper layers.

Hierarchical compositional methods are able to learn compositions up to the object/scene level. Yuille et al. [245] developed a method to learn deformable templates of objects for efficient parsing. While this method learns the representation from images, interest points over the object's boundaries are picked by hand, and provided as ground truth to the algorithm. This methodology is not feasible for large datasets because marking such landmarks is time-consuming, and in most cases, prone to errors. In [103], Kokkinos and Yuille presented a hierarchical object representation based on a graph-based structure. Their method is able parse objects in cluttered scenes at multiple scales, and only requires the bounding box of the objects in order to learn higher-level representations. They have an efficient inference algorithm resulting from a tree-shaped object representation.

Alternatively, a compositional architecture can be considered as a context-sensitive grammar consisting of scenes, objects, parts and visual primitives. In the work of Zhu and Mumford [247], the rules of the grammar define the relationships between nodes and their parents in a hierarchical structure. The grammar encodes an And-Or representation where And-nodes model co-occurences, and Or-nodes encompass alternatives (choices). For example, the deformations and articulations of object part can be represented with an Or-node in the template definition. Based on the same idea, Si and Zhu [200] proposed And-Or templates which are fully generative to the object pixels. Additionally, And-Or templates have been successfully adapted to various tasks including car detection [234], human activity prediction [166] and graph mining [242].

Fidler and Leonardis proposed a hierarchical compositional architecture, termed Learned Hierarchy of Parts (LHOP) in [59, 57, 58]. LHOP is a multi-layer part learning method, and is generative to the pixel level. The parts in LHOP encode a high degree of variability due to the recursive formulation of the layers. At the top-level of the vocabulary, parts represent full object shapes. The proposed method in this thesis, CHOP, is based on the key building blocks of LHOP. The key difference between the two algorithms is the formulation of the learning problem in purely graph-theoretic terms in CHOP, which results in a more refined and unified architecture. The number of levels is automatically determined based on the complexity and resolution of the images in the training dataset in CHOP, while LHOP uses a fixed number of layers.

The co-occurences of real-world objects in scenes give valuable contextual information which may help object detection. Large-scale datasets such as Microsoft Co-Co [130] and LSUN [237] have made it possible to train algorithms that learn the interactions of objects. Sudderth et al. proposed a scene-level hierarchical parsing algorithm in [208]. In [208], Object-part relationships and object-object interactions are represented in a single framework. Similarly, during the training of the hierarchical generative model of Li et al. [125], they labeled and segmented each object, as well as annotating the whole scene. In addition to object recognition, the hierarchical scene parsing methods can also be used to perform high-level tasks such as scene annotation [205, 98]. While contextual information can be valuable in solving ambiguities, most methods do not utilise object-to-object interactions to perform recognition [246].

It is possible to make a comparative review of different compositional hierarchies, such as Hierarchical Shape Models [103], And-Or networks [247], HMAX [174], LHOP [58] or CHOP [2]. However, such comparisons are bound to be qualitative. All aforementioned frameworks share common elements, such as part shareability and fast inference. Hierarchical Shape Models [103] have the advantage of encoding feature scales at the part level, coping with varying sub-part sizes. And-Or networks [247], LHOP [58] and CHOP [2] explicitly group shape alternatives, increasing the power of the representations. The proposed method in this thesis, CHOP, provides a graph-theoretic solution which is relatively easy to implement. It can be argued that the main disadvantage of the compositional hierarchies is that they do not share a common notation, terminology or formulation. This creates a natural barrier in their development, as there is not a single framework which provides the building blocks to implement a newly designed hierarchy. In contrast, DNNs are almost exclusively built on shared principles and building blocks. It is possible to implement novel or existing methods using common frameworks, such as TensorFlow, Theano, Caffe, PyTorch or others.

As put forth in Section 2.1, there are biological justifications for the important design decisions taken in compositional hierarchies, such as line-based shape features [174], part-shareability [109] and hierarchical processing [221]. Compositional hierarchies are advantageous in that the resulting representations are intuitive and interpretable. This is in sharp contrast with DNNs, where the models are inherently entangled because solvers such as Stochastic Gradient Descent (SGD) or its derivatives operate in continuous feature spaces. Despite the efforts [239], modern Deep Neural Networks (DNNs) remain as largely black-box methods: The learned features are not visualised or interpreted, but the overall performance dictates the model selection and optimisation process. Attempts to create a principled theory explaining *why deep neural networks work so well* are ongoing [145, 233]. Despite these disadvantages, DNNs have become popular in the last decade, since they achieved state-of-the-art results on almost all relevant visual challenges [13].

The next section contains an overview of deep neural networks which are currently leading the field in most computer vision challenges.

2.5 Deep Neural Networks

Machine learning has transformed our lives in unprecedented ways. Systems' capability to learn, without being explicitly programmed, allowed scientists to create data-driven algorithms which perform many everyday tasks. Many innovations in the last two decades, including efficient web search, recommendation systems, autonomous cars, intelligent voice-based personal assistants (e.g. Siri, Alexa) and machine translation, depend on machine learning to accomplish complex tasks. Various fields of Artificial Intelligence (AI) have seen improvements as well, including natural language processing [148, 114], computer vision [107], speech generation [132], machine translation [235], and many more. Deep neural networks are an integral part of machine learning, and they have been extensively reviewed in recent years [190, 135, 73]. In the rest of this section, we summarise Deep Neural Networks (DNNs).

With the increasing availability of GP-GPUs (General Purpose Graphics Processing Unit), deep learning architectures have been applied successfully to many tasks in the vision community [107, 31]. Neural networks with the back-propagation learning procedure have been popular in the last decade [190]. In Section 2.2, we gave examples of flat approaches which can be considered equivalent to a single layer of a multi-layer Convolutional Neural Network (CNN). In order to cascade multiple layers, deep architectures often make use of pooling operations to reduce the complexity of the data between layers [65]. Lin et al. [129] replaced the linear pooling operation in convolutional neural networks with non-linear micro-level neural networks. [129] achieved state-of-the-art performance on CIFAR-10 and CIFAR-100 object classification datasets [106].

The learned features in DNNs can incorporate different channels of information either separately [152, 46] or jointly [204]. In [119], each feature is restricted to a subset of available channels in RGB-D information, which yields features with better generalisation ability. Gupta et al. [74] proposed a rich feature learning framework to learn features from RGB-D images. The depth channel and RGB channels are processed separately in order to learn meaningful features for each data type. The methods listed in this section mostly consist of networks trained on RGB (colour) images, while additional papers on RGB-D data are introduced in the second half of the thesis.

Deep Neural Networks (DNNs) are multi-layer networks which can be trained for a specific task, defined by a loss function. Although their origins can be traced back to the Perceptron by Rosenblatt [180] in 1958, an efficient way to train them was only discovered in 70's, and published in the 80's. Backpropagation algorithm [182] made it possible to define a loss function (e.g. based on the classification task) and tune the weights to minimise the error back-propagated to each node in the network. LeNet by LeCun et al. [115] is a 5-layer fully convolutional neural network trained with backpropagation for handwritten character recognition task. LeNet (1989) is a tiny network compared to today's very deep architectures [210], but its importance comes from being the first successfully trained CNN implementation. DNNs fell out of popularity during 90's and 00's due to a variety of reasons. Feasibility of DNNs were limited due to their high computational needs, the gradient vanishing problem, and the fact that they required large labeled datasets to combat overfitting, all of which were successfully addressed in the last decade. The success of competing feature engineering approaches, coupled with Support Vector Machines (SVM) [34], also contributed to this situation.

The recent insurgence of deep networks can be attributed to the breakthrough paper of Alex Krizhevsky et al. in 2012 [107]. AlexNet [107] is a 8-layer Convolutional Neural Network (CNN), consisting of 5 convolutional layers for feature extraction, and 3 fully-connected layers for classification. It achieved a top-5 classification error of 15.3% in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012), 10.9 percentage points better than the runner-up method (26.2%). AlexNet was trained on a 1.2M images from 1000 classes, used Max Pooling at the end of the convolutional layers and employed

Dropout [84, 207] to avoid over-fitting. AlexNet has since been surpassed in terms of performance [13], but it marks the first time a neural network with millions of parameters was successfully used in a large-scale recognition task.

R-CNN by Girshick et al. [65] showed that combining segmentation and CNNs can achieve state-of-the-art performance in object detection, outperforming others by a wide margin. In [65], candidate regions are extracted with selective segmentation. The bounding box of each region is warped into the same size, and a CNN is used to obtain a fixed-size feature vector for each window, which is given as input to a two-class SVM (yes/no) for each object class. [65] reported an increase in the state-of-the-art of around %30. Girshick later speeded-up R-CNN [67] by running the CNN only once per image and sharing the features across different windows (regions), as well as replacing the SVM classifiers with a neural network. An even faster version of R-CNN was developed by Ren et al. in [171], focusing on simultaneous object boundary and class proposals. Mask R-CNN [78], built on [171], predicts high-quality object segmentation masks in addition to the bounding boxes.

Ultra-deep neural networks have recently been quite popular due to their superior performance [210]. VGG-16 (16 layers) and VGG-19 (19 layers) networks obtained top performances in the ILSVRC-2014 challenge [202]. A crucial difference between AlexNet and VGG architectures is the use of smaller convolutional filters (3×3) throughout the VGG networks. GoogLeNet (22 layers) won the ILSVRC-2014 competition, and its core contribution was the use of the inception modules [211]. The inception modules combine convolutional filters of multiple sizes (e.g. 1×1, 3×3, 5×5) in a single layer, and let the model pick which one to use. Since then, networks have gone even deeper, as evidenced by Microsoft's 152-layer ResNet (won ILSVRC-2015, surpassed human performance) [79], and 1000-layer ResNet tested on the CIFAR-10 dataset [106]. As networks go deeper, they become more difficult to train (the *gradient degradation* problem). ResNet addresses this issue by adding identity connections between the layers, and learning residual mappings in the convolutional layers, instead of fitting a desired mapping to all layers.

In addition to going deeper in the recent years, different network architectures have been proposed as well. Generative Adversarial Networks (GANs) consist of two neural networks competing in a zero-sum game framework [69]. In this game, a generative model G generates samples by mapping a set of latent variables to the image space to deceive a discriminative network D. On the other side, the discriminative network D learns to distinguish between the generated samples and the real data. The aim is to obtain a generative model G which learns the true data distribution, and have D equal to $\frac{1}{2}$ everywhere. Chen et al. proposed InfoGAN [28], which extends GAN by semantically grouping latent variables. The method learns to disentangle pose from lighting in 3D rendered images, writing styles from digit shapes in the MNIST dataset [83], and digits from background in the SVHN dataset [151].

CNNs have yielded good performances in various datasets and tasks, as shown earlier. On the downside, their training involves the estimation of millions of parameters, which requires very large datasets in return. A commonly-used solution to circumvent the dataset requirements of CNNs is to pre-train the network on a related dataset (e.g. ImageNet [38]), and perform fine-tuning by using the intended (usually smaller) dataset on a specific task. For example, Oquab et al. [157] presented a method that enables existing learned neural networks to be used on new datasets by transferring the knowledge. Based on the assumption that the source and target datasets have similar properties, they transferred the layers from a CNN trained on ImageNet [38] to a CNN trained for the Pascal VOC [48] dataset. Using pre-trained weights instead of random initialisation often yields better performance, especially when the training data is scarce [198].

Another important drawback of CNNs is their black-box style processing, which has been addressed by Zeiler and Fergus in [239] in the form of a visual analysis tool for deep learning networks. Their work provides an insight into the hidden nature of deep networks, by visualising the mid-level features and analysing the effect of varying different parameters on the quality of the features. Their method outperformed AlexNet [107] as a result of the improvements based on their analysis. Despite recent efforts to build a mathematical theory of convolutional neural networks [145, 233], their appeal comes from their success in the vision tasks [13], not from our understanding of why they really work so well. However, this is a question that is beyond the scope of this thesis.

In the next chapter, our proposed CHOP architecture is presented. The links between hierarchical compositional methods and deep neural networks are established in the final chapter.

CHAPTER 3

LEARNING HIERARCHICAL SHAPE REPRESENTATIONS

In this chapter, a summary of our previous work [2, 158] on compositional shape hierarchies is given. The proposed method, Compositional Hierarchy of Parts (CHOP), is a system that learns a hierarchical set of 2D shape features from images. The data at each layer of the hierarchy is represented using a set of graphs, where *parts* (frequently occurring 2D shapes) are considered as common cliques in these graphs. The graphs consist of a set of nodes and directed edges, where nodes correspond to shape instances, and edges encode spatial relationships. Each training image has its own *image graph*, where the information in the image is encoded in a 2-dimensional lattice. At the first level, graph nodes represent feature responses, and the directed edges link neighbouring nodes in *receptive fields*. In higher layers, the graph nodes, called *part realisations*, correspond to instances of abstract structures in the real data. The image graphs are compressed recursively across multiple layers by discovering frequent patterns in the data. In order to remove redundancy at each layer, a max-pooling operation is performed.

The proposed method, CHOP, is a hybrid generative-descriptive model for 2D shape representation learning. Unlike other hierarchical compositional methods [103, 247, 174, 58], CHOP measures the generative and descriptive properties of object parts in a principled, graph theoretic approach. Each part in the vocabulary is a structured tree graph, and and the instances of a part can be *generated* from the corresponding graph by sampling. Selecting parts for the next layer in the vocabulary is based on Occam's Razor principle [123], which favours the simplest models that explain the data. As a result, the learned multi-layer vocabulary of shapes consists of elements which are most *descriptive* (given the dataset). In the lower layers of the hierarchy, parts are often shared among different objects and classes. As the learning progresses to higher layers, vocabulary parts tend to get more specialised, allowing for fast indexing. The inference of the parts in the vocabulary can be performed efficiently, exploiting the graph-based shape matching and indexing features.

Compositional Hierarchy of Parts (CHOP), relies on the mechanisms proposed by Learned Hierarchy of Parts (LHOP) by Fidler et al. [58]. Both are contour-based hierarchical shape learning methods. Pair-wise spatial part distributions, part compositionality and shareability, unsupervised training and indexing-based inference can be considered as the commonalities between the two frameworks. The learning of vocabulary layers is performed in a feed-forward manner in both, although LHOP has a fine-tuning stage where each part's spatial statistics are updated in a top-down pass. LHOP relies on part frequency and data coverage for part selection, whereas CHOP has an MDL-based approach that aims to compress the data graphs. CHOP can be considered as a re-formulated and updated version of LHOP which is based on graph theory and compression. This re-formulation leads to a unified framework where learning and inference mechanisms can be defined in terms of graph-theoretic algorithms.

The information flow of the proposed system is given in Figure 3.1. The first layer $l = 1$ can be considered as a pre-processing step which extracts line-based features from the dataset. Object colours and intensity values are filtered out from all images, leaving behind the quantised edge structures which together constitute the shapes. In the *learned* higher layers, random graphs represent parts (partial shapes), and parts' instances in the input dataset are referred to as part realisations. Layer-wise training starts by learning the spatial statistical relationships between the parts, using a combination of *Minimum Conditional Entropy Clustering* (MCEC) method [124] and 2D Gaussian distributions.

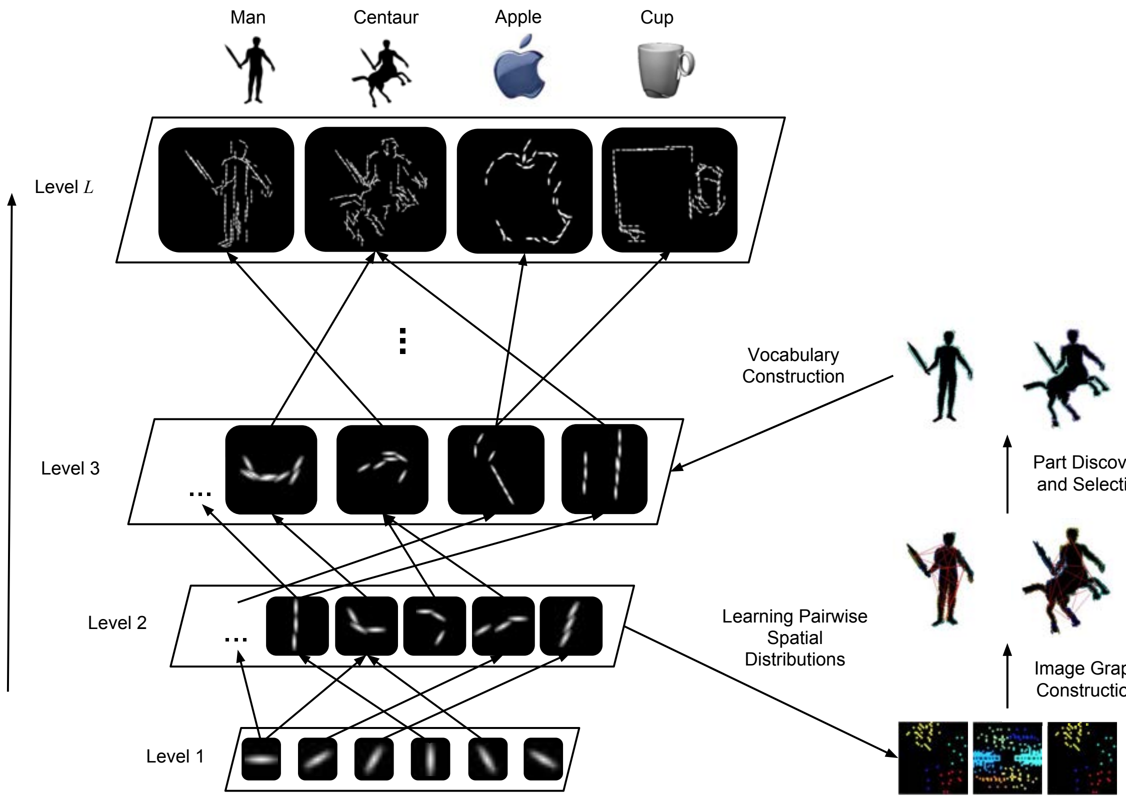


Figure 3.1: The left side of the figure shows a hierarchical vocabulary of parts, while the right side shows key components of learning. Left: Each layer of the vocabulary consists of a number of parts which are compositions of the parts from the previous layer. Right: Layer-wise learning starts by learning the pairwise spatial distributions among the parts. A graph-based representation, termed *image graph*, is created for every shape. Finally, the method selects a common set of frequent subgraphs as the parts which constitute the next layer of the vocabulary.

This step involves estimating the probability distributions governing the relative spatial positioning of two parts $\mathcal{P}_i$ and $\mathcal{P}_j$, for all part pairs in the vocabulary. All possible realisations of these parts throughout the dataset are considered. The distributions specifying the relative positioning of parts, thus representing geometry, are encoded in the edge labels in the graph structure. An higher layer's parts are built by creating compositions of the lower layer's parts in the form of connected subgraphs. Each part composition is evaluated by its ability to compress the dataset's representation at the lower level, using the Minimum-Description Length (MDL) principle [175]. We employ a greedy scheme to select a compact set of parts for the higher layer, by iteratively selecting parts which minimise the conditional description length of the remaining dataset representation at each step. The instances of the selected parts are discovered via subgraph isomorphism. These steps are recursively applied until no more compositions can be learned.

3.1 Definitions

Below, the basic definitions of our framework is given. In the rest of this chapter, a *layer* is considered as a set of parts or hidden nodes which are at the same *level* l of the hierarchy, i.e. $l = 3$.

Definition. Vocabulary A hierarchical compositional shape vocabulary $\Omega = \{\Omega^l\}_{l=1}^{L}$ consists of multiple vocabulary layers Ω^l, s.t. $0 < l < L$. Figure 3.1 illustrates some of the vocabulary layers learned by our system, along with key concepts of learning. Each vocabulary layer $\Omega^l = (\mathcal{P}^l, \mathcal{M}^l)$ contains a set of parts $\mathcal{P}^l$ and a set of modes $\mathcal{M}^l$. Where possible, each level-specific part set $\mathcal{P}^l$ is called a *layer* for brevity. A layer $\mathcal{P}^l$ consists of *parts*, hierarchical shape representations that are combinations/compositions of the parts in the previous layer $\mathcal{P}^{l-1}$, with the exception of the first layer $\mathcal{P}^1$. $\mathcal{P}^1$ contains $\theta = 6$ parts $\{\mathcal{P}_i^1\}_{i=1}^{\theta}$, each of which corresponds to a different edge type, as explained in Section 3.3.1. The set of modes $\mathcal{M}^l$ includes information relevant to the relative positioning of the parts, and will be explained later in the chapter.

Definition. Part A part $\mathcal{P}_i^l$ in a layer $\mathcal{P}^l = \{\mathcal{P}_i^l\}_{i=1}^{A_l}$ is an abstract entity that represents a (possibly partial) shape of a specific complexity. The parts in the hierarchy are defined as hierarchical tree structures that can be deformed in *learned ways* to account for the variabilities in the data. The i^{th} part in layer $\mathcal{P}^l$ at the l^{th} level is defined as $\mathcal{P}_i^l = (\mathcal{G}_i^l, \mathcal{Y}_i^l)$. $\mathcal{G}_i^l = (\mathcal{V}_i^l, \mathcal{E}_i^l)$ is the graph description of the part $\mathcal{P}_i^l$, where $\mathcal{V}_i^l$ is the set of nodes (parts) from the previous layer, and $\mathcal{E}_i^l$ is a set of directed edges encoding spatial relationships between the nodes. In higher layers $l > 1$, the graph description $\mathcal{G}_i^l$ is a composition of the nodes from the previous layer $\mathcal{P}^{l-1}$. At the first level $l = 1$, both node set $\mathcal{V}_i^l = \emptyset$ and edge set $\mathcal{E}_i^l = \emptyset$ are empty, $\forall i$. As explained in the previous section, first layer parts are considered to be the leaf nodes at the bottom of the hierarchy, hence they do not contain any graph descriptions. $\mathcal{Y}_i^l \in \mathbb{Z}^+$ is part label or identifier, which is explained in Section 3.3.5.

Definition. Part Realisation A part realisation is an instance of a part in the real data. It represents a possibly deformed instance of a part in an image. Formally, a part realisation $R_j^l(n,m)$ in image s_n, at location m and level l is defined as $R_j^l(n,m) = (G_j^l(n,m), Y_j^l)$. In this definition, Y_j^l is the index of the corresponding part $\mathcal{P}_i^l = (\mathcal{G}_i^l, \mathcal{Y}_i^l)$, s.t. $Y_j^l = i, \exists \mathcal{P}_i^l \in \mathcal{P}^l$. $G_j^l(n,m)$ is a subgraph in s_n's image graph *at the previous level*, and is isomorphic to the corresponding part's graph description $\mathcal{G}_i^l$. Similarly with part description, the part realisation graph $G_j^l(n,m) = \{V_j^l(n,m), E_j^l(n,m)\}$ contains a set of nodes $V_j^l(n,m)$ and a set of directed edges $E_j^l(n,m)$. As a result of the isomorphism, $R_j^l(n,m)$ is an instantiation of the part $\mathcal{P}_i^l$ in the image s_n, at a specific location m. For brevity, image index n and location m will omitted from the notation where possible, in the rest of this document. We refer to the index of the image of a part realisation $R_j^l(n,m)$ as n_j^l, and its position as m_j^l.

The first level part realisations in $R^1 = \{R_j^1\}_{j=1}^{B_1}$ have empty node and edge sets, with B_1 being the number of part realisations in the first level. Each part realisation R_i^1 in R^1 corresponds to a feature response $f_{n,m}^i \in \hat{F}$, where $\hat{F}$ is the set of edge-like features at the first level of the hierarchy. As a result, there is a one-to-one and onto correspondence between R^l and $\hat{F}$. In other words, every feature response becomes a data point (part realisation) at the first level. In the consecutive layers where $l > 1$, parts consist of combinations of the parts in the previous layer.

The data obtained from the images, $\hat{F}$, is recursively compressed through a learned vocabulary Ω. The part realisations at the l^{th} level, $R^l = \{R_j^l\}_{j=1}^{B_l}$, correspond to the compressed version of the data $\hat{F}$ using $\{\Omega^j\}_{j=1}^{l}$. B_l represents the number of realisations at the l^{th} level of the hierarchy.

Definition. Receptive Field A receptive field (RF_j^l) is a circular area around a part realisation R_j^l, which defines the connectivity between the centre node R_j^l and any other realisation in its neighbourhood R_k^l, s.t. $R_k^l \in \aleph(R_j^l), \exists j, k \in \mathbb{Z}^+, j \neq k$. Simply put, two nodes R_j^l and R_k^l are connected if they are in the receptive field of each other. The receptive field size is constant throughout the hierarchy and is defined by a radius *rad* for every part in every layer. In higher layers, the part realisations' positions are downsampled while receptive field radius *rad* is kept fixed, hence the receptive fields cover a larger area in the actual images. Downsampling increases the visual field and complexity of the parts in the vocabulary.

Definition. Receptive Field Graph A receptive field graph ($RG_j^l(n,m)$) of a part realisation $R_j^l(n,m)$ is the star-shaped graph which is inferred from the Receptive field RF_j^l of a part realisation $R_j^l(n,m)$. The peripheral nodes in $RG_j^l(n,m)$ consist of all part realisations in RF_j^l, except for $R_j^l(n,m)$. Receptive field graphs do not contain any loops. They only include directed edges between the centred node $R_j^l(n,m)$ and peripheral nodes, and the edges originate from the centre.

In this document, a *graph* contains nodes (vertices) and directed edges. The nodes can either be parts or their realisations in the images. The edges encode the spatial (geometric) relationships between the nodes. In the case of part realisations, a directed edge $e_{jk}^l \in RG_j^l$ is defined as

$$e_{jk}^l = \begin{cases} (R_j^l, R_k^l, \phi_{jk}^l), & \text{if} \quad R_k^l \in \aleph(R_j^l), j \neq k \\ \emptyset, & otherwise \end{cases}, \tag{3.1}$$

where $\aleph(R_j^l)$ is the neighbourhood function that specifies the part realisations that reside in the receptive field RF_j^l of the part realisation R_j^l in an image s_n, $\forall R_j^l, R_k^l \in R^l, j \neq k$. ϕ_{jk}^l defines the spatial statistical relationship between R_j^l and R_k^l, as explained in Section 3.3.2. The edge e_{jk}^l can be used to refer to the parameters of the estimated probability distribution that the relative spatial positioning of R_j^l and R_k^l is sampled from. Instead of using the difference of their coordinates as the edge label ϕ_{jk}^l, the system learns *patterns* of differences of coordinates among similar pairs, and discretises the edges by grouping similar cases. This information is used to obtain discrete labels for the edges, and the labels are stored in ϕ^l_{jk} for every edge. The reader is advised to refer to Section 3.3.2 to learn more about how these statistics are collected and used.

Definition. Image/Data Graph The data from each image s_n is represented by the corresponding Image Graph $\mathbb{G}^l_n$ across all layers of the hierarchy. The image graph of an image s_n is the union of all of the receptive field graphs in s_n, where nodes are shared among overlapping receptive fields. Formally, the image graph $\mathbb{G}^l_n$ of an image s_n at the l^{th} level is defined as $\mathbb{G}^l_n = \bigcup\limits_j RG^l_j(xm)$, $\exists x\ n = x$.

The image graph of image s_n, $\mathbb{G}^l_n = (\mathbb{V}^l_n, \mathbb{E}^l_n)$, consists of a node list $\mathbb{V}^l_n$ and an edge list $\mathbb{E}^l_n$. Node set $\mathbb{V}^l_n$ includes all part realisations that belong to the image s_n at the l^{th} level, s.t. $\mathbb{V}^l_n = \{R^l_j(xm)\}$, $\exists j R^l_j(xm)$, $x = n$. The image graphs belonging to two different images are disjoint, as each image has its own realisation set, and there are no overlaps between them. The data graph at level l, $\mathbb{G}^l = \{\mathbb{G}^l_n\}^{N_{tr}}_{n=1}$, is the union of all image graphs at level l (N_{tr} is the number of training images). $\mathbb{G}^l$ is the *lattice* or *data graph*, which is the graph that encodes the data in all images at the level l. The method compresses $\mathbb{G}^l$ recursively by finding frequent subgraphs in $\mathbb{G}^l$, which constitute the parts in the next layer $\mathcal{P}^{l+1}$. The details of the compression procedure can be found in Section 3.3.3.

3.2 Overview

Compositional Hierarchy of Parts (CHOP [2]) is an unsupervised learning framework for 2D object shape representations. The method is built on the principle recursive data compression using structures of similar complexity. The first step is to build graph representations of the input shapes, where the nodes correspond to line-based basic features, and edges encode spatial relationships. A multi-layer vocabulary is learned by finding sets of frequent subgraphs in the representations. Each layer is trained by compressing the data coming from the previous layer. A layer is a level of the hierarchy containing

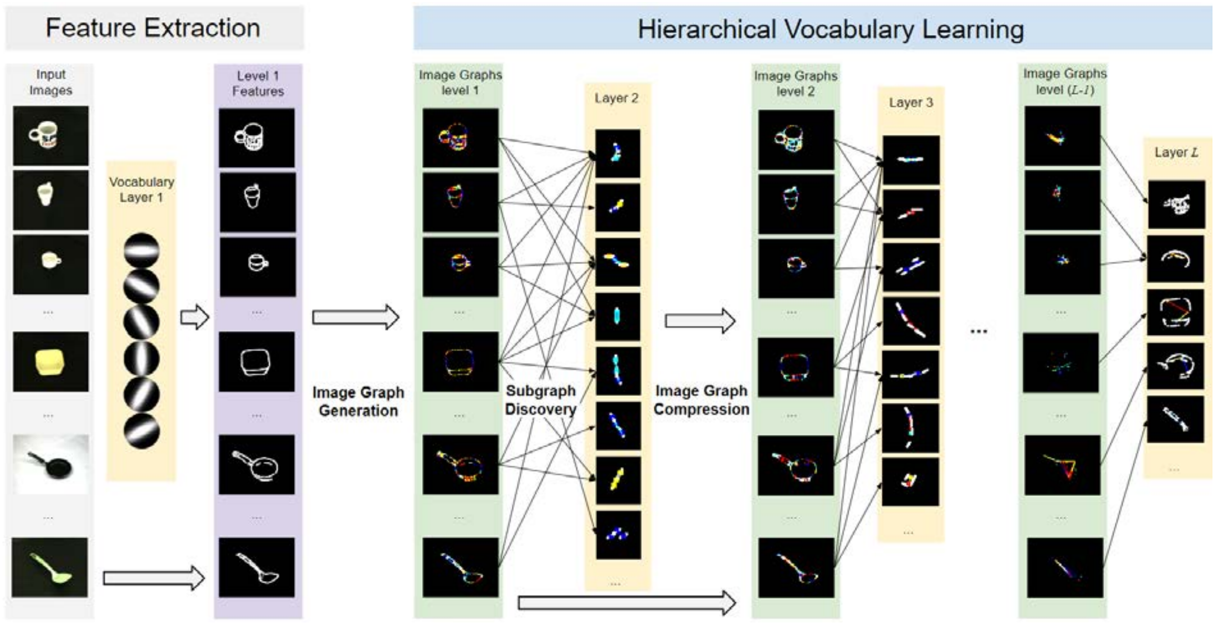


Figure 3.2: CHOP System architecture.

shape parts of similar complexity. The layers are built based on the Minimum Description Length (MDL) principle [175]: The parts which compress the data best are included in a layer. In this section, the proposed architecture is briefly explained. The recursive learning algorithm is summarised in Section 3.2.1, while the inference procedure for test images is given in Section 3.2.2.

3.2.1 Vocabulary Learning

A vocabulary of shapes $\Omega = \{\Omega^l : l = 1, 2, \ldots, L\}$, or parts, is learned iteratively. Each layer of the vocabulary Ω^l consists of compositions of the parts from the previous layer, $\mathcal{P}^{l-1}$, with the exception of Ω^1. Figure 3.2 shows the system architecture. Learning starts by using edge filters for extracting line-like features from the training images. Next, each image is converted into an image graph, where the visual features constitute the nodes, and the edges encode spatial relationships. Learning of the hierarchical shape vocabulary is performed recursively by finding frequently occurring subgraphs in the image graphs and compressing them using these subgraphs. Key steps of the learning algorithm are outlined in Algorithm 1.

```
    Input  : Training images $S^{tr}$.
    Output: Hierarchical shape vocabulary $\Omega$.
 1  A set of suppressed features $\hat{F}$ are extracted from training images $S^{tr}$ (Section
      3.3.1);
 2  Parts $\mathcal{P}^l$, realisations $R^l$ and data graph $\mathbb{G}^l$, where $l = 1$, is constructed from $\hat{F}$
      (Section 3.1);
 3  Edges of the image graphs $\mathbb{G}^l$ are discretised (Section 3.3.2), yielding modes $\mathcal{M}^l$;
 4  Construct the vocabulary layer $\Omega^l$ from $\mathcal{P}^l$ and $\mathcal{M}^l$;
 5  Candidate parts $\mathcal{G}^l$ are discovered by mining $\mathbb{G}^l$ (Section 3.3.3);
 6  Part selection is performed to select the next level's parts $\mathcal{P}^{l+1}$ from $\mathcal{G}^l$, and their
      realisations $R^{l+1}$ (Section 3.3.4);
 7  Parts in $\mathcal{P}^{l+1}$ are grouped based on their appearance similarities (Section 3.3.5),
 8  Positions of $R^{l+1}$ are downsampled by $\sigma$;
 9  Max pooling is performed in $R^{l+1}$ (to reduce the number of part realisations);
10  $\mathbb{G}^{l+1}$ is generated from $\mathcal{P}^{l+1}$ and $R^{l+1}$ (Section 12);
11  $l := l + 1$, go back to step 3.
```
Algorithm 1: Learning the hierarchical vocabulary.

The purpose of training is to learn a hierarchical shape vocabulary $\Omega = \{f^l : l = 1, 2, \ldots, L\}$. The steps of the learning procedure are explained in the following sections. The process continues until no new layers are discovered. In step **8**, the positions of R^{l+1} are downsampled (usually by $\sigma = 2$), which effectively increases the size of the receptive field across higher layers. This allows us to learn more complex structures. Step **9** ensures that the number of realisations are kept at a manageable level. Max pooling suppresses overlapping responses among part realisations that have the same identifier, position and image id. Once the mining process is complete, a multi-layer vocabulary is obtained as Ω. Next section contains an overview of the test image inference procedure, which uses the data structures explained previously.

3.2.2 Test Inference

Multi-layer inference in a test image follows the same principles and mechanisms as vocabulary learning. The learned vocabulary parts are *inferred* in the test image's graph representation. The inference of vocabulary parts is a layer-by-layer process that makes use of part indexing (parent-child relationships) in order to detect learned parts' realisa-

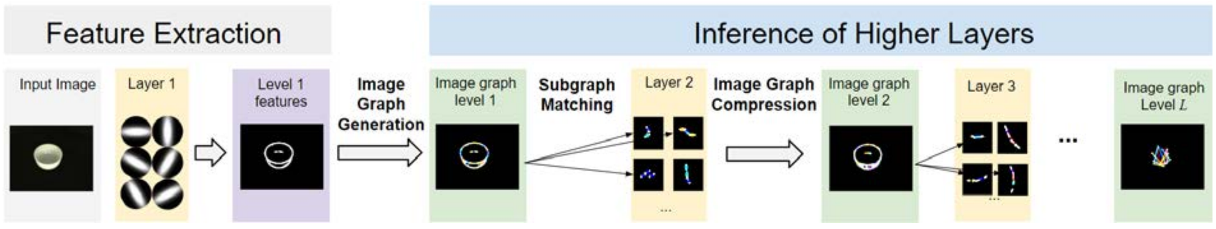


Figure 3.3: Inference in test images. The inference procedure runs in a very similar manner to the training. When moving from one layer to another, modes $\mathcal{M}^l$ or parts $\mathcal{P}^l$ are obtained from the learned vocabulary Ω.

```
  Input  : Test image $S^{te}$.
  Output: Set of realisations $R = \{R^l\}_l^L$
1 A set of suppressed features $\hat{F}^{te}$ is extracted from $s_{te}$,
2 The realisations $R^l$ and image graph $\mathbb{G}^l$, where $l = 1$, is constructed from $\hat{F}^{te}$,
3 Edges of $\mathbb{G}^l$ are discretised using learned modes $\mathcal{M}^l$,
4 Isomorphisms of $\mathcal{P}^{l+1}$ are searched in $\mathbb{G}^l$, which results in $R^{l+1}$,
5 Positions of $R^{l+1}$ are downsampled by $\sigma$,
6 Max pooling is performed in $R^{l+1}$ (to reduce $|R^{l+1}|$),
7 $\mathbb{G}^{l+1}$ is constructed from $\mathcal{P}^{l+1}$ and $R^{l+1}$,
8 $l := l + 1$, go back to step 3.
```

Algorithm 2: Inference in a test image.

tions in a sample image. It can be considered analogous to recursively compressing an image using a hierarchical vocabulary of visual words. Figure 3.3 provides a description of the recursive compression process. Given a test image s_{te} and the learned vocabulary $\Omega = \{\Omega^l : l = 1, 2, \ldots, L\}$, the high-level flow of inference is given in Algorithm 2.

The inference process starts with the same procedures as learning (Steps **1-3**). In step **3**, the discretisation process uses the duplet distributions learned in Section 3.3.2. This ensures that test images and training examples share the same connection types. Realisations R^{l+1} of next layer's parts $\mathcal{P}^{l+1}$ are searched in test image graph $\mathbb{G}^l$ in step **4**, using subgraph isomorphism. Steps **5** and **6** are performed in order to increase the size of the receptive field and reduce the number of realisations, respectively. The inference procedure continues until no new parts can be found in the test image.

3.3 Compositional Hierarchy of Parts (CHOP)

This section explains the algorithms underlying the proposed method, Compositional Hierarchy of Parts (CHOP). Following the high-level introduction of the previous chapter, the building blocks of the system are explained, starting with the extraction of line-like edge features in Section 3.3.1. Next, the *patterns* of the spatial relationships between the features in Section 3.3.2 are learned. The part discovery process which is at the core of the method is given in Section 3.3.3, followed by the selection procedure in Section 3.3.4. Next, the part grouping process which groups similar parts together is explained in Section 3.3.5. The section concludes with the reconstruction of the next level's parts and realisations in Section 3.3.6.

3.3.1 Feature Extraction

The method for learning hierarchical shape vocabularies starts by processing the training images using pre-defined, polarised edge-detection filters in order to extract low-level features. Each filter corresponds to a different edge direction. The filtering operation allows us to decompose the shape into local edge segments which jointly represent the input shape. Smooth lighting transitions and the colour of an object, while they may provide relevant information, are ignored because they do not contribute to the edge information.

Given a set of training images $S^{tr} = \{s_n\}_{n=1}^{N_{tr}}$, a set of filtering operations $\mathbf{f} = \{\mathbf{f}_i\}_{i=1}^{\theta}$ are performed on each image s_n in order to extract its edge features. These features constitute the first layer of the shape hierarchy. In this thesis, $\theta = 6$ oriented polarised filter pairs are used in order to detect edges in 6 different edge directions in an image, as shown in Fig. 3.4. In the figure, the filters (top row) are paired with their inverted versions (bottom row), and they are separated by 30 degrees to quantise the possible edge directions into $\theta = 6$ bins. The results of the convolution of each filter pair with the image are superimposed onto the pair's result image, thus creating 6 unique images in total. The advantage of pairing is that the detected edges are invariant to the secondary direction of the edge, which is perpendicular to the main edge direction. In other words, the feature activations are invariant to the actual intensity values on both sides of an edge. It is the absolute value of the difference that matters.

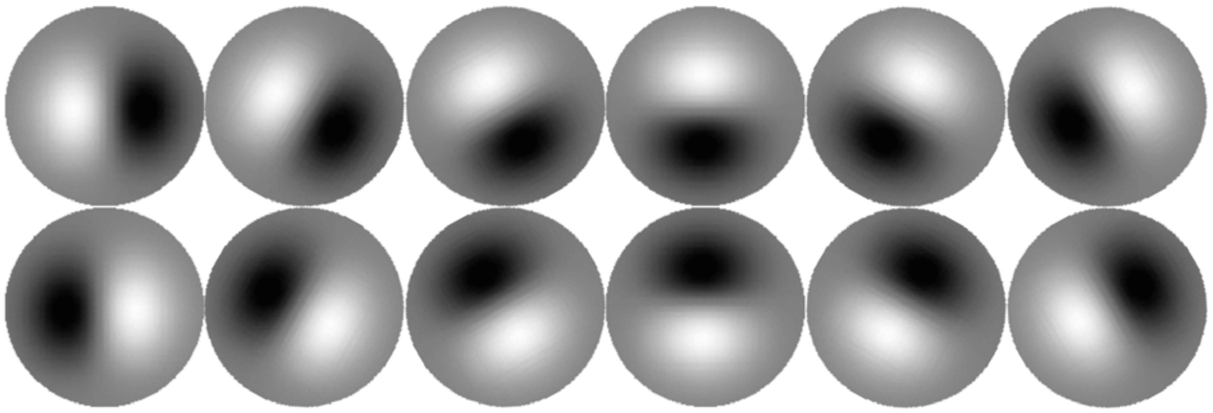


Figure 3.4: The filter pairs used in feature extraction. Top row shows original filters, and bottom row shows their inverted pairs. The filters are convolved with an image and the output of paired filters are superposed to obtain 6 unique response images.

Formally, when a training image s_n is processed with the filtering operations in $\mathbf{f}$, a set of feature responses $F_n = \{F_n^i\}_{i=1}^{\theta}$ are obtained, where $F_n^i = \{f_{n,m}^i \in \mathbb{R}\}_{m=(1,1)}^{(M_{nr},M_{nc})}$. F_n^i is the set of filter responses in the i^{th} edge direction in image s_n. $f_{n,m}^i$ is the response in the i^{th} edge direction in image s_n and at pixel location m. The convolution operation extracts responses from all locations in an image, with M_{nr} being the number of rows in the training image s_n, and M_{nc} specifying the number of columns. The set of initial filter responses from all edge directions in all images is constructed as $F = \bigcup_{n=1}^{N_{tr}} F_n$. Then, the redundancy encoded in F is removed with non-maxima suppression, obtaining $\hat{F} = \bigcup_{n=1}^{N_{tr}} \hat{F}_n$.

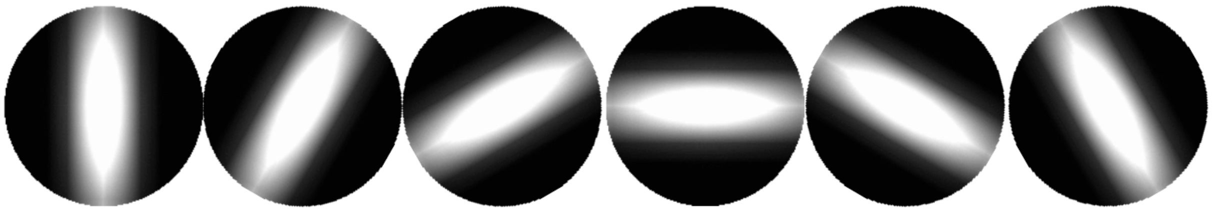


Figure 3.5: Gabor filters used for visualisation. Each filter pair in the filtering operations $\mathbf{f}$ is illustrated with a visualisation filter in the same direction. The visualisation filters are ordered to match the original feature extraction filters in Figure 3.4.

Within a suppressed image feature set $\hat{F}_n \subset F_n$, the local peaks hinder weaker responses in their local neighbourhood $\aleph(m)$, defined by Euclidean distance in $\mathbb{R}^2$ (since images are two dimensional). Additionally, $\hat{F}_n$ only includes features where the feature response is greater than a pre-defined threshold T, i.e. $f^i_{n,m} > T$, $\exists i, m$. Thresholding ensures that only strong edge responses are taken into account. Finally, a new set of features $\hat{F} = \bigcup_{n=1}^{N_{tr}} \hat{F}_n$, where $\hat{F}_n \subset F_n$, is obtained. higher layers are recursively learned based on this processed information.

The shape vocabulary Ω is based on simple line segments in layer 1, and parts $\mathcal{P}^{l+1}$ at level $l+1$ are built based on the previous layer's elements $\mathcal{P}^l$. Because of this recursive formulation, the relative complexity of consecutive layers throughout the shape vocabulary Ω remains constant.

In the rest of this document, for the visualisation of features, Gabor filters will be used. This choice stems from the fact that the filtering operations $\mathbf{f}$ are designed to extract edges of specific directions, and 2D Gabor filters can represent local edge segments. Figure 3.5 illustrates the 6 corresponding Gabor filters selected for visualisation.

3.3.2 Learning Pairwise Spatial Distributions of Parts

CHOP builds a hierarchical, contour-based shape vocabulary, where each layer of the vocabulary depends on the previous layer. Vocabulary layers are learned by discovering frequent subgraphs in the lattice $\mathbb{G}^l$, which represents the data compressed with the vocabulary layers $\{\Omega^i\}_{i=1}^{l}$. The directed edges $\mathbb{E}^l$ which connect the part realisations in $\mathbb{G}^l$ encode the spatial relationships between neighbouring realisations. While exact positions may be useful in the reconstruction of an object, the *patterns* of relative positions are of special interest here. The statistical information between two parts are learned by clustering their instances' (realisations) arrangements and encoding the information in $\mathbb{G}^l$ in terms of edge labels.

The system aims to learn the underlying distribution that models the relative spatial positioning of part $\mathcal{P}^l_j$ with respect to another part $\mathcal{P}^l_i$, for all possible part pairs $\forall i, j$. For example, let us consider $\mathcal{P}_i^l$ as the *centre* or *reference* part, and $\mathcal{P}_j^l$ as the peripheral part. $\mathcal{P}_i^l$ is at the centre of the receptive field. The spatial relations between these two parts $\mathcal{P}_i^l$ and $\mathcal{P}_j^l$ are recorded by using a point set $\mathcal{X}_{ij}$ which contains the relative positions of pairs of corresponding samples collected from the dataset. The samples are drawn from the pixel locations in a square that tightly contains the circular receptive field, which spans the coordinates $[-rad, rad] \times [-rad, rad]$. rad is the receptive field radius. Figure 3.6 shows the relationship between a receptive field and the collected samples $\mathcal{X}_{ij}$. There is no data coming from outside the receptive field, illustrated with grey pixels.

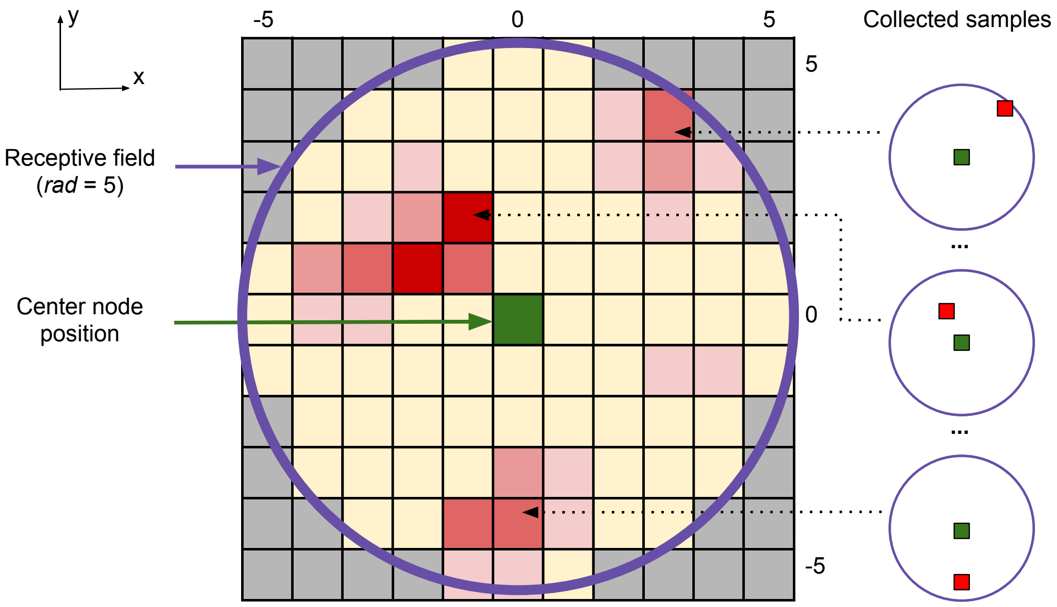


Figure 3.6: A point set $\mathcal{X}_{ij}$ collects samples from all receptive fields in the data. The centre part $\mathcal{P}_i^l$ is placed at the centre of the receptive field. The set records where peripheral nodes are located by recording each instantiation of $\mathcal{P}_j^l$ in the receptive field of $\mathcal{P}_i^l$ in the data. In the shown visualisation, darker shades of red means more samples are included in the set. The grey areas are outside the receptive field. (Best viewed in colour)

Collecting the points in the set $\mathcal{X}_{ij}$ requires a full pass over the training set. All receptive fields over all training images contribute to the statistics collection process. Consider two part realisations R_x^l and R_y^l, which are instances of parts $\mathcal{P}_i^l$ and $\mathcal{P}_j^l$, respectively. If R_y^l is in the receptive field of R_x^l, which requires them being in the same image graph and in the vicinity of each other, they contribute to the set $\mathcal{X}_{ij}$. The relative position of R_y^l with respect to R_x^l is calculated as the difference of their positions $m' = m_y^l - m_x^l$. m'

takes values from $\{-rad, \ldots, 0, \ldots rad\} \times \{-rad, \ldots, 0, \ldots rad\}$. m' is added to the set $\mathcal{X}_{ij}$. After all such pairs have been added to the set, the next step is to group/cluster the collected samples, as given below. In the rest of this section, $\mathcal{X}_{ij}$ will be referred to as $\mathcal{X}$, in order not to overcrowd the notation.

1. As explained above, the relative spatial data obtained from the part pair $\mathcal{P}_i^l$ and $\mathcal{P}_j^l$ is aggregated into the set $\mathcal{X} = \{x_i | i = 1, \ldots, n\}$, $x_i \in \{-rad, \ldots, 0, \ldots rad\} \times \{-rad, \ldots, 0, \ldots rad\}$. In order to analyse $\mathcal{X}$, a straightforward approach would be to use a clustering technique such as K-means [49], but that would necessitate knowing the number of clusters k in advance. Since it is not possible to know k for every set $\mathcal{X}$ in advance, Minimum Conditional Entropy Clustering (MCEC) [124] is applied to find k automatically.

   Formally, the clustering problem can be stated as follows: Given the dataset $\mathcal{X} = \{x_i | i = 1, \ldots, n\}$ and an integer $c_0 > 1$, the task is to map each element of $\mathcal{X}$ onto a cluster label in a set $\mathcal{C} = \{\mathcal{C}_i | i = 1, \ldots, c_0\}$. Each data point represents the edge (connection) between two part realisations R_x^l and R_y^l, and is assigned a cluster id $\mathcal{C}_i$ at the end of the clustering process. While clustering is a well-defined problem, the key difficulty is finding the correct number of clusters.

   MCEC works by optimising the following criterion using an iterative algorithm:

$$H(\mathcal{C}|x) = -\int \sum_{j=1}^{c_0} p(\mathcal{C}_j|x) \log p(\mathcal{C}_j|x) p(x) dx \tag{3.2}$$

   where $\mathcal{C}$ is the random variable of the edge type taking values in $\{\mathcal{C}_i | i = 1, \ldots, c_0\}$, given the random variable x, which represents the input data points. The conditional entropy $H(\mathcal{C}|x)$ is the uncertainty of a label $\mathcal{C}$ once the input x is known. In order to perform clustering using MCEC, first, the data $\mathcal{X} = \{x_j | j = 1, \ldots, n\}$ is clustered using K-means, to obtain an initial group label $C_0 = \{c_j | j = 1, \ldots, n\}$ for every data point, where $c_j \in \{\mathcal{C}_i | i = 1, \ldots, c_0\}$, $\forall j$. c_0 is the maximum number of clusters, which is set heuristically to a high number (set to 10 in this thesis). MCEC iteratively merges clusters $\mathcal{C} = \{\mathcal{C}_i | i = 1, \ldots, c_0\}$ by optimising $H(\mathcal{C}|x)$ and finds a local solution. The algorithm avoids trivial solutions such as putting all data points in a single cluster by using an initial partitioning and an iterative solver.

2. Once the clusters are discovered, a cluster label c_i is obtained for every data point x_i. The final number of clusters c is equal to or smaller than c_0. A 2D Gaussian is fit to the samples of every cluster in the final cluster set $\mathcal{C}_{final} = \{\mathcal{C}_k | k = 1, \ldots, c\}$. Thus, for a part pair $\mathcal{P}_i^l$ and $\mathcal{P}_j^l$ and cluster $\mathcal{C}_k$, the system learns a Gaussian function $g_{ijk}^l = (\mu_{ijk}^l, \Sigma_{ijk}^l)$, where μ_{ijk}^l is the mean and Σ_{ijk}^l is the covariance matrix of the samples in the cluster. These Gaussians are called the *modes* of the distribution that represents the relative positioning of $\mathcal{P}_j^l$ with respect to $\mathcal{P}_i^l$. To find the edge label ϕ_{xy}^l between two connected realisations R_x^l and R_y^l, $\phi_{xy}^l = \hat{k}$ is computed, $\forall x, y$ as $\hat{k} = \arg\min_{k \in \mathcal{C}_{final}} D(m', \mu_{ijk}^l, \Sigma_{ijk}^l)$, where $D(m', \mu_{ijk}^l, \Sigma_{ijk}^l) = (m' - \mu_{ijk}^l)^T (\Sigma_{ijk}^l)^{-1} (m' - \mu_{ijk}^l)$ is the Mahalanobis distance between m' and μ_{ijk}^l, $m' = m_y^l - m_x^l$ is the relative position of R_y^l with respect to R_x^l, m_x^l and m_y^l are the positions of R_x^l and R_y^l, respectively. As a result, each sample is assigned to the closest cluster (according to Mahalanobis distance) and the cluster label is encoded in the lattice $\mathbb{G}^l$ as the relevant edge's label.

Once the edge labels in the lattice $\mathbb{G}^l$ are set, the next step is to find frequent subgraphs (parts) in $\mathbb{G}^l$, which constitute the parts in the next layer. A detailed explanation of the part discovery process is given in the next section.

3.3.3 Part Discovery

Given the sets of parts $\mathcal{P}^l$, part realisations R^l and image graphs $\mathbb{G}^l$, higher layer parts $\mathcal{P}^{l+1}$ and realisations R^{l+1} can be *discovered* by frequent subgraph discovery. Parts and their realisations are discovered recursively using layer-to-layer mappings $\Psi_{l,l+1}$, as defined in Equation (3.3):

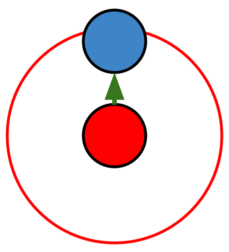
$$\Psi_{l,l+1} : (\mathcal{P}^l, R^l, \mathbb{G}^l) \rightarrow (\mathcal{P}^{l+1}, R^{l+1}), \forall l = 1, 2, \ldots, L-1, \tag{3.3}$$

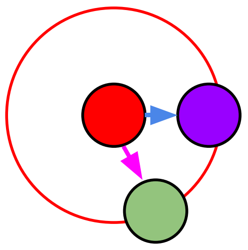
where $\mathcal{P}^l = \{\mathcal{P}^l_i\}_{i=1}^{A_l}$, $R^l = \{R^l_i\}_{i=1}^{B_l}$, $\mathcal{P}^{l+1} = \{\mathcal{P}^{l+1}_i\}_{i=1}^{A_{l+1}}$, $R^{l+1} = \{R^{l+1}_i\}_{i=1}^{B_{l+1}}$ and $\mathbb{G}^l$ is the lattice (data graph). A_l and B_l are the number of parts and realisations at level l, respectively. The mapping procedure $\Psi_{l,l+1}$ essentially involves discovering frequently occurring subgraphs in $\mathbb{G}^l$, and compressing $\mathbb{G}^l$ using these subgraphs. In order to discover the next layer parts $\mathcal{P}^{l+1}$, first, a set of candidate subgraphs $\mathcal{G}^l = \{\mathcal{G}^l_j\}_{j=1}^{N^l_S}$ which are frequent in $\mathbb{G}^l$ are enumerated. Only star-shaped subgraphs are considered to ensure the enumerated subgraphs fit in a circular receptive field (see Figure 3.7a-c for examples). Second, a selection method is run to find the *best* set of subgraphs which compress the lattice $\mathbb{G}^l$. Third, a more compact set of parts are selected by a second procedure, as explained in Section 3.3.4. Finally, the next level's parts and realisations are constructed. In the rest of this section, the details about the first two stages of $\Psi_{l,l+1}$ are given, namely, subgraph enumeration and evaluation.

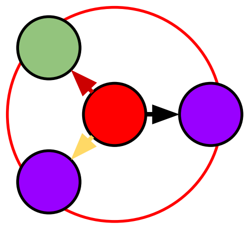
Enumeration of Next Layer Parts

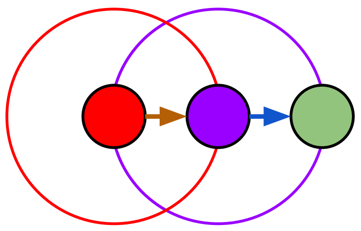
In the graph enumeration step, candidate graphs $\mathcal{G}^l$ are generated from $\mathbb{G}^l$. Connected subgraphs are of interest, and there is a limit on the maximum number of nodes contained in a subgraph (see experiments for details). A candidate subgraph of size n, $\mathcal{G}^l_j = \{\mathcal{V}^l_j, \mathcal{E}^l_j\}$, consists of a node set $\mathcal{V}_j = \{\mathcal{V}^l_{j1}, \mathcal{V}^l_{j2}, \ldots, \mathcal{V}^l_{jn}\}$ and an edge set $\mathcal{E}^l_j = \{e^l_{j,1,2}, e^l_{j,1,3}, \ldots, e^l_{j,1,n}\}$. $\mathcal{V}^l_{j1}$ is the centre node. The rest of the nodes in $\mathcal{V}_j$ constitute the peripheral (secondary) nodes. The edge list $\mathcal{E}^l_j$ only contains the edges originating from the centre node towards the peripheral nodes. Two restrictions apply when generating the candidate subgraphs:

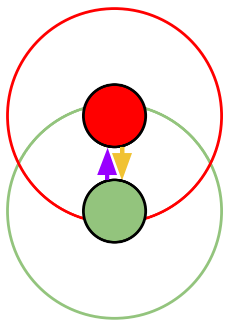
- Each candidate $\mathcal{G}^l_j \in \mathbb{G}^l$ is required to include nodes $\mathcal{V}^l_i$ and edges $\mathcal{E}^l_i$ from only one receptive field graph RG^l_i, $\exists i$. This selective candidate generation procedure enforces $\mathcal{G}^l_j$ to not extend across multiple receptive fields. In Figure 3.7, the top row contains valid examples, while the middle row shows invalid cases where a subgraph

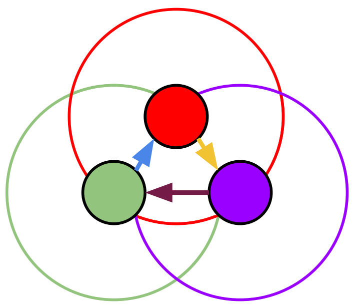

(a) $\mathcal{G}_1^l$ (Valid)

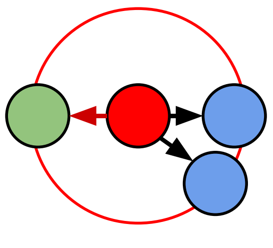

(b) $\mathcal{G}_2^l$ (Valid)

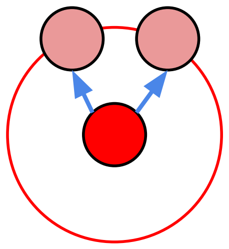

(c) $\mathcal{G}_3^l$ (Valid)


(d) $\mathcal{G}_4^l$ (Invalid)


(e) $\mathcal{G}_5^l$ (Invalid)


(f) $\mathcal{G}_6^l$ (Invalid)


(g) $\mathcal{G}_7^l$ (Invalid)


(h) $\mathcal{G}_8^l$ (Invalid)

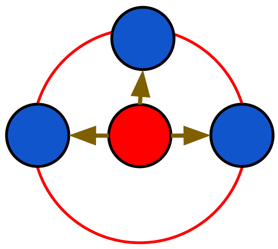

(i) $\mathcal{G}_9^l$ (Invalid)

Figure 3.7: Valid and invalid subgraphs. Each node and edge colour indicates a different label. The circles show receptive fields of the nodes. Top row contains three valid, star-shaped subgraphs, which do not contain any edges between peripheral nodes. Middle row contains subgraphs where edges originate from multiple nodes, hence they are considered invalid. In the bottom row, all subgraphs include identical edge-node pairs, making them invalid examples. (best viewed in colour)

contains edges from multiple receptive fields.

- A candidate subgraph $\mathcal{G}_j^l$ may not contain identical secondary node+edge pairs. All secondary nodes either differ by their node labels, or the connecting edges' labels. Figure 3.7g-i illustrates invalid cases where a subgraph has node+edge repetitions. Figure 3.7c, on the other hand, is a valid example where a candidate subgraph contains duplicate peripheral nodes, but the corresponding edges have different labels. As a result of this rule, each candidate $\mathcal{G}_j^l$ has a *canonical* order of node+edge pairs, which simplifies the subgraph matching problem, as given later in this section.

A brute-force solution for subgraph enumeration would be to consider all possible compositions of the parts in the previous layer $\mathcal{P}^l$. Let us specify $|\mathcal{P}^l| = A_l$, and c_0 as the maximum number of edge types between two parts. It is possible to enumerate A_l subgraphs having a single node, possibly up to $(A_l)^2 \times c_0$ two-node subgraphs, $(A_l)^3 \times (c_0)^2$ three-node subgraphs and so on. This approach quickly becomes infeasible as the learning is shifted to the higher layers in the hierarchy, where a layer can potentially contain tens of thousands of parts. The computational complexity of the trivial solution prohibits discovery of the new layers. While only the subgraphs that are actually encountered in the data are enumerated, this assertion still holds true.

In this thesis, the alternative to the brute-force solution is SUBDUE [33], a frequent subgraph discovery algorithm that enumerates and evaluates candidate subgraphs in $\mathbb{G}^l$ using a *restricted* search space. This restriction ensures that total computation does not exceed a user-defined threshold. The discovery algorithm is explained in Algorithm 3. SUBDUE starts by putting single-node structures into the *parentList* (Step **2**), and extends all parents by one edge/node pair into *childList* (Step **4**). Each graph in *childList* is evaluated using a Minimum Description Length criterion [175] (Step **5**). The generated subgraphs are stored in *partList* (Step **6**). Then, *childList* is pruned using *beam* parameter (Step **7**), and *parentList* is replaced with *childList* (Step **9**). The algorithm repeats until the size of the graphs in *parentList* reach *maxSize*, or *parentList* gets empty. The top *numBest* graphs which have been enumerated are stored, and the algorithm con-

```
   Input  : $\mathbb{G}^l = (\mathbb{V}^l, \mathbb{E}^l)$: Object graph, beam, numBest, maxSize.
   Output: Selected subgraphs $\mathcal{G}^l$.
1  parentList := null; childList := null; partList := null; k = 0;
   where childList,partList are priority queues ordered by MDL scores.
2  Initialise parentList with single node parts;
3  while (parentList is not empty) and (k < maxSize) do
4      Extend parts in parentList in all possible ways into childList;
5      Evaluate parts in childList using Equation (3.4);
6      Merge elements of childList and partList into partList;
7      Trim childList to beam top parts;
8      parentList := null;
9      Swap parentList and childList;
10     k++;
   end
11 Trim partList to numBest top parts;
12 $\mathcal{G}^l$ := partList;
```

Algorithm 3: Enumerating of new compositions.

cludes. The complexity of the algorithm can be controlled by the *beam* and *maxSize* parameters, both of which are given in the experimental section.

The generated subgraphs are collected in the candidate part list $\mathcal{G}^l = \{\mathcal{G}_j^l\}_{j=1}^{\hat{A}_l}$, where $\hat{A}_l$ is the number of enumerations. The candidate graphs $\mathcal{G}^l$ are matched to their instances in the lattice $\mathbb{G}^l$ in the next step, using subgraph isomorphism (SI). This problem is solved as follows.

Finding Part Realisations using Subgraph Isomorphism

The candidate part (graph) list $\mathcal{G}^l$ is obtained in the previous step. Given a candidate graph $\mathcal{G}_j^l$, SUBDUE searches for all matching subgraphs in the lattice $\mathbb{G}^l$ for the instances of $\mathcal{G}_j^l$. The results are collected in $iso(\mathcal{G}_j^l)$, the set of subgraphs in $\mathbb{G}_l$ that are isomorphic to $\mathcal{G}_j^l$. Finding the isomorphisms of a graph $\mathcal{G}_j^l$ in a larger graph $\mathbb{G}^l$ is known as the Subgraph Isomorphism (SI) problem, which is known to be NP-complete. In order to solve it, we use simple brute-force approach as explained by Ullmann in [220]. Matching a candidate graph to the data is analogous to finding the instances of a query shape in the corresponding images.

The formulation of the SI problem requires a function *match*() that performs the exact matching of two sub-graphs based on their edge and node labels. It returns *true* if two input graphs are a perfect match (exact isomorphism), and $false$ otherwise. Two graphs are isomorphic if their node and edge labels match, as well as the topology of the graphs. More formally, graph isomorphism is an edge-preserving vertex bijection (one-to-one correspondence) which preserves equivalence classes of labels, i.e. vertices with equivalent labels are mapped to vertices with equivalent labels and vice versa, while the same holds true for edge labels [26]. Two vertices/nodes are considered to be equivalent if their node labels (part indices) match. Two edges are equivalent if the edge labels match, as well as the nodes on both sides of the edges. One may think that exact isomorphism would make the shape matching process *brittle*, i.e. small shape differences would lead to node/edge differences and break the matching procedure. Because of the learned spatial statistics encoded in edges (Section 3.3.2) and part grouping (Section 3.3.5), it is possible to learn hierarchical deformable models of parts which can represent many instances with slight shape variations.

Given a pattern (candidate graph) $\mathcal{G}_i^l$, the system first discovers all matching subgraphs $G_i^l = \{G_{ij}^l\}_{j=1}^{\hat{B}_i^l}$ in the lattice $\mathbb{G}^l$, where $\forall i, match(\mathcal{G}_i^l, G_{ij}^l) = true$, and $\hat{B}_i^l$ is the number of matching subgraphs. Then, each subgraph G_{ij}^l is replaced by a single realisation R_{ij}^{l+1} in the higher layer in order to obtain $\mathbb{G}^l|\mathcal{G}_i^l$, the compressed version of $\mathbb{G}^l$ with $\mathcal{G}_i^l$. For each matching subgraph $G_{ij}^l = \{V_{ij}^l, E_{ij}^l\}$ in $\mathbb{G}^l$, the nodes V_{ij}^l and internal edges E_{ij}^l are deleted, and replaced with a single realisation in the higher layer R_{ij}^{l+1}. Additionally, the external edges that are connected to each one of the nodes in V_{ij}^l are removed, and re-attached to R_{ij}^{l+1}. Using this procedure, all possible parts individually compress $\mathbb{G}^l$, and they are evaluated by the quality of the compression. Finally, a representative set of parts that *collectively* compress and cover $\mathbb{G}^l$ are selected, the higher level's realisations R^{l+1} are retrieved from their instances. The candidate parts are evaluated as follows.

Evaluation of Compositions of Parts Using MDL

The Minimum Description Length principle [175] states that the best hypothesis/model for a given set of data is the one which best compresses the data. It is a formalisation of Occam's Razor, which Ptolemy put as "We consider it a good principle to explain the phenomena by the simplest hypothesis possible" [123] [1]. The graph search tool used, SUBDUE [33], applied this idea to graphs. SUBDUE evaluates subgraphs based on their ability to compress the input graph $\mathbb{G}^l$. This work follows the same method when evaluating and ranking parts during learning. In SUBDUE, the label of a part $\mathcal{P}^l_j$ is defined based on its compression value $value(\mathcal{G}^l_j, \mathbb{G}^l)$ computed in Equation (3.4). The parts are sorted according to their compression values in ascending order, and assigned their respective indices as part labels.

$$value(\mathcal{G}^l_j, \mathbb{G}_l) = \frac{DL(\mathcal{G}^l_j) + DL(\mathbb{G}^l|\mathcal{G}^l_j)}{DL(\mathbb{G}^l)}. \tag{3.4}$$

where $value(\mathcal{G}^l_j, \mathbb{G}_l)$ is the compression value of the lattice $\mathbb{G}^l$ given a subgraph $\mathcal{G}^l_j$. $\mathbb{G}^l|\mathcal{G}^l_j$ is the compressed version of the input graph $\mathbb{G}^l$, given $\mathcal{G}^l_j$. Description length $DL(G)$ of a graph G is calculated using the number of bits to represent the node labels, edge labels and adjacency information of G [33].

The compositions of the parts in $\mathcal{P}^l$ which *best describe* the training set are discovered, using Equation (3.4). Each composition is evaluated based on how well it compresses the input data $\mathbb{G}^l$. The compression ratio is calculated by the reduction in the description length $DL(.)$ of input graph $\mathbb{G}^l$ after the compression. During the inference, we search for a set of graphs $\mathcal{G}^l = \{\mathcal{G}^l_j\}_{j=1}^{\hat{A}_{l+1}}$ which minimise the description length of $\mathbb{G}_l$ as

$$\mathcal{G}^l = \arg\min_{\mathcal{G}^l_j} value(\mathcal{G}^l_j, \mathbb{G}^l). \tag{3.5}$$

In order to limit our search space, the candidate subgraphs $\mathcal{G}^l_j$ are ranked by their

[1]While the principle is attributed to William of Ockham (*circa* 1287-1347), its origins are traceable to Ptolemy (c. AD 90 - c. AD 168), and even Aristotle (BC 384 - 322). I think Ptolemy puts it best, hence I chose to include a quote from him.

compression scores in ascending order (lower is better), and select the top $\hat{A}_{l+1}$ graphs. $\hat{A}_{l+1}$ is the number of generated subgraphs for the next layer (given in the experimental section). Each is then assigned a label which is equal to its ranking index. The next section describes the final part selection procedure, which is used to create a compact vocabulary of parts.

3.3.4 Part Selection

The number of possible parts grows exponentially as the algorithm progresses through higher layers. Three factors are responsible for this increase. First, each new part is a composition of the parts from the previous layer, which introduces combinatorial growth. Second, there can be a number of different edge types between pairs of nodes. Finally, a relatively high (10) limit on the number of allowed sub-parts is set, which means duplets, triplets and larger compositions of sub-parts can be candidate parts for the next layer. The number of enumerated parts can easily surpass hundreds of thousands in the higher layers, making learning infeasible.

In order to reduce the number of parts that represent a set of shapes, an MDL-based selection mechanism is employed. The part selection procedure aims to select the minimal set of parts that *collectively* compress the entire training set. It runs in a greedy fashion to pick the next best part to compress the uncovered portion of the dataset, until the whole data is covered to a high degree. This collective selection process runs over a pool of parts $\mathcal{G}^l$ which *individually* compress the data well. Next, an MDL-based part selection algorithm is executed over the set of generated parts $\mathcal{G}^l$, as explained in Algorithm 4.

Support of a part, *support*$(\mathcal{G}_j^{l+1})$, is the set of layer 1 realisations that contribute to $\mathcal{G}_j^l$'s instances. Similarly, support of a part realisation *support*(R_i^l) is the set of layer 1 realisations which activated R_i^l. In step **4**, the current data graph $\mathbb{G}'$ is pruned so that the part realisations with covered supports are removed. This procedure iteratively compresses $\mathbb{G}^l$ using the next best candidate. The realisations in $\mathbb{G}^l$ whose supports are fully covered by the existing part set are deleted at every iteration.

```
Input  : Data graph $\mathbb{G}^l$.
Output: Part list $\mathcal{P}^{l+1}$ for layer $l+1$.
1 The current data graph $\mathbb{G}'$ is initialised with the data graph $\mathbb{G}^l$;
2 Among all generated subgraphs $\mathcal{G}^l$, the one $\mathcal{G}^l_j$ with the best MDL score
    $value(\mathcal{G}^l_j, \mathbb{G}')$ is added to the next layer's part list $\mathcal{P}^{l+1}$ (Equation (3.4));
3 $support(\mathcal{G}^l_j)$ is regarded to be covered, and does not contribute to the MDL
    calculations in the following steps.
4 $\mathbb{G}'$ is pruned so that every realisation in $R^l$ which has a fully covered support is
    deleted;
5 2-5 is performed until all training data is covered.
```
Algorithm 4: Selecting parts.

At the end of the part selection process, the selected group of parts $\mathcal{P}^{l+1}$ and their realisations R^{l+1} constitute the next layer. A potential pitfall here is that $\mathcal{P}^{l+1}$ may be sub-optimal according to the final task, due to the greedy selection algorithm. This situation arises since CHOP is an entirely feed-forward architecture, and each layer can only be constructed once the previous layer is finalised. The part selection criterion (MDL) favours frequent parts that compress the data well, which may result in the loss of infrequent, complex structures, and these structures may prove to be useful in higher layers. To the best of our knowledge, such problems can only be solved by adding a feedback loop to the vocabulary building process to optimise the hierarchical vocabulary jointly. The design of such a system is beyond the scope of the thesis.

Before the algorithm proceeds with learning of the next layer, a part grouping operation is performed. The reason for this operation is that similar looking parts (possibly with slight variations) are included in $\mathcal{P}^{l+1}$ with different labels. This results in an overfitting of parts to the training set, and many structures that are activated by few instances in higher layers. In return, the generalisation ability of the learned vocabulary is reduced. This issue is remedied by grouping parts based on appearance similarity and assigning labels to each cluster, thus marking them as "variations of the same shape". This process is entirely unsupervised, and is explained in Section 3.3.5.

3.3.5 Grouping Parts into OR Nodes

After the set of parts for the next layer $\mathcal{P}^{l+1}$ and their realisations R^{l+1} are obtained, the next step is to perform a part grouping operation that uses unsupervised agglomerative clustering of parts based on their appearance features. Part grouping is necessary because the part representations are not canonical, i.e. the same shape can be decomposed into multiple parse trees (each part has a parse tree). Figure 3.8 illustrates a simple case where two third-level parts (X and Y), while having different parse trees, represent the exact same structure. By comparing their appearance in the image space (shown at layer 1), it is possible to group X and Y together, so that they have the same label. The part grouping process can be explained as follows:

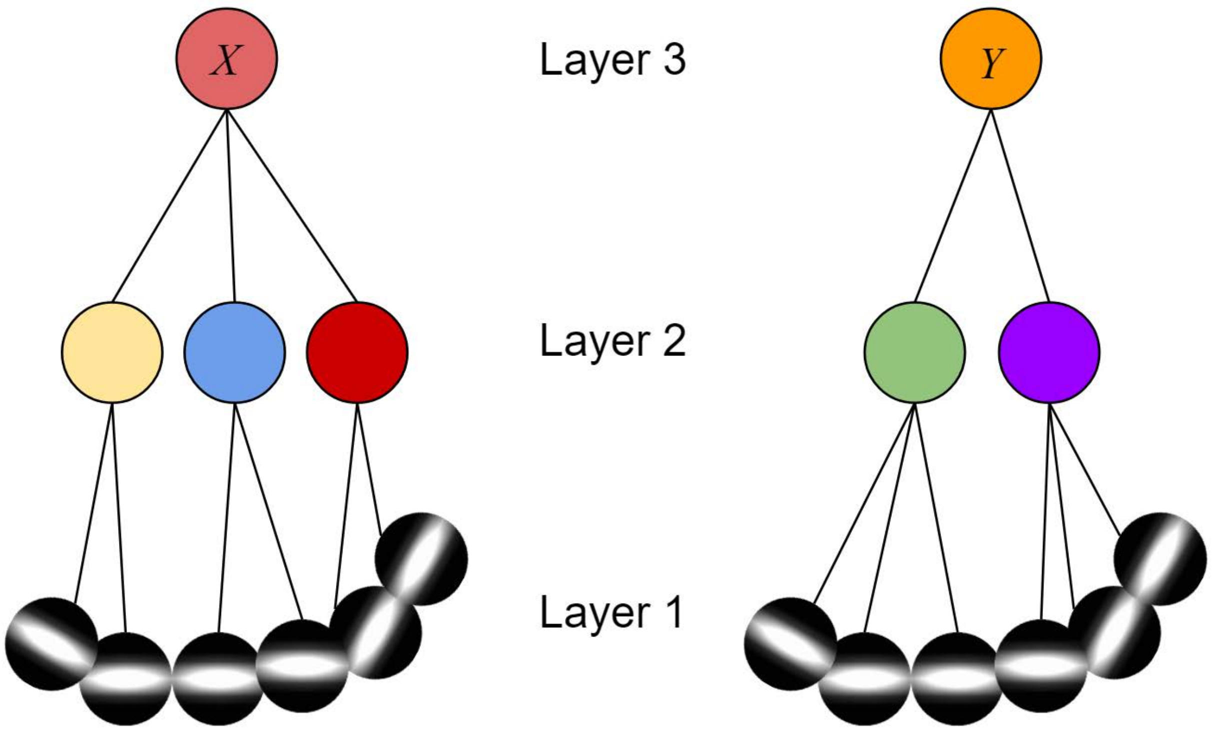


Figure 3.8: X and Y are two layer-3 parts which have the same appearance, but different parse trees. This case shows that part representations are not canonical, and there are multiple ways to describe the same shape. Part grouping groups X and Y together, and makes them *alternatives* for one another.

1. For each part $\mathcal{P}_j^{l+1}$ in $\mathcal{P}^{l+1}$, a part projection $image(\mathcal{P}_j^{l+1})$ is created by imagining a projection of $\mathcal{P}_j^{l+1}$ in terms of layer 1 nodes. This works by projecting the part

back to the image plane by its *mean* spatial parameters. The resulting image is a *mean shape* for the part.

2. A shape descriptor is obtained from each $image(\mathcal{P}_j^{l+1})$.The Shape Context (SC) [12] is used in this thesis. The SC descriptor, as shown in Figure 3.9, is a circular histogram of uniformly sampled points on the shape. The origin of the histogram is placed at the centre of gravity of the shape. The shape descriptor (on the right) is obtained by counting the number of points in each bin and concatenating the results.

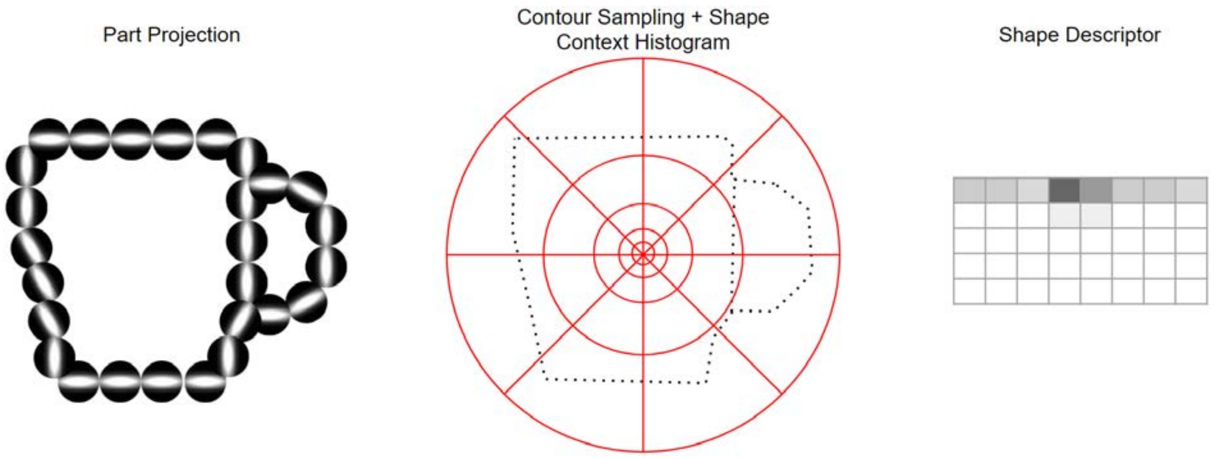


Figure 3.9: The histogram for shape context descriptor. The input part's projection is on the left. The points are uniformly sampled along the contour, and the SC histogram is placed at the mean of the sampled points (middle). The 8×5 shape descriptor is on the right.

3. The parts are grouped using agglomerative clustering based on their shape descriptors, using Euclidean distance.

4. If two parts $\mathcal{P}_j^{l+1} = (\mathcal{G}_j^{l+1}, \mathcal{Y}_j^{l+1})$ and $\mathcal{P}_k^{l+1} = (\mathcal{G}_k^{l+1}, \mathcal{Y}_k^{l+1})$ fall into the same cluster, they are assigned the same label $\mathcal{Y}_{cl}^{l+1}$, with *cl* being the cluster id. In other words, $\mathcal{Y}_j^{l+1} = \mathcal{Y}_k^{l+1} = \mathcal{Y}_{cl}^{l+1}$.

The part grouping procedure ensures that variations of the same shape are assigned identical labels. The aim of including OR nodes is to combat over-fitting to small shape variations in the data. This operation effectively groups parts which are similar in appearance features, regardless of their hierarchical decomposition. The number of final OR node groups θ^l depends on the level l, and is given in the experimental section. The cluster labels are encoded in the part labels $\mathcal{Y}_i^{l+1}$ for every part $\mathcal{P}_i^{l+1} = (\mathcal{G}_i^{l+1}, \mathcal{Y}_i^{l+1})$.

The next level's vocabulary and realisations are built as follows.

3.3.6 Construction of the Next Layer and Its Realisations

After the set of frequent subgraphs and their part indices are obtained for the $(l+1)^{th}$ layer, first, a set of parts $\mathcal{P}^{l+1} = \{\mathcal{P}_i^{l+1}\}_{i=1}^{A_{l+1}}$ are constructed, where $\mathcal{P}_i^{l+1} = (\mathcal{G}_i^{l+1}, \mathcal{Y}_i^{l+1})$ is the i^{th} part of $(l+1)^{th}$ layer. The label $\mathcal{Y}_i^{l+1}$ is the cluster label coming from the OR node grouping in Section 3.3.5. $\mathcal{P}^{l+1}$ is the set of *compositions* of the parts from the previous layer $\mathcal{P}^l$.

For each subgraph G_{ij}^l in $iso(\mathcal{P}_i^{l+1})$, a new realisation is placed in higher level's realisation set $\hat{R}^{l+1} = \{R_{ij}^{l+1}\}$, s.t. $R_{ij}^{l+1} = (G_{ij}^l, Y_i^{l+1})$, where $Y_i^{l+1} = i$ is the relevant part's identifier. While not listed in this notation, each realisation R_{ij}^{l+1} is associated with an image n_{ij}^{l+1} and an image position m_{ij}^{l+1}. The image identifier of the new realisation n_{ij}^{l+1} is the same as any node in its graph description G_{ij}^l (since all nodes come from the same image). The position of the new realisation m_{ij}^{l+1} is the centre of gravity (mean position) of all of the nodes in its graph description G_{ij}^l. In order to remove the redundancy in $\hat{R}^{l+1}$, the resolution of image positions m_{ij}^{l+1} is reduced, and max pooling is performed among each OR node group's realisations. During this operation, each realisation has the same value (activation), hence the max pooling operation *randomly* selects one realisation to keep among overlapping samples . The result is a new set of realisations $R^{l+1} \subseteq \hat{R}^{l+1}$.

The next two sections contain information about the experiments from both of our papers [2, 158], as well as the training parameters used to train the methods.

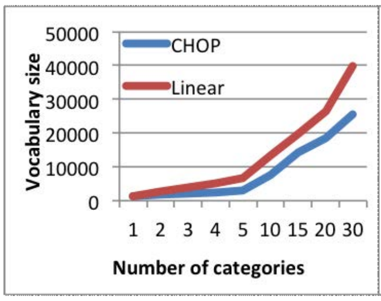
3.4 Experiments

This section gives an experimental analysis of CHOP on six benchmark datasets. The tested hypothesis is that hierarchical, deep architectures will improve upon the performance of current methods in shape retrieval and classification tasks. In the shape retrieval task, CHOP was compared with the state-of-the-art algorithms on 3 datasets. Section 3.5 gives an experimental overview of the shape classification task, where CHOP is evaluated on 5 benchmark datasets. We performed multiple trials of experiments where possible, aggregating their results by calculating and listing their means and standard deviations. The reported results are not tested for statistical significance.

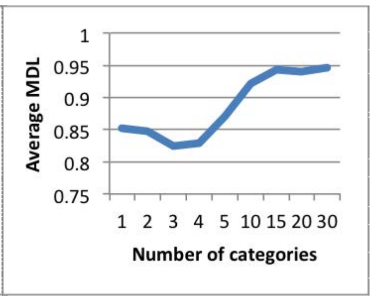
The proposed method was tested on the Washington Image [126], MPEG-7 Core Experiment CE-Shape 1 [113], ETHZ Shape Classes [56], 40-sample articulated tools Tools-40 [75], 35-sample multi-class Tools-35 [24], and Mythological Creatures [24] datasets. The experimental setup used $\theta = 6$ filter pairs, and set the downsampling ratio σ to 0.5. The number of maximum modes in pair-wise learning is set to $c_0 = 10$. The radius rad of receptive fields was set to 7. The part discovery process used $beam = 100$ and $maxSize = 10$ parameters. No OR-node grouping was used in the experiments reported in this section, except for Section 3.4.4. The Matlab version of CHOP can be accessed here.

3.4.1 Analysis of Generative and Descriptive Elements

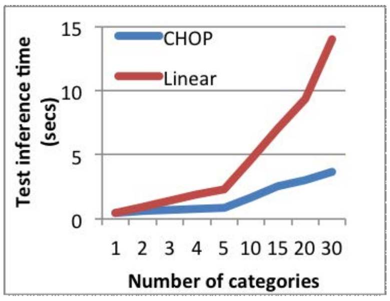
This section analyses different aspects of the learned vocabularies, by looking at the number of classes, camera viewpoints, objects, size of the vocabulary, MDL values and test inference times. The relationship between the vocabulary size, growth rate and added models/classes are examined to understand the part shareability behaviour of the proposed method. The inference time of CHOP is calculated by averaging the running times of the inference algorithm which is employed on test images. Below are explained the different experiments conducted for shape retrieval on the benchmark datasets.



(a)



(b)



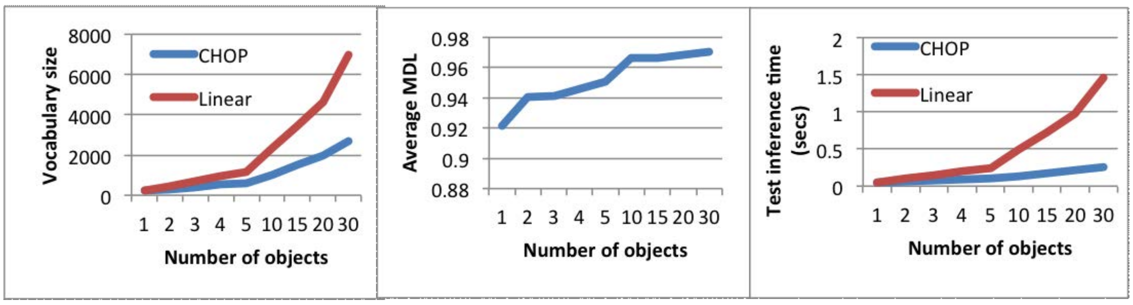
(c)

Figure 3.10: Analyses with different number of classes. (Best viewed in colour)

Effect of Varying Number of Classes In this set of experiments, 30 classes from MPEG-7 Core Experiment CE Shape 1 dataset [113] were used. 5 shapes from each class to were used train the shape hierarchy.

The shareability of the parts in the vocabulary is directly connected with the growth rate of the vocabulary with respect to the number of classes. Figure 3.10.a illustrates that the number of parts in the vocabulary grows sub-linearly, as the number of classes increase. CHOP has higher part-shareability in lower layers, due to the smaller receptive fields in image space. The test image inference time, as shown in Figure 3.10.c, also increases in proportion to the vocabulary size. This effect is observed because the inference of a list of parts is a linear process, i.e. every part's subgraph needs to be matched with the data. The average time of full inference on a test image is between 0.5 and 3 seconds, and is dependent on the number of classes. The average MDL values of top parts (usually 10) is another analysed aspect, as seen in Figure 3.10.b. The MDL values improve at 3-4 classes, and then tend to increase towards more complex datasets (lower is better). The inter-class shape differences effectively limit the part shareability between classes.

In Figures 3.10, 3.11 and 3.12, CHOP is compared with estimated linear growth baselines. The linear baseline plot in each figure is calculated by obtaining the 1-unit data point on the left-most side of a plot experimentally, and linearly scaling (multiplying) it to up to 30 units.



(a) (b) (c)

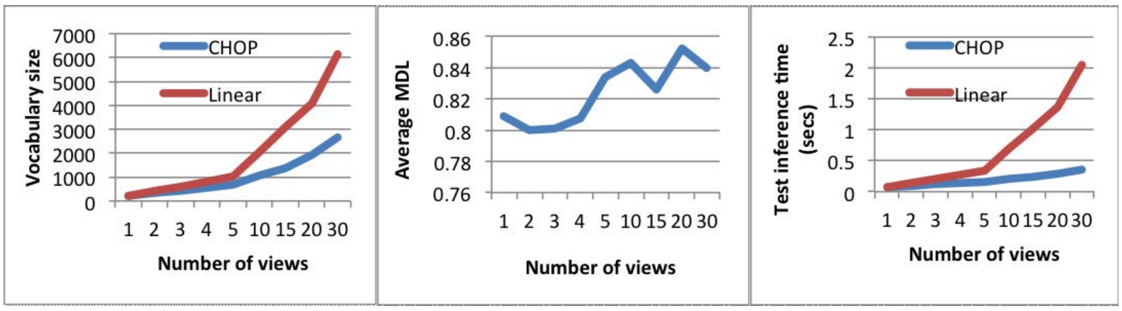
Figure 3.11: Analyses with different number of objects. (Best viewed in colour)

3.4.2 Analyses with Different Number of Objects

The effect of increasing the number of objects from the same class on the vocabulary is examined in this section. Up to 30 objects from the "Apple Logos" class in ETHZ Shape Classes dataset [56] were used to train CHOP. In Figure 3.11.a, it can be noted that vocabulary size has a sub-linear growth, and the vocabulary exhibits more shareability than in multi-class training (compared to Figure 3.10.a). Figure 3.11.b shows a gradual increase in MDL values as expected, because the variability among the shapes increase. The part shareability among the samples of the same class are better than samples from different classes.

3.4.3 Analyses with Different Number of Views

The experiments in this section examine the case where multiple views of the same object are used for learning. The system was trained on different views of a textured mug from Washington Image dataset [126]. The roughly symmetrical shape of a mug, except for its textures and the handle, allows for fairly high shareability of the parts in the vocabulary, as evidenced by Figure 3.12.a and 3.12.c. One thing to note is that due to its symmetrical properties, the parts representing a mug's body may be ranked higher than distinctive parts such as the mug handle. This is due to a mug handle having varying appearances from different viewpoints, with each view being infrequent by itself. This may be seen as a drawback for MDL-based training, and this aspect is addressed in the conclusion



(a) (b) (c)

Figure 3.12: Analyses with different number of views. (Best viewed in colour)

chapter.

3.4.4 OR Node Grouping Analysis

The shape retrieval and classification experiments in the rest of this chapter are obtained using an earlier version of CHOP which does not contain the OR node grouping procedure presented in Section 3.3.5. The idea of grouping parts based on their appearance similarities has been explored in the Learned Hierarchy of Parts (LHOP) architecture [58]. In order to understand the effect of OR-node based grouping in CHOP, a synthetic 3D object dataset consisting of 20 classes, containing 20 objects each, was prepared for shape classification. 15 objects from each class were used for training, while 5 objects from each class were allocated for testing. The training dataset consists of 20000 images. Each test image was classified to 1 of 20 classes, using the class of the top level node inferred using CHOP. Using part grouping improves the performance from 24.9% to 29.5% on a top-1 classification task, and from 60.7% to 72.1% on a top-5 classification task. The improvement resulting from part grouping can be attributed to the observation that a more compact and smaller vocabulary is learned, increasing the generalisation performance of the algorithm.

3.4.5 Shape Retrieval Experiments

Similarly with [168, 199, 201], the shape descriptors consist of the eigenvalues of the adjacency matrices that belong to the image graphs of shapes. The edge weights $e^l_{ab} \in E^l_j$ of an edge weighted graph $W^l_j = (V^l_j, E^l_j)$, which is a copy of the image graph $\mathbb{G}^l_j = (\mathbb{V}^l_j, \mathbb{E}^l_j)$, are defined as

$$e^l_{ab} = \begin{cases} \pi_k, & \text{if} \quad R^l_a \text{ is connected to } R^l_b, \quad \forall R^l_a, R^l_b \in \mathbb{V}^l_j \\ 0, & otherwise \end{cases}, \tag{3.6}$$

where π_k is the cluster index of the edge, coming from the solution of Equation (3.2). Second, the weighted adjacency matrix of W^l_j is computed, and its eigenvalues are used as shape descriptors for every image. The difference between two shapes is defined as the Euclidean distance between their shape descriptors, and use this method for shape retrieval.

The proposed shape retrieval method was compared with the state-of-the-art shape retrieval algorithms using inner-distance (ID) measures as shape descriptors which are robust to articulation [75]. All methods were tested on the Tools-40 dataset [75], which contains silhouettes of 8 different objects having 5 different articulations, thus consisting of 40 images. The task on which all algorithms run is to retrieve closest 1-4 articulations of a query object among the remaining 39 images. Table 3.1 contains a summary of the top 1,2,3, and 4 retrieval results. CHOP outperforms SC+DP [75] and MDS+SC+DP methods [75] in all retrieval experiments. IDSC+DP [75], which combines texture information with a shape descriptor, has a better Top 1 retrieval performance, while CHOP performs better than IDSC+DP in Top 4 retrieval results. This observation is related to texture of an object having further discriminative information about a shape. The downside is that texture information may dominate shape information, and can lead-to overfitting, as evidenced by IDSC+DP's Top 4 retrieval results in Table 3.1.

In a different set of experiments, the shape retrieval performances of CHOP and two

Algorithms	Top 1	Top 2	Top 3	Top 4
SC+DP [75]	20/40	10/40	11/40	5/40
MDS+SC+DP [75]	36/40	26/40	17/40	15/40
IDSC+DP [75]	**40/40**	34/40	**35/40**	27/40
CHOP	37/40	**35/40**	**35/40**	**29/40**

Table 3.1: Comparison of shape retrieval performances (%) on Tools-40 dataset.

other shape retrieval methods in [150] were compared in Mythological Creatures (3 classes, 15 shapes) and Tools-35 (4 classes, 35 shapes) datasets. The Mythological Creatures dataset consists of 3 classes (Human, Horse, Centaur) and includes 5 articulations of a shape in each class. Human and Centaur classes share the higher part of the body, while Centaur and Horse share the lower part. The Tools-35 dataset consists of 4 classes of objects, namely, 15 pliers, 10 scissors, 5 knives and 5 pincers. Similarly with Mythological Creatures, Tools-35 includes a different articulation for each object in a class. Following the suggestion of [150], a so-called Bullseye test was performed to compare CHOP, Contour-ID [150] and Contour-HF [150]. In the Bullseye test, each query image is used to retrieve the 5 most similar candidates from the rest of the objects, and the performance is calculated as the ratio of correct hits among all retrieved objects, across all query images. Table 3.2 shows that CHOP performs better than Contour-ID and Contour-HF [150], which use contour-based shape descriptors and are invariant to local articulations and deformations of a shape.

Datasets	Contour-ID [150]	Contour-HF [150]	CHOP
Tools-35	84.57	84.57	**87.86**
Myth	77.33	90.67	**93.33**

Table 3.2: Comparison of shape retrieval performances (%) on Myth and Tools-35.

3.5 Additional Experiments and Analysis

This section summarises the shape classification experiments in [158]. First, the invariance and robustness properties of the shape vocabularies learned by the proposed method on a classification task are examined. The used dataset contains added noise and geometric transformations for this experiment. Second, shape representations are learned using CHOP for shape classification on two benchmark datasets. The proposed methods in [158] have two key features added to the CHOP codebase. [1] First, a loss function was implemented in the inference procedure in order to perform shape classification. Second, four distance functions were computed to embed object parts in shape subspaces having different invariance features. The details of these changes are outside the scope of this thesis. The reader is advised to refer to the paper [158] for a full discussion.

3.5.1 Robustness and Invariance Analysis on Artificial Datasets

The suggested approach is analysed and compared to the state-of-the-art on three artificial benchmark datasets. In these datasets, shapes consist of single and/or multiple components, with added noise and geometric transformations to make them more challenging [112]. The size of images in each of them is 28×28. The three datasets are:

- Convex Shapes (**CS**) [112] dataset contains 8000 training and 50000 test images, belonging to two classes (convex, non-convex). The task defined is a binary classification of the shape regions into convex or non-convex classes.
- MNIST-Rotation (**M-R**) [112] dataset consists of 12000 training and 50000 test images of hand-written digits, belonging to 10 classes (0-9). It is an enriched version of MNIST [83, 185], where digit samples are rotated uniformly between 0 and 2π radians and added to the set.
- MNIST-RBI (**M-RBI**) [112] is, similarly with **M-R**, a modified version of MNIST. Each image contains a rotated version of an MNIST digit on a background which is randomly sampled from a grayscale image.

[1]These two extra features were implemented by Dr. Mete Ozay. The author of this thesis designed and implemented the core CHOP algorithm and assisted with the experiments in the corresponding paper.

Algorithm	**CS**	**M-R**	**M-RBI**
DBN-3 [112]	81.37	87.70	71.49
DAE2 [27]	81.56	88.06	55.08
CAE2 [27]	80.70	86.38	51.75
mDAE2 [27]	81.90	89.64	53.88
CHOP	**84.15**	**90.16**	**77.15**

Table 3.3: Classification performance of the algorithms on the benchmark shape datasets.

The proposed algorithm is compared with three-layer deep belief networks (**DBN-3**), denoising (**DAE-2**), contractive (**CAE-2**) and marginalised denoising auto-encoders (**mDAE-2**) with two layers.

Table 3.3 illustrates that the proposed method, CHOP, outperforms the state-of-the-art methods in 2015. CHOP performs significantly better in classifying the noisy, corrupted images in the **M-RBI** dataset. This is due the employed graph compression algorithms removing infrequent, spurious patterns from background images. The MDL criterion, coupled with the limited resources enforced by setting a maximum number of parts in every layer, helps reduce the noise in the data representation. While the margin is small, CHOP outperforms the other methods in **CS** and **M-R** datasets as well. In this set of experiments, the intention is to create a scale, rotation and translation invariant representation, details of which will not be included here, and can be accessed in [158].

3.5.2 Comparison with State-of-the-art Shape Classification Methods Employed on Shape Manifolds

S-kFP [93]	**S-kFPG** [94]	**MKL** [93]	**SM** [93]	**CHOP**
57.75 ± 2.0	60.37 ± 1.6	60.84 ± 2.0	63.98 ± 1.6	$\mathbf{66.17 \pm 2.3}$

Table 3.4: Classification performance of the algorithms on the Butterfly Dataset.

This section contains a comparison of the performance of CHOP and state-of-the-art classification methods using shape manifolds by performing shape classification on the Butterfly Dataset [230]. This dataset consists of 823 images and 10 classes. 40 shapes

CSS [209]	**IDSC** [131]	**CSSP** [6]
69.7	73.6	78.4
NN-SVM [7]	**Lim** [128]	**CHOP**
84.30 ± 1.0	80.4	**85.9 ± 2.4**

Table 3.5: Classification performance of algorithms on the Animals Dataset.

from each class were randomly picked for training the method, and rest of the dataset for testing, following the procedure introduced in [93]. This test was repeated 10 times, and created the mean/variance of the classification results in Table 3.4.

When dealing with real-world datasets which may contain object articulations and non-rigid body transformations, the amount of variability in geometric patterns across viewpoints, classes and objects increases. Algorithms should model this variability to be able to represent the data. As a result, the methods which employ shape spaces with more diverse geometric properties were observed to outperform others in these datasets. Of the algorithms presented in Table 3.4, **S-kFP** and **MKL** use full Procrustes distance on Kendall's shape space [93]. Table 3.4 shows that CHOP's shape representation is able to deal with the variability in Butterfly dataset well, as it outperforms the other methods.

3.5.3 Comparison with State-of-the-art Shape Descriptors and Vocabularies

The final experiments compare the algorithms on the shape classification task on the Animals Dataset [6], which contains 20 classes and 2000 images. Compared to MPEG-7 [113], this dataset contains significantly more variation. The shapes contain articulations and non-rigid body transformations, as well as considerable within-class variation among the objects of a class [7]. 50 shapes per class were randomly picked, and the rest of the images were used for testing, as advised by [7]. The procedure was repeated 10 times, and the results can be found in Table 3.5.

Table 3.5 shows that CHOP outperforms the state-of-the-art algorithms. One important aspect is that our method's performance was measured over 10 trials, unlike 100 trials reported for **NN-SVM** [7]. The lower variance of **NN-SVM** can be partly attributed to the reduced variability as a result of higher number of trials.

Part II

Deep Dexterous Grasping

CHAPTER 4

RELATED WORK: GRASPING

Grasping ability is essential for humans in that it enables us to interact with the objects around, including drinking, cooking, throwing, holding and pulling actions, as well as many others. For robotics research, grasping has been a challenging problem and widely studied since the beginning of the field. An important aspect of grasping is generalisation: How can a robot grasp new, unseen objects which did not contribute to the training/design of a system? A second question, which is also relevant to this thesis, is how can a system generalise over new object poses? Below, an overview of the grasping methods in the literature is given, while attempting to find answers for both questions.

The method presented in this chapter attempts to solve the problem of dexterous grasping of previously unseen objects using a single-pose depth camera. In this context, ”previously unseen” objects refer to object instances that the system was not trained with, although the system may have seen similar objects before. Only non-deformable household objects which are graspable using a single human hand are considered for the experiments. The tested objects contain plastic, wood, metal and porcelain parts, and some have textures. The objects are placed in the environment in arbitrary poses, and viewed from a single, arbitrary camera viewpoint. The proposed approach combines a grasp generation method with an evaluative learner, both of which fall under the umbrella of learning methodologies in Section 4.3.

4.1 Overview

We consider the existing literature on grasping under two broad categories: Geometrically-driven grasping, and learning-based approaches. Geometrically-driven methods analyse the object shape and propose grasps based on the end-effector kinematics (hand or gripper) [16, 232, 177]. The shape analysis can be done by matching of a known object model to the scene, or simply by interpreting a depth/colour image. Learning-based methods come in a variety of different forms. Human supervision can be used to predict grasp configurations [80, 119, 206]. It is also possible to learn an end-to-end grasp policy, as shown by Levine et al. [122, 121] and Bousmalis et al. [21]. The path followed in this thesis is learning to predict the success probabilities of grasps on unknown objects. A Convolutional Neural Network was used for grasp success prediction, evaluating the grasps from the generative model of Kopicki et al [105]. In the rest of this chapter, a review of the grasping literature is given to place the work in the right context.

4.2 Geometric Methods

Analytic, model-based methods use the laws of physics, mechanics and geometry to design grasping algorithms [16, 136, 163, 149, 134]. The physical elements of grasping can be incorporated in a system in the form of grasp metrics, such as force closure [15, 134] or caging [177]. In most cases, due to the complexity of the physical interactions between the objects, scene and the end-effector, these methods make restrictive assumptions. Data-driven approaches overcome these limitations by relying less on our understanding about the physical forces, and designing high-capacity learning models which learn the physical properties indirectly from the data.

Bicchi and Kumar reviewed analytic approaches in [16] and identified a lack of methods that can deal with positional errors. The visual perception of the object pose and shape is prone to errors due to the noise in the data acquisition process. Nguyen [153] explored *independent contact regions* (ICRs) [164, 183] for modeling force-closure grasps which are robust to the variance in the finger locations. Alternatively, caging can be used for performing grasps which are robust to the positional errors [177]. Rodriguez et al. [177] introduced the concept of *pregrasping cages*, caging configurations from which the object can be grasped by maintaining the enclosure property, regardless of the exact finger positions. They showed that while it is guaranteed to keep the object trapped during grasping using two-finger grippers, the same does not hold true for more than two fingers. Seo et al. [193] used caging to produce immobilising pinch grasps using a gripper. In the case of dexterous hands having more than 2 fingers, it is difficult to rely on caging for grasp planning.

The algorithms presented in this section utilise prior knowledge about the objects in question, including mass, shape, friction coefficients, mass distribution and elasticity. In addition to these, the full kinematics and dynamics of the robot, along with the characteristics of all interacting objects in the scene (e.g. a table) are known. With the knowledge of the physical properties of all entities in a scene, it is possible to compute the contact forces on the finger links during a grasp. Grasp quality metrics can be defined based on this collected information [55, 176, 197]. ϵ-metric [55] is widely-used for measuring the *quality* of a force-closure grasp by the size of the sphere fully contained in the grasp wrench space [42]. Weisz and Allen [232] used the ϵ-metric to find stable grasps under object pose uncertainty. [232] relies on GraspIt! [149] for generating plausible force-closure grasps. A common criticism of force-closure grasps and the ϵ-metric is that they are do not correlate with the actual success or *stability* of a grasp [8], resulting in *fragile* grasps [42]. Despite the efforts, grasp quality metrics have proven to be insufficient in predicting actual grasp success [11, 100, 68].

Even though analytic methods have been successfully applied in multi-finger grasp generation [22, 70, 76, 179, 187], there are problems with this approach. The precise geometric and physical properties of the objects are not always available to the robot in the real world. The surface friction coefficients, mass distribution, weight and centre of mass of an object are among the parameters which may not be known in advance with high accuracy. Estimating them is a challenge: Zhang and Trinkle [241] used a particle filter to track and estimate some of the physical properties of an object, while it is being pushed by the robot. The mass [243, 195] or surface friction coefficients [178] can be estimated by interacting with the object and observing its motion. Similar methods operate under restrictions, as some rely on a simulation environment [177, 179] or only consider 2D objects [193, 241]. The estimation of the objects' parameters is prone to errors, limiting the application of analytic methods in uncontrolled environments. The errors introduced at the estimation stage can be enough to cause promising grasps to fail, as explained below.

In the case of grasping previously unknown objects, analytic methods rely on the estimation of the object shape using a variety of means. Dune et al. [45] approximated the object shape with a quadric, whose minor axis guided the wrist pose. In [45], the centre of mass of an object is approximately the approach target, and the shape of the object drives the hand pre-shape. Similarly, Lippiello et al. [133] used a quadric approximation of the object shape as the initialisation, and employed an interleaved *grasp configuration and object shape optimisation* algorithm to refine the grasps. It is possible to reconstruct the missing parts of an object model based on the shape symmetry assumption [18, 147], or a 3D Convolutional Neural Network [224]. The errors in the shape completion process, along with further errors in the estimation of parameters such as friction, mass or mass distribution, are enough to render seemingly good grasps unstable [243, 8]. Another downside of analytic methods is that they make assumptions to make the calculations tractable, such as hard contacts with fixed contact surfaces [16] and static friction parameters [197]. Due to the lack of robustness and low performances, researchers have recently moved away from analytic approaches towards learning-based methods, which have fewer restrictions.

When using grippers as end-effectors, the problem of finding a successful grasp configuration often reduces to searching geometric structures that fit the gripper's aperture [165, 218, 101, 173, 104, 215]. As opposed to the complexity of dexterous hands which can have three or more fingers, grippers offer an intuitive search space: graspable object parts can be associated with parallel lines, cylindrical structures, handles, etc. Apart from the obvious choice of point clouds [96], 2D intensity images [165] or 3D depth data [101] can be used to detect graspable object sections. While using pre-determined features for grasping is feasible for grippers, it does not scale to grasping with dexterous hands. The complexity of the contacts between an object and a human hand is considerably higher than the case of a gripper (e.g. thumb and index finger). As explained in the next section, learning-based methods have performed significantly better in dexterous grasping.

4.3 Data-Driven Methods

Analytic methods have been successfully applied in grasping problems [22, 70, 76], but they have limited capabilities in dealing with many real-world scenarios, due to the inaccuracies in the sensory data and kinematic models. This lead to a range of techniques that operate under more relaxed assumptions. In this section, a summary of the previous work on learning-based grasping approaches is given. Grasp planning can be divided in 2 main sub-problems, both of which can be solved separately: grasp generation and grasp evaluation. Section 4.3.1 focuses on the methods which process visual information and learn a mapping between the data and candidate grasps. In Section 4.3.2, we list the techniques for learning to evaluate and rank the generated grasps, based on the predicted success. Finally, this section concludes with the most relevant related work to the proposed work, grasping methods based on Deep Neural Networks (DNNs), in Section 4.3.3.

4.3.1 Generating Grasps from Sensory Data

In the case of grasping an unknown object, the object shape data can be obtained using a colour or depth camera. A mapping between the perceived data and the grasp configurations can be learned. Hand-crafted features such as SIFT [189, 60], shape primitives [162] or object sub-parts [108, 39] can be used to represent an object's shape. The complexity of the grasp configuration space depends on the end-effector. In the case of a gripper, only a wrist pose needs to be specified [80], possibly along with the arm joint positions [189]. When performing dexterous grasping with multi-fingered hands, finger joint positions become part of the configuration space as well [3, 189, 105]. There are many papers which learn grasps from the demonstration by a human supervisor [81, 105, 87]. Other works create an offline dictionary of grasps associated with the object features [40]. Given the recent success of feature learning methods, they can replace the hand-crafted features, as in the case of deep learning for grasping [169, 111].

The Generative Model (GM) presented in Section 5.2 is based on the work of Kopicki et al. [105]. The method learns from 10 grasps demonstrated by a human operator, and is able to generate new grasps for previously unseen object-pose pairs. It associates a likelihood with every candidate, which can be used for ranking them and picking the most likely grasp. This ranking has been outperformed by the proposed Generative-Evaluative Model Architecture (GEA) in Section 5.1.

4.3.2 Learning to Evaluate Grasps

Learning-based methods have been applied in the task of predicting whether a grasp is going to succeed or fail. Generative models can sample thousands of grasps efficiently in the matter of seconds [105]. In a real-world setting, the robot needs to pick the grasp which is estimated to be most successful, among the generated options. This task can be approached in a computationally inexpensive manner, by generating a synthetic dataset [142], accompanied by analytic quality metrics computed for each grasp. The collected data can be used to train a learner to predict the likelihood of the candidate grasps in a test case, based on the grasp quality metrics. Mahler et al. [142, 141, 143] created the Dex-Net dataset and used it to train a Grasp-Quality Convolutional Neural Network (GQ-CNN) for parallel-jaw grasping. Analytic metrics are advantegous in the sense that expensive real-robot grasping experiments do not have to be performed in order to collect labeled data. On the other hand, as explained in Section 4.2, these metrics [176] are not good indicators of actual grasp success. Furthermore, scaling from grippers to dexterous hands remains using analytic methods remains an open problem, limiting the application of analytic metrics for complex end-effectors having more than 2 fingers.

A straightforward solution for the grasp evaluation problem is to learn from experience. Candidate grasps can be obtained using a generative model, and each grasp can be performed in the real settings [122, 121, 161, 1, 169, 89] or in simulation [95, 91, 21, 97]. Hyttinen et al. [89] learned grasp stability from tactile information, using 192 labeled grasps. The number of grasps collected increased over time: Wang et al. [231] used the 885-image Cornell Grasp Detection Dataset [1] to train a Convolutional Neural Network (CNN). Pinto and Gupta [161] trained a CNN to predict grasp locations using 50,000 data points. Levine et al. [122] developed a method that successfully learned an end-to-end grasping policy, given uncalibrated coloured images of a robotic arm and objects. They collected 800,000 grasps using 14 identical robots over the course of two months. They also demonstrated the transfer of this knowledge to a similar task in [121], using a second grasp dataset containing over 900,000 grasps, collected using 8 robots of a different model. Similar collective robot learning setups have been explored before [90, 99, 110]. Large-scale data collection setups equipped with robotic arms are rare, simply due to the prohibitive costs of creating such an environment. In addition to this, carrying out the experiments is a time-consuming process, and needs a certain level of supervision. For dexterous grasping, such a setup may be infeasible, due to the fragile structure of some dexterous robot hands (e.g. DLR-II) and the constant need of supervision. For comparison, the data-efficient Generative Model (GM) presented in the next chapter uses only 10 human-demonstrated grasps for training.

The need to collect a high number of grasps stems from the fact that modern machine learning techniques, such as Deep Neural Networks (DNNs), require thousands of samples in order to avoid over-fitting. Recent works have thus replaced parallel robot setups with a simulated environment [71, 95, 91, 21, 192, 184, 216]. With the advancements in physics simulators, it has been possible to re-create robot setups in simulation with high realism and collect the training data in a low-cost manner. Millions of grasps on randomised scenes can be executed on *the cloud* in a very short time, and can be labeled based on their observed success or failure. Physics simulators such as MuJoCo have been able to simulate the physical interactions and contact forces accurately [217]. The real challenge lies in the differences between simulated visual data (colour/depth images) and the data coming from the real robot sensors. Building on the work of Levine et al. [122, 121] for gripper-based grasping using real robots, Bousmalis et al. [21] used simulation to collect additional training data for the architecture of [122]. The discrepancies between the raw RGB images and rendered simulation images were addressed as a domain-adaptation problem, solved with a DNN. Depth images, on the other hand, can hide the challenging appearance features of real objects (textures, lighting, reflections), and are easier to simulate [19]. Gualtieri et al. [71] and Viereck et al. [227] have performed knowledge transfer from simulated depth images to real robot experiments. Similarly, the data collection setup proposed in this thesis uses the Kinect Simulator [19] to produce realistic depth images of objects. The training grasps are performed in an experimental setup based on the MuJoCo physics simulator [217], and the trained model is tested in a real robot environment.

4.3.3 Using Deep Learning for Dexterous Grasping

The applications of Deep Neural Networks (DNNs) in grasping can be classified under three branches: First, a DNN can be used to predict the success probability of a grasp using a gripper configuration [120, 71, 161, 122, 121, 141, 95, 192]. Such methods are not directly comparable with the proposed work in this thesis, since grasping using a gripper is a significantly different problem than one using a dexterous hand. The DLR-II hand used in the experiments in this thesis, seen in Figure 5.1, contains 20 Degrees of Freedom (DoF), vs. a typical gripper with a DoF of 2-4. Second, there have been attempts to *generate* a grasping configuration given an image as an input [169, 111, 226]. Third, DNNs have been used in dexterous grasping [140, 225, 244, 97, 226]. In this section, applications of deep learning in dexterous grasping are of special interest. The reader can refer to Section 2.5 for a detailed overview of DNNs and their applications in computer vision tasks.

Methods using DNNs for Multi-fingered grasping [140, 225, 244, 97, 226] are deemed to be closer to the proposed work in this thesis. All of the cited methods utilise a simulated environment in order to collect the training data. DNNs can be used for training an evaluative model which predicts the probability of success or stability for any given grasp configuration and visual data pair. Kappler et al. [97] trained a CNN to predict the grasp success probability, using the ϵ-metric [55] and a novel physics-based metric. The method requires full points clouds of objects, and the hand pre-shape does not adapt to the specific object or pose during grasping. Varley et al. [225] used a deep network to detect fingertip and palm locations of stable grasps, from single-view RGBD images. Zhou and Hauser [244] designed a network to learn successful grasps for a three-fingered under-actuated hand. Unlike [97], both [225] and [244] are able to modify the hand pre-shape and adapt to a new object's shape and pose. In [226], Veres et al. trained combined generative-discriminative network architecture based on autoencoders to generate grasp locations. Lu et al. [140] proposed a multi-finger grasp planning architecture based on a DNN. [140] does not require an external planner, and initial configurations of grasp parameters are optimised using Gradient Descent in a restricted search space, maximising the probability of success. The authors also released a realistically simulated multi-finger grasp dataset.

Grasping using a partial view of the object is a significantly more difficult problem than grasping using complete object models. The work of Kopicki et al. [105], on which the method presented in this thesis builds, has considerably higher performance when dealing with complete point clouds of objects. Similarly with our approach, the paper of Varley et al. [225] solves the problem of finger placement on incomplete, partial object data for stable grasps. The proposed evaluative CNN predicts a grasp success probability for an input grasp, along with the partial object view. In another method which is close to ours in spirit, Song et al. [206] devised a probabilistic model for discovering grasp contact points, supervised by a human teacher. Among the papers which are considered to be most related, only two have presented results in real robot experiments [225] and [140]. Varley et al. [225] have achieved 75% grasp success on 8 test objects, while Lu et al. [140] reported 84% performance on 5 test objects. In comparison, the proposed methodology has produced comparable results (77.7%) on 40 previously unseen objects.

The presented method in this part of the thesis consists of two building blocks: A generative model which creates the candidate grasps, and an evaluative model which ranks the generated configurations. The generative model, based on [105], has a clear advantage over the analytic methods used for grasp generation in the literature: It works with partial shape data from a single view, compared to the full 3D model requirements of the analytic methods. Our method is based on local contact surfaces and discards the global shape information, making it easier to match to new objects and improving the generalisation ability. Once a high number of grasps are generated, they are provided as input to the evaluative model. The evaluative model consists of a novel CNN architecture, which is trained based on simulated data. The top grasp, based on the ranking proposed by the evaluative model, is performed. A comparison with the original ranking calculated by the generative model is given in the experimental section. The method is tested both in simulation and on the real robot. The results, explained in Section 5.6, clearly show that the network is able to transfer the grasp success prediction ability from simulation to the real robot. To our knowledge, this is the first technique which attempts to combine a *learned* generative architecture with a *learned* evaluative model in dexterous grasping from a single view.

In the next chapter, a summary of the proposed Generative-Evaluative Model architecture is given, along with the experimental results in simulation and real world. The method is being prepared for a submission for the prestigious International Journal of Robotics Research (IJRR).

CHAPTER 5

DEXTEROUS GRASPING OF NOVEL OBJECTS FROM A SINGLE VIEW WITH A GENERATIVE-EVALUATIVE LEARNER

Grasping novel objects using given a single-view image of the scene using a dexterous hand is a challenging task. The complexity of the problem arises from multiple sources. First, a dexterous hand, such as the human hand, has an inherently complicated structure which makes it difficult to plan grasps in a precise manner. Grasping is not a fully pre-planned action. It is rather a adaptive process which relies on the tactile feedback coming from the hand. If the object is heavy, a firmer grip is required. If it's slippery, higher-friction areas are sought. Second, there are many hidden constraints. A mug full of liquid should be handled differently than an empty one, since a full mug has more constraints during the handling. Digging deeper, even a mug full of hot water can be grasped differently than one with cold one, since the consequences of spilling water onto a finger are different for these cases. Third, the object shape is only partially known, and the rest of the object shape has to be predicted. Humans are pretty good at predicting the hidden parts of objects due to years of training. In order to get closer to human performance, robots need to deal with dexterous hands, different object characteristics and partial views. The solution presented in this chapter plans a full grasp trajectory for an object given a partial view.

The proposed grasping system consists of two distinct units: a *learned* generative

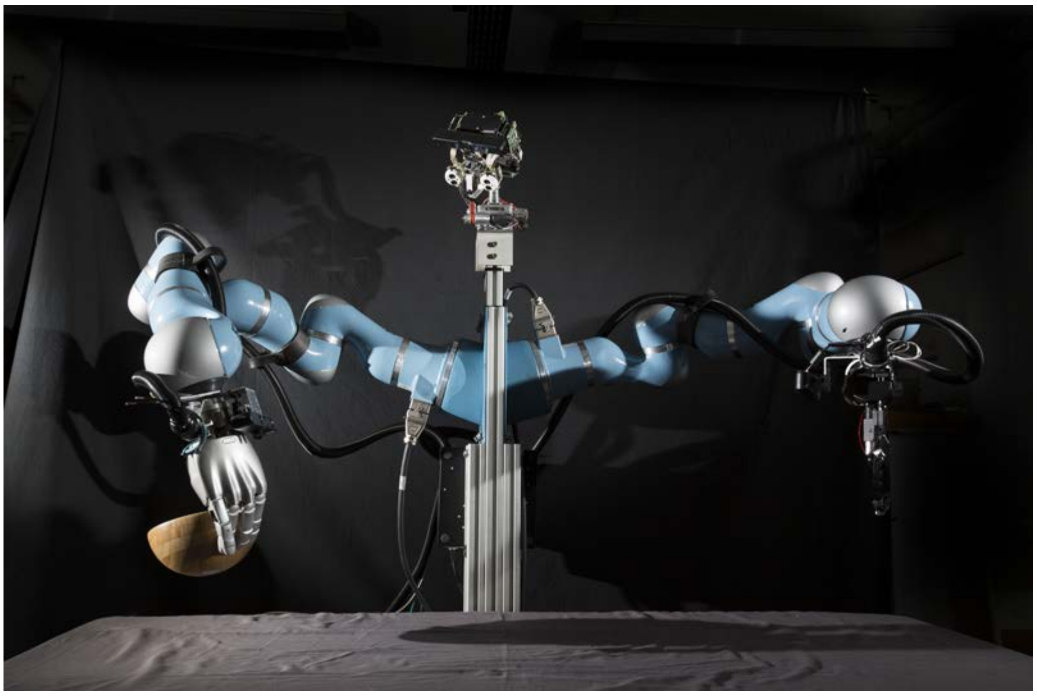


Figure 5.1: The Boris robot platform on which the methods were tested.

module which creates plausible candidate grasps, and a *learned* evaluative module which ranks the candidate grasps based on their predicted success. The generative module is based on the work of Kopicki et al. [105]. It learns from a number of demonstrated real grasps, and the model is able to generalise over unseen objects. A single-view depth image of an object is provided as input to the system, resulting in a number of plausible candidate grasps. These grasps are then performed in a realistic simulation environment, and grasp success, as well as grasp stability parameters are recorded. Using the grasp success data, an evaluative Convolutional Neural Network (CNN) is trained. Once the trained system encounters a new object, first, a number of candidate grasps are generated. Then, the CNN is used to predict the success probability of all generated grasps, and the grasps are ranked based on their predicted success. If the test is run in simulation, the top-ranked grasp in which the dexterous hand does not collide with the simulated table is performed. In the real-world tests, the grasps which are not kinematically feasible due to the robot configuration are removed, in addition to those colliding with the table. Figure 5.1 shows the Boris robot platform where the grasps were executed.

This chapter is organised as follows. First, the proposed hybrid generative-evaluative learning architecture is explained in Section 5.1. In Section 5.2, the current state-of-the-art in grasp generation for dexterous grasping of novel objects using a single-view depth camera is presented. The proposed method in this chapter builds on this work. Next, in Section 5.3, a detailed overview of the proposed generative-evaluative learning architecture is given. This architecture combines a learning system with low data constraints with a deep network, which typically needs more data. We used a simulation environment to create the data to train the neural network in a very short time. Section 5.4 is about the features and limitations of this simulation environment. Section 5.5 is dedicated to the analysis of both methods on simulated data. The real-world experiments and analysis can be found in Section 5.6. Finally, the chapter concludes with a discussion presented in Section 5.7.

5.1 Hybrid Generative-Evaluative Model Architecture

The approach in learning dexterous grasping of novel objects consists of two modules. The first component is the generative module $h = g(I_i)$, which creates a number of grasp configurations $h = \{h_i\}_{i=0}^{H}$. The input for the generative module is a depth image from a single view. h contains H *candidate* grasps in the frame of reference of the camera used to acquire I_i. Each candidate grasp configuration h_i contains a series of waypoints, with each waypoint representing the hand configuration at some point along the grasp approach trajectory.

The second module is a evaluative model $f(I_i', h_i)$, which takes the colourised image I_i' and a candidate grasp h_i, and outputs the predicted grasp success probability of the grasp h_i. The single-channel image I_i is converted into a three channel image I_i' with the addition of mean curvature and gaussian curvature channels to the depth image.

The proposed hybrid generative-evaluative architecture is given in Figure 5.2. The

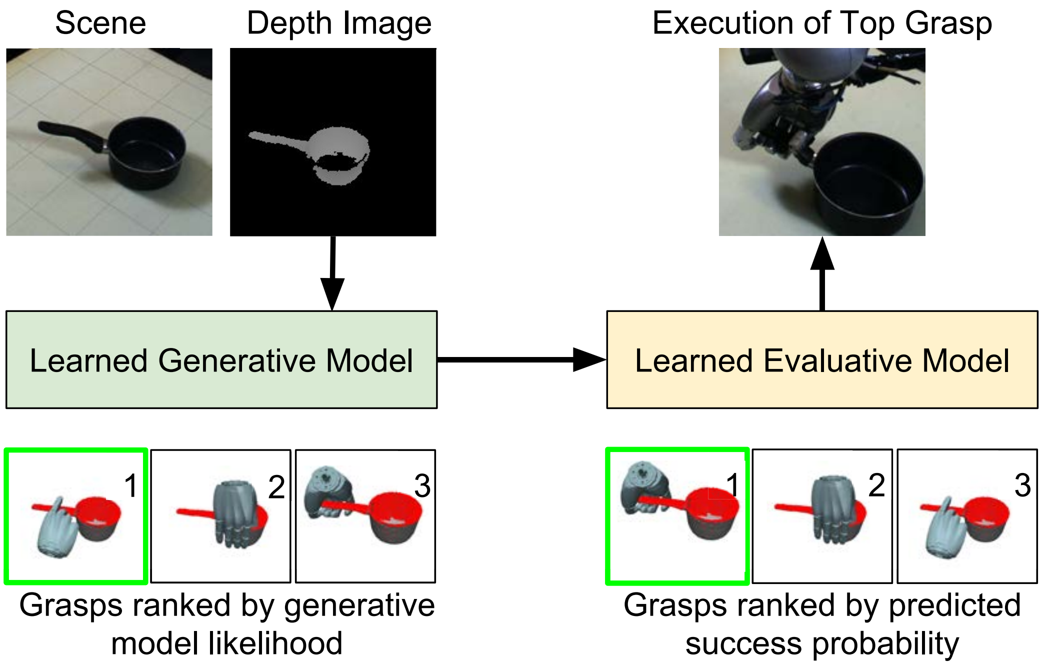


Figure 5.2: The proposed hybrid generative-evaluative learning architecture. First, a depth image of the scene is acquired (top left). The generative model, trained by a human demonstrator, generates candidate grasps and ranks them based on their model likelihood (bottom left). The candidate grasps are re-ranked by the evaluative model, trained using simulated data (bottom right). The top grasp which is *kinematically feasible* is executed by the robot (top right).

Generative Model is based on the Learning from Demonstration (LfD) principle [105]. The Evaluative Model consists of a deep network and requires more training data. A realistic simulation environment was created to obtain the data to train the evaluative network. The scene in Figure 5.2 is a typical test case of the algorithm, using the trained models. The core contribution of the paper is to combine two learned models to generate plausible grasps and rank them based on the predicted success. Next, the Generative Model is given.

5.2 Grasp Generation Model Learning and Transfer

The Generative Model (GM) architecture, which creates the candidate grasps, is explained here. While the author of this thesis did not contribute to the work in this section, it is included to preserve the self-sufficiency of the manuscript. This section builds on the grasp generation model of Kopicki et al. [105], who performed learning of generative models for dexterous grasps from a human demonstrator. [105] is an extremely data-efficient learning method which can generalise over unseen objects from a few training grasps. Learning is achieved in 3 stages: learning the model, transferring the model onto new objects, and grasp generation. Below, the method is explained step by step.

5.2.1 Model Learning

The model learning stage is the step where learning from demonstration occurs. During the demonstration by a human operator, three different types of information is collected: First, the object model is acquired. Next, a contact model for each finger link is learned, using the contacts between the object model and the hand links. Finally, a hand configuration model is obtained. The object model is discarded after training.

Object model

First, a depth camera acquires the point cloud of the scene from a single view. The (partial) point cloud of the object is augmented with the estimated principal curvatures and the normal vector of the local surface at that point. For example, the j^{th} point in the cloud is represented by a triple $x_j = (p_j, q_j, r_j)$, where p_j is a position in $\mathbb{R}^3$, q_j is the surface normal in $SO(3)$, and $r_j = (r_{j,1}, r_{j,2}) \in \mathbb{R}^2$ are the principal curvatures. The object model is thus defined as a kernel density estimate of the density over x:

$$O(x) \simeq \sum_{j=1}^{K_O} w_j \mathcal{K}(x|x_j, \sigma) \tag{5.1}$$

where the bandwidth is $\sigma = (\sigma_p, \sigma_q, \sigma_r)$, and K_O is the number of points in the object's point cloud. All weights $w_j = 1/K_O$ are identical. $\mathcal{K}$ is given below in a product of experts formulation (a product of distributions):

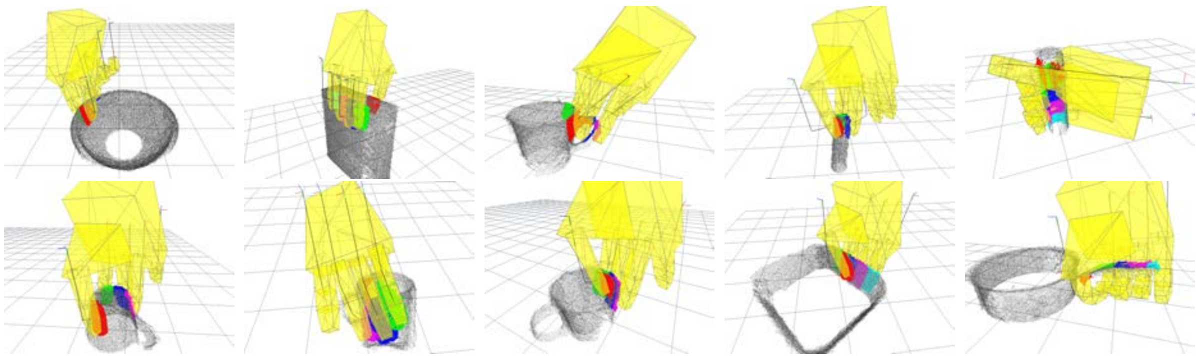


Figure 5.3: The training grasps used for training the generative model. The object models are shown in black, the final pose of the hand is in yellow, and the partial point clouds in each contact model are shown in different colours.

$$\mathcal{K}(x|\mu, \sigma) = \mathcal{N}_3(p|\mu_p, \sigma_p)\Theta(q|\mu_q, \sigma_q)\mathcal{N}_2(r|\mu_r, \sigma_r) \tag{5.2}$$

where μ is the kernel's mean, σ is the bandwidth, $\mathcal{N}_n$ is an n-variate isotropic Gaussian kernel, and Θ represents a pair of antipodal von Mises-Fisher distributions.

Contact models

During the demonstration of a grasp, the final hand pose, along with the object's model, is recorded. From this data, all finger links L and the corresponding partial object features, which are in the vicinity of each link, are found. For each link L_i in the hand, a new contact model M_i is learned. The object features which are closer to the link L_i than a distance threshold δ_i contribute to the contact model M_i. The contact model M_i is defined as:

$$M_i(x) \simeq \frac{1}{Z} \sum_{j=1}^{K_{M_i}} w_{ij} \mathcal{K}(x|x_j, \sigma_x) \tag{5.3}$$

where K_{M_i} is the number of object features in the vicinity of L_i, Z is a normalisation constant, w_{ij} is a weight that exponentially decreases as the distance between the closest point a_{ij} on link L_i and object feature $x_j = (p_j, q_j, r_j)$ rises:

$$w_{ij} = \begin{cases} \exp(-\lambda||p_j - a_{ij}||^2) & \text{if } ||p_j - a_{ij}|| < \delta_i \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

As it can be seen from the definition above, the contact models depend on local surface features, not the entire object. These local features can be detected on new objects' point clouds, which means that the learned contact models are transferrable between the objects. In this thesis, principal curvatures r are used for defining surfaces; however, other surface descriptors can be used instead of the principal curvatures method. The contact models are, in essence, agnostic to the choice of the surface descriptors. The quality of the descriptors affects the quality of the contact models, since the descriptor is used for matching the local surfaces encountered during training to the tested model. Next, we explain how the kinematic constraints of the robotic hand model contribute to the learning process.

5.2.2 Hand Configuration Model

During the demonstration of a grasp, the configurations of the hand $h_c \in \mathbb{R}^D$ is saved at a number of points along the trajectory, where D is the number of Degrees of Freedom (DoF) in the hand. The collected information includes the position of the hand, as well as the finger joint positions. The hand configuration model is defined as:

$$C(h_c) \equiv \sum_{\gamma \in [-\beta, \beta]} w(h_c(\gamma)) \mathcal{N}_D(h_c | h_c(\gamma), \sigma_{h_c}) \tag{5.5}$$

where $w(h_c(\gamma)) = \exp(-\alpha \| h_c(\gamma) - h_c^g \|^2)$, γ is an interpolation parameter between the start (h_c^t) and the end (h_c^g) points of a trajectory. $w(h_c(\gamma))$ is controlled via Eq. 5.6 below, and the β parameter allows hand configuration extrapolation.

$$h_c(\gamma) = (1 - \gamma) h_c^g + \gamma h_c^t \tag{5.6}$$

5.2.3 Transferring Grasps

The learned contact models can be transferred to a new object o_{new}. First, a single-view depth image of the object o_{new} is acquired, and converted to an augmented point cloud with the added features, as explained earlier. The resulting object features are recast as a density O_{new}, using Eq. 5.1. A learned contact model M_i can be transferred onto the new object o_{new} by convolving M_i with O_{new}. The convolution operation is approximated with a Monte-Carlo procedure, and the outcome is a kernel density model of the pose s of the finger link L_i for the object o_{new}. As a result of the Monte-Carlo procedure, sampled poses for the link L_i are obtained. The j^{th} sample is represented with $\hat{s}_{ij} = (\hat{p}_{ij}, \hat{q}_{ij})$. Each sample $\hat{s}_{ij}$ is assigned a weight w_{ij}, which is equal to it likelihood.

Collectively, the generated samples are used to construct a query density estimate:

$$Q_i(s) \simeq \sum_{j=1}^{K_{Q_i}} w_{ij} \mathcal{N}_3(p|\hat{p}_{ij}, \sigma_p) \Theta(q|\hat{q}_{ij}, \sigma_q) \tag{5.7}$$

where $\sum_j w_{ij} = 1$. A different query density for every contact model and object pair is built. Next, we explain how the hand configuration model and query densities together generate the grasps.

5.2.4 Grasp Generation

In this section, the grasp generation process is explained. The three given elements are: a query density k, a sample $s_k \sim Q_k$, and a hand configuration model sample $h_c \sim C$. Coupled with the kinematics of the hand, these samples represent a full grasp $h = (h_w, h_c)$, where h_w is the wrist pose, and h_c is the hand joint configuration. The sampled grasp h is then optimised using a hill-climbing procedure:

$$\arg\max\,(h_w, h_c) C(h_c) \prod_{Q_i \in \mathcal{Q}} Q_i \left(k_i^{\text{for}} \left(h_w, h_c \right) \right) \tag{5.8}$$

where k_i^{for} is the forward kinematic function of the hand. The hill-climbing procedure optimises the grasp to maximise the likelihood. This likelihood can be considered as an analytic method for measuring grasp quality, and is later shown to be inferior to a learning based method for measuring grasp success.

During the grasp generation and optimisation, non-promising grasps with low likelihood are periodically pruned. The pruning step can be run for multiple times as the optimisation progresses (**2** times in the implementation used in this thesis), which allows the algorithm to finish in a reasonable amount of time without compromising the quality of the grasps. Typically, a different generative model is learned for each training grasp and view pair, and all models can be used to generate grasps when a new point cloud is given. The generative model ranks the candidate grasps coming from all models based on their likelihood, which is calculated using the product of experts methodology. The idea that this ranking may be sub-optimal in picking the best possible grasp inspired the evaluative work in this chapter. The proposed generative-evaluative architecture predicts a grasp success for each grasp, and in turn, uses it to re-rank the grasps. The original ranking presented in this section serves as the baseline method in the experiments.

5.2.5 Training Grasps for This Study

In the experiments presented in this thesis, the Generative Model (GM) was trained with ten example grasps, as shown in Figure 5.3. For each grasp, seven views of the considered object were acquired, and a different model was trained for each grasp-view pair. As a result, the algorithm learned 70 models for generating grasps, coming from 7 views per each of the 10 grasps. GM acts as an ensemble of these models. Compared to [105], the Generative Model used in this thesis contains two improvements. First, for each view, a pre-processing step filter out the points on the object's surface model whose surface normals differ from the surface normal of the finger link by more than +/- 90 degrees. Second, a two-stage candidate grasp selection scheme was implemented. Among the grasps considered by GM, half were selected globally from the top (most likely) grasps, regardless of the type. The other half were forced to spread evenly among all 10 grasp types, ensuring that all grasp types are available for the training of the evaluative models, explained in the next section.

5.3 Evaluative Model Architectures

In this section, two different evaluative model architectures are explained. The grasp generator model, given in the previous section, requires very little training data. It can generate 1000 candidate grasps, ranked according to their estimated likelihoods, within 20 seconds on a 2x Intel Xeon E5-2650 v2 Eight Core 2.6GHz. The generative model does not estimate a probability of success for the generated grasps. An evaluative model, which is a Deep Neural Network (DNN), is used specifically for this purpose. DNNs have shown good performances in learning to evaluate grasps using grippers [122, 120] and dexterous hands [224, 140].

The generative method ignores the global information about the object, such as the overall shape and the object class, to a large extent. It takes the robot hand model and the local object shape features into account. Moreover, it only has access to the partial object shape, due to the noise in the image acquisition process and self-occlusions. The success of an executed grasp, however, depends on other factors such as the full object shape, mass, mass distribution, surface friction and deformability, among others. An evaluative network indirectly learns the effect of these parameters on the grasp outcome. The data provided to the evaluative network is collected from randomly generated scenes, therefore each scene has a different random combination of the parameters. The primary purpose of the network is to learn robust grasps across different conditions, and this is a complex task. The first challenge is that the kinematic model of the hand is unknown to the evaluative network. It only has access to the parameters that *configure* the hand: the wrist and joint positions. Second, the system is weakly supervised with the grasp result (success or failure), and no further labels are provided.

Both proposed evaluative models have similar characteristics. The evaluative DNN based on the VGG-16 network [202], named Evaluative Model 1 (EM1), is given in Figure 5.5. A slightly different version based on the ResNet-50 network, termed Evaluative Model 2 (EM2), is shown in Figure 5.6. [1] Regardless of the type, a grasp evaluation network has the functional form $f(I_t, h_t)$, where I_t is a colourised depth image of the object, and h_t contains a series of wrist poses and joint configurations for the hand, converted to the camera's frame of reference. The network's output layer calculates a probability of success for the image-grasp pair I_t, h_t. The model initially processes the grasp parameters and visual information in separate channels, and combines them to learn the final output.

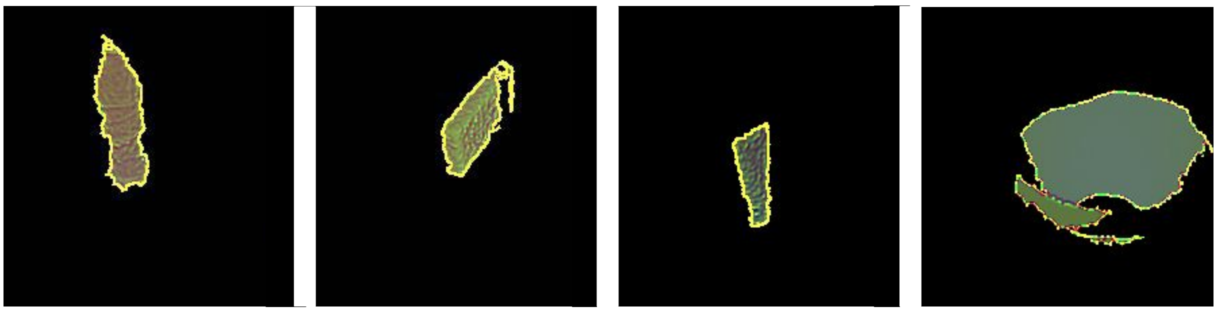


Figure 5.4: Colourised depth images. From left to right, the objects are: coke bottle, chocolate box, hand cream, and bowl.

The depth image is colourised before it is passed as input to the evaluative network. The colorisation process converts the single-channel depth data to a 3-channel RGB image. We first crop the middle 460×460 section of the 640×480 depth image, and downsample it to 224×224. Then, two more channels of information are added: mean and Gaussian curvatures, obtained from the object's surfaces in the image. Figure 5.4 contains four examples of colourised depth images. This procedure both provides a meaningful set of features to the network, and makes the input directly compatible with VGG-16 and ResNet, which require images of size $224 \times 224 \times 3$.

The grasp parameter data h_t consists of 10 trajectory waypoints represented by $27 \times 10 = 270$ floating point numbers, and 10 extra numbers reserved for the grasp type. Each of the 10 training grasps is treated as a different class, and h_t uses the 1-of-N encoding system. Based on the grasp type ([1-10]), the corresponding entry is set to 1, while the

[1]The design of the networks belongs to the author. They were implemented and trained in TensorFlow by Chao Zhao, a MSc student collaborating with Umit Rusen Aktas and Prof. Jeremy L. Wyatt.

rest remain 0. The grasp parameters are converted to the coordinate system of the camera which was used to obtain the corresponding depth image. The transformed parameters are processed using a Fully-Connected (FC) layer consisting of 1024 nodes, and the output is *element-wise added* to the visual features. The joint visual features and grasp parameter data are processed together in higher layers.

All FC layers have RELU activation functions, except for the output layer, which has softmax activations. The output layer has two nodes, corresponding to the success and failure probabilities of the grasp. A cross-entropy loss is used to train the neural network, as given in Eq. 5.9.

$$H_{y'}(y) := -\sum_i (y'_i \log(y_i) + (1 - y'_i)\log(1 - y_i)) \tag{5.9}$$

where y'_i is the class label of the grasp, which is either 1 (success) or 0 (failure), and $y_i = f(I_i, h_i)$ is is the predicted label of the grasp pair (I_i, h_i).

The proposed evaluative models EM1 and EM2 share the common features explained above. The models are introduced below. Only their unique properties are highlighted.

5.3.1 Evaluative Model 1 (EM1)

Figure 5.5 demonstrates the architecture of the first proposed evaluative network. The colourised depth image is processed with the VGG-16 network [202] to obtain the high-level image features. The VGG-16 network is initialised with the weights obtained from ImageNet training. Only the last three convolutional layers are trained, and the first 13 layers remain frozen. This decision was made in order to speed up training.

The grasp parameters and image features pass through fully-connected layers with 1024 hidden nodes (FC-1024) layers in order to obtain two feature vectors of length 1024. The features are combined using the element-wise addition operation, and are further processed using 4 FC-1024 layers. Similarly with [122], the features are combined using addition and not concatenation. In this thesis, this decision was made based on the

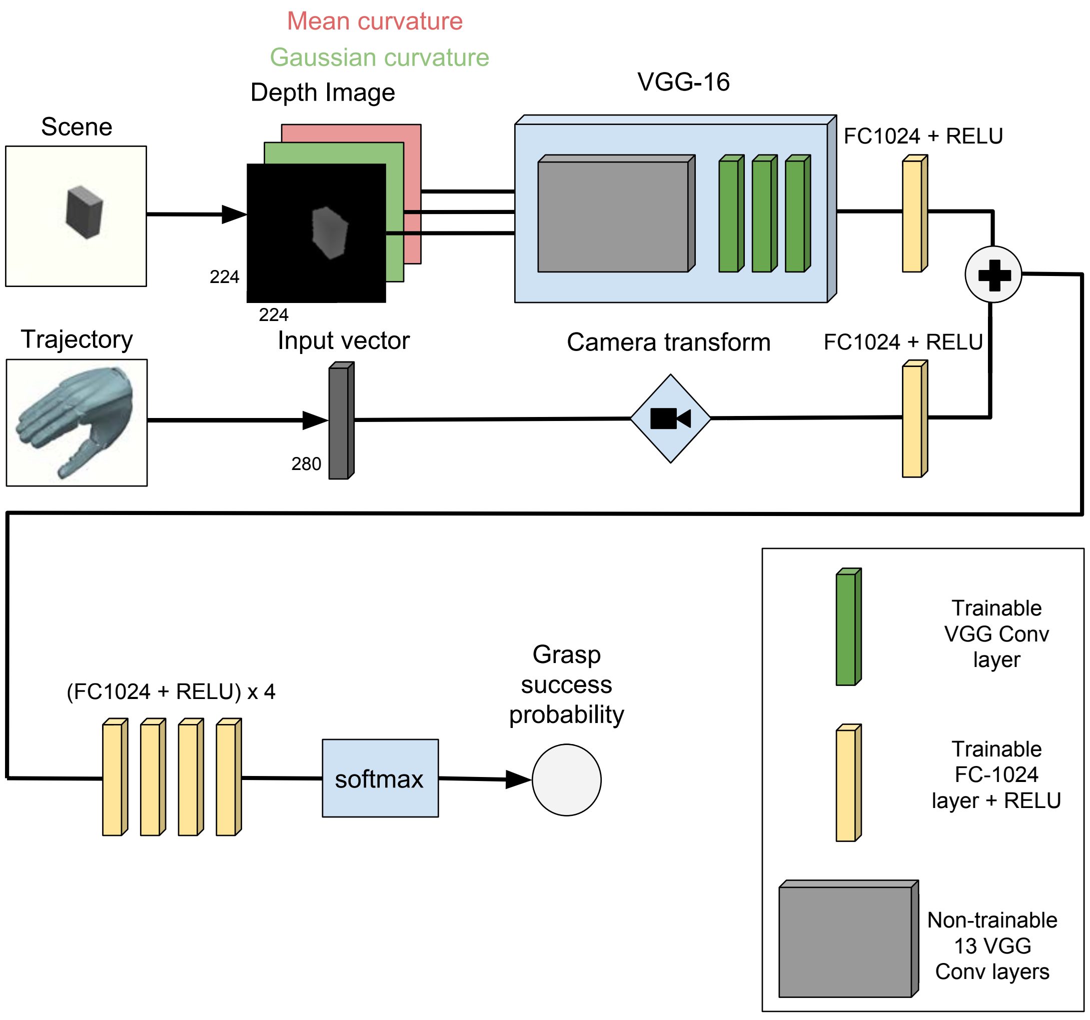


Figure 5.5: The VGG-16-based evaluative deep neural network architecture. The first 13 convolutional layers of VGG-16 are not trained. Similarly to Levine et al. [122], the two channels of information (grasp parameters and visual features) are joined via element-wise addition. The combined information is then passed through four fully-connected layers activated with RELU. The final softmax layer has grasp success and and failure nodes, and learns to predict the success probability of a grasp.

observation that addition yielded a marginally better performance in the experiments (not included in this thesis). Furthermore, concatenation and addition can be considered as interchangeable operations when combining different information pathways in deep networks [44]. The final FC-1024 layers form the associations between the visual features and hand parameters, and contain most of the parameters in the network.

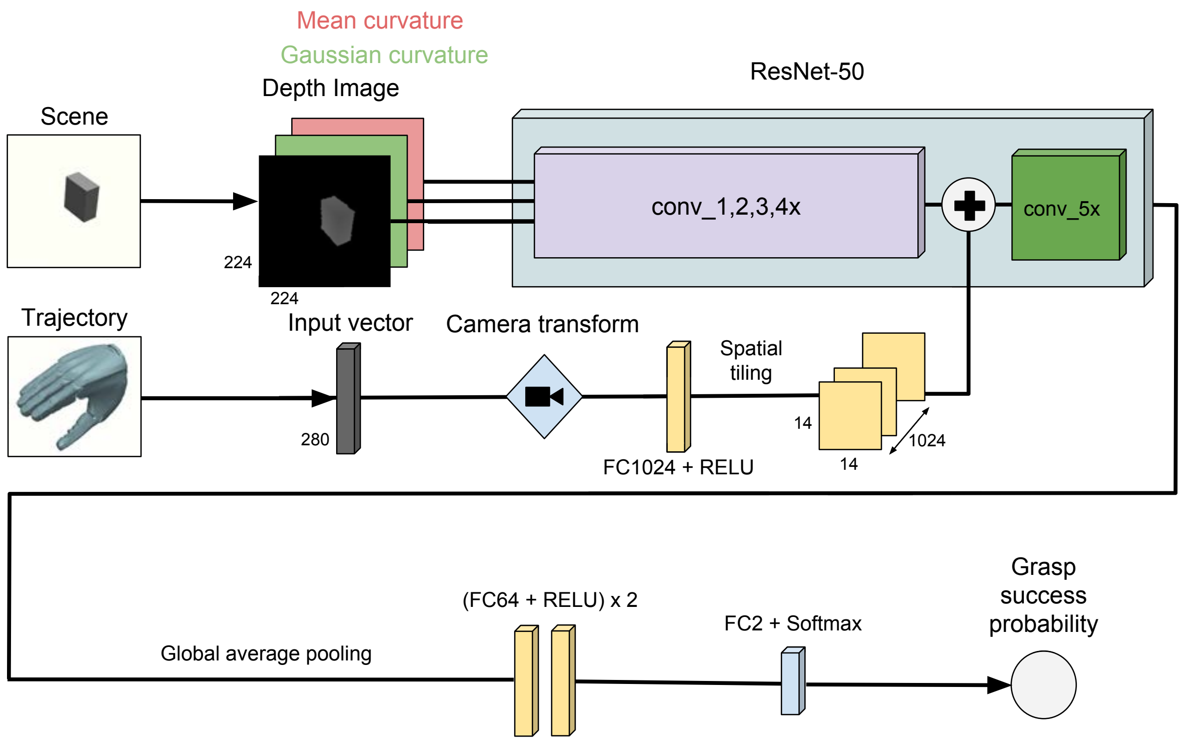


Figure 5.6: The ResNet-based evaluative deep neural network architecture (EM2). Spatial tiling is used to repeat the grasp parameters before they join the image processing pathway. This network requires fewer FC layers due to the earlier marriage of information channels.

5.3.2 Evaluative Model 2 (EM2)

The second evaluative model, termed EM2, uses the ResNet-50 network in order to obtain the high-level image features. Similarly with VGG-16, the ResNet-50 architecture is initialised with the weights that have been obtained during ImageNet training. One exception is the last convolution block conv_5x, which is initialised randomly. In the EM2 architecture, ResNet-50 network is broken down into two parts: the first 4 convolutional blocks are used to extract the visual features. The final convolutional block, which has 9 convolutional layers, combines the image features and grasp parameters. Similarly with EM1, element-wise addition joins the two channels of information. Spatial tiling is used to convert the processed grasp parameters, a vector of size 1024, to a matrix of size $14 \times 14 \times 1024$. Because the last convolutional block conv_5x processes combined information, this network is designed with only 2 FC layers with 64 hidden nodes each.

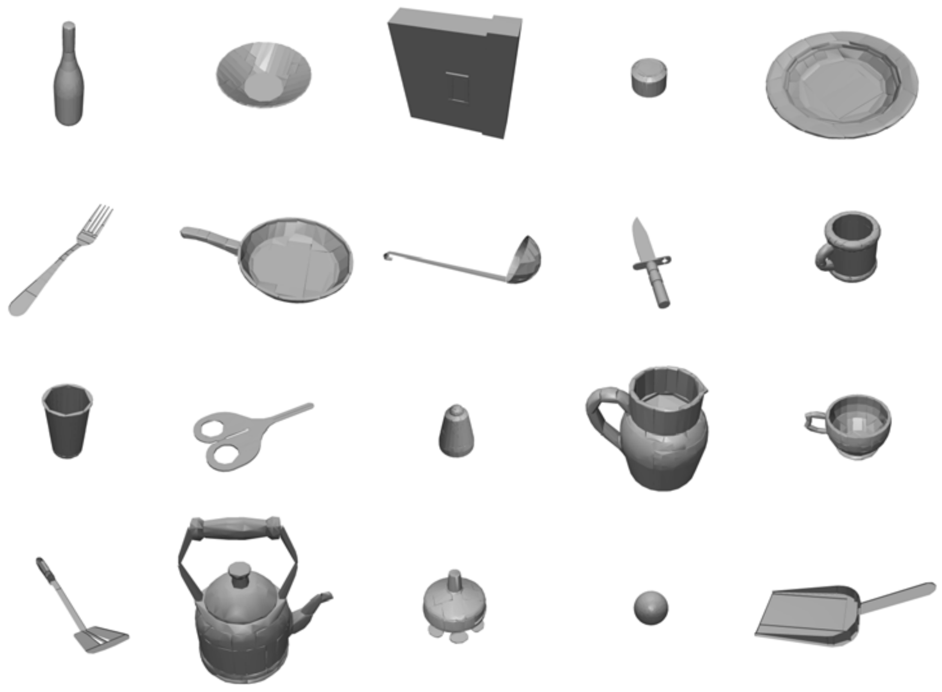


Figure 5.7: Representative object images from all 20 classes in the 3D model dataset. The objects were chosen from graspable everyday classes, and each class contains between 1-25 objects.

The output layer is the same as EM1.

Both networks were trained on simulated data and tested on the real robot, as well as in simulation. The generative-evaluative architectures which use EM1 and EM2 are called GEA1 and GEA2, respectively. The next section focuses on the collection of the training data.

5.3.3 Data Collection Methodology

The data collection process runs in batches, and each batch can be considered as a *scene* with a single object placed on a table. A simulated depth camera takes an image of the scene from a random angle and distance, and a partial, view-based point cloud of the object is obtained. This point cloud is used to generate a number of candidate grasps using the generative approach GM. The grasps are performed consecutively in the scene, and the object is placed in its initial pose before a grasp is performed. Below, an outline of the data collection process in a single scene is given:

1. A new instance of an object from one of the 20 classes in the 3D model dataset is generated, and placed on the virtual table with a random pose. The object is placed at a random location within the $10 \times 10\ cm^2$ square area in the middle of the table. The scale, weight and surface friction parameters of the object are sampled from realistic distributions.

2. A depth image I_s is acquired using a simulated depth camera, which looks at another random point within the $10 \times 10\ cm^2$ square area in the middle of the table. The $elevation_s$ of the camera location is sampled from the range [30, 57] degrees, while the $azimuth_s$ is sampled from the range $[0, 2\pi]$. The distance of the camera location from the centre of the table is between 45 and 75 centimetres. The acquired depth image is converted to a point cloud P_s, and a colourised image I'_s.

3. The candidate grasps $h = \{h_i\}_{i=0}^{K}$ are generated using the generative model GM. During training, 10 top-ranked grasps (according to the likelihood given by the model) from all 10 grasp types are selected for execution on the object in simulation. In the real robot experiments, the top 100 grasps from each one of the 10 grasp types are evaluated, which gives the evaluative network a better chance of finding a successful grasp.

4. In order to increase the variation in the dataset, more simulated point clouds of the object are generated. Up to 20 new simulated camera positions are created according to the procedure in step 2. Figure 5.8 shows the sampled camera positions in an example scene. Each grasp is randomly associated with a camera view sampled in this step.

5. Each candidate grasp is executed in the same scene. The image, wrist parameters, finger joint angles and the grasp outcome are stored for every candidate grasp. The grasp parameters which consist of the wrist parameters and the joint angles are converted to the frame of reference of the associated view.

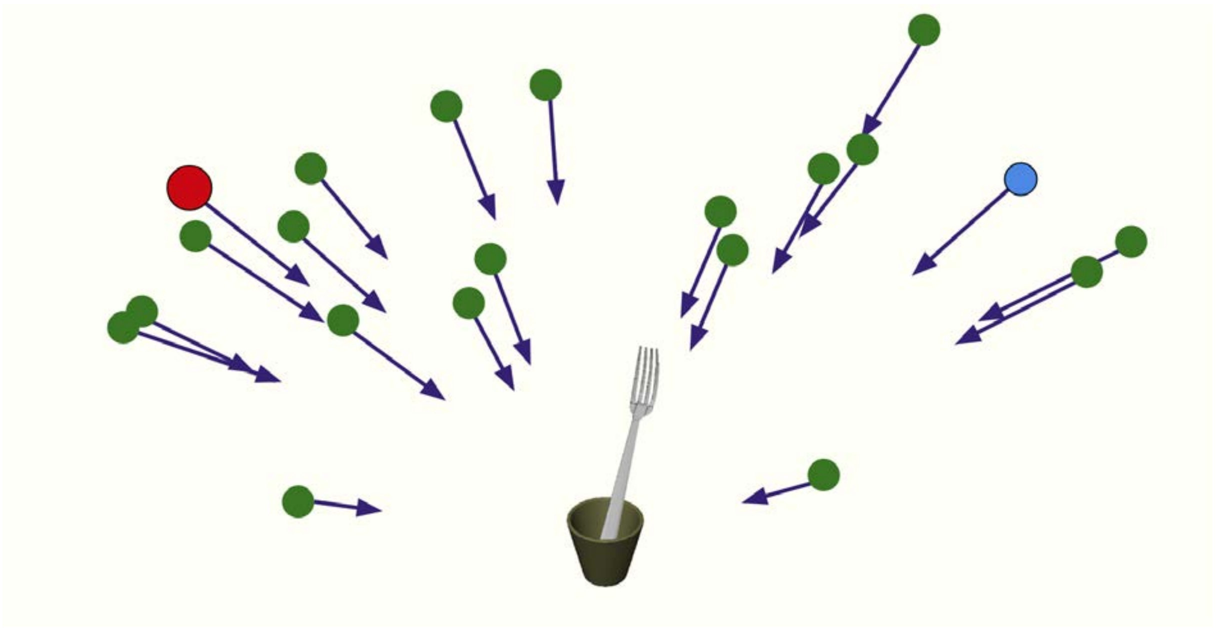


Figure 5.8: In simulation, 20 distinct camera poses are sampled. Each green point illustrates a sample camera location, whereas the gaze directions are shown with purple arrows. The actual camera pose on Boris is fixed, and demonstrated with the red point (larger). One of the views is randomly picked to be the reference view (blue), and the point cloud of the object which feeds to the generative model is obtained using this view.

A candidate grasp $h_i = \{w_{i0}, ..., w_{in}\}$ consists of a series of waypoints w_{i0}, ..., w_{in}. A waypoint w_{ik} is a 27-element vector that specifies the full configuration of the hand in the joint space: 3 dimensions for the 3D coordinates and 4 dimensions for the orientation of the wrist (in quaternions), and 20 parameters specifying each finger joint's activation. Typically $n = 10$ waypoints are used to represent a grasp trajectory. The grasp parameters are created in the world coordinates, hence they need to be converted to the corresponding view's frame of reference. With the point cloud and grasp parameters represented in the camera's point of view, the key goal of the deep network is plan robust grasps by learning the relationship between the point cloud, grasp parameters and grasp success. Using the camera pose differentiates the proposed method from [122], where camera coordinates are not used. It should be noted that the possible camera locations in this study covers a much larger space, as illustrated in Figure 5.8. Converting the data to the camera frame is thus necessary.

In each scene S_i, up to 20 depth images $\{I_{ik}\}_{k=0}^{19}$ are acquired, as explained above.

Figure 5.8 illustrates the sampled views in a scene with a fork. In addition to the reference view (shown in blue) used to generate the point cloud, 19 views are picked around the object. The multi-view approach ensures that the dataset has significantly more variation than it would have had if only the actual camera pose on the real robot (shown in red) were to be used. The increased number of views also means that the evaluative networks are trained on a more difficult problem than the one solved in the real experiments. The candidate grasps are created using the reference depth image I_{i0}, obtained from the reference camera pose. It should be noted that while up to 100 grasps are executed in each scene, only 20 depth images are acquired. This is a compromise between using a single view for all grasps and obtaining a different point cloud for each grasp. In most cases, performing a grasp takes less time than acquiring a simulated point cloud.

When a candidate grasp is performed, if the object is lifted 1 metres above the virtual table and held there for 2 seconds, it is considered as success. If, at any point, the hand fails to grasp the object or drops it, the grasp is marked as a failure. The next section explains the steps taken to ensure the simulation matches reality as much as possible.

5.4 Features and Constraints of the Virtual Environment

It is time-consuming and expensive to collect grasping data with real robots. Levine et al. [122] have created large-scale data collection setups with robotic arms and grippers. Not only this is a very expensive option, it is also infeasible for dexterous grasping. Compared with grippers, dexterous hands can be much more fragile due to their complexity. A possible alternative to the parallel robot setups is to create a simulation environment for data collection, thanks to advances in physics simulators. A simulated experimental setup was constructed in order to execute the grasps, which allowed the collection of the required data in a short period of time with no supervision. A particularly important aspect of the work is that the collected dataset is intended to capture the uncertainty coming from unobservable variables, such as mass and friction. In the proposed setup, a wide range of masses and friction parameters were incorporated in order to prepare a dataset where *robust* grasps can be identified.

The data collection procedure starts by creating a scene which has a single object on a virtual table, as well as a depth camera looking at a point near the object. The camera acquires a depth image of the object, and the point cloud is segmented from the background. It is then processed by the Generative Model (GM) [105] to generate the candidate grasps. Each candidate grasp is executed in the initial state of the scene. The core characteristics of the simulator setup are given below.

- The data collection procedure was carried out using MuJoCo [217] as the base physics simulator and visualiser.
- The 3D model dataset consists of a total of 294 objects from 20 graspable classes. Additionally, the objects are scaled randomly in each scene, practically creating an infinite number of objects.
- The 3D models used in the data collection process had to be decomposed into convex parts using the V-HACD algorithm [146]. MuJoCo necessitates this step in order to correctly calculate the collisions between an object and the hand links (fingers, palm).
- The distributions for the object weights, sizes and frictional coefficients were determined from real-world data. These distributions were used to sample the properties of the objects in simulated scenes.
- A realistic 3D mesh-model of the DLR-II hand has been used in the simulator. There are no constraints on how the hand can approach an object, other than the collisions with the virtual table.
- A simulated depth camera imitating the Carmine 1.09 depth sensor installed on the Boris platform has been developed. For this purpose, modifications were made to

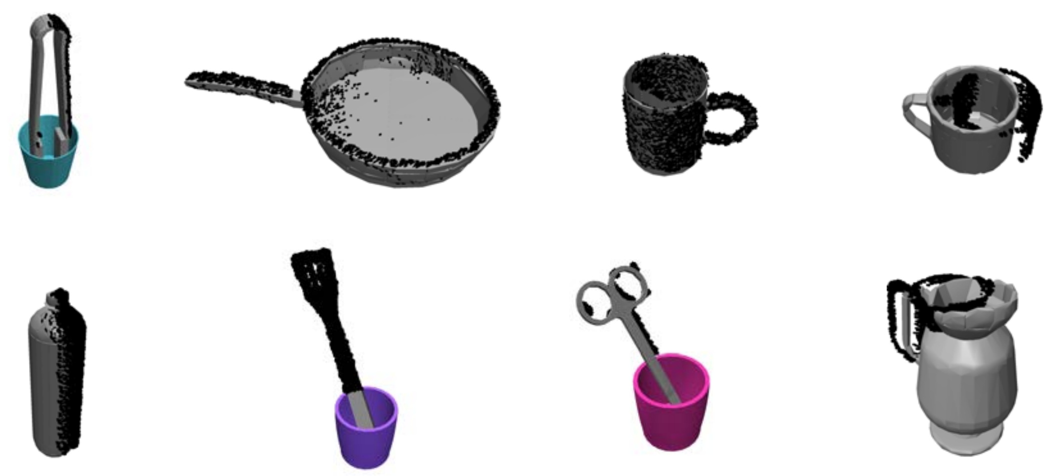


Figure 5.9: In order to account for the calibration errors in real experiments, calibration noise is added to the simulated point clouds. From left to right, each column contains two simulated point clouds with added noise, sampled from Normal distributions of $\mu = 0$ and $\sigma = 0.1$, 0.2, 0.5 and 1.0 centimetres, respectively. After the point cloud is acquired, a noise vector $< d_x, d_y, d_z >$ with d_x, d_y, d_z sampled from $N(\mu, \sigma)$ is added to the positions of every point. Each point in the point cloud is shifted by the same noise vector. In the collected dataset for this thesis, μ is 0 and σ is set to 0.4.

the Blensor Kinect Sensor Simulator [19]. Additional calibration errors were added to the point clouds (See Figure 5.9) in order to match reality as much as possible.

- As well as the mass, scale and friction coefficients of the placed object, other variables such as camera location, direction and distance to the object have been assigned randomly in each scene. The object location and pose varies from scene to scene as well.

- In order to fit the real-world robot setup, the DLR-II hand in simulation is controlled using an impedance controller.

The reasons for some of these critical decisions are now given in slightly more detail. First, in order to create a realistic simulation environment, the MuJoCo [217] physics simulator was chosen over other simulators (OpenSim, BulletPhysics, ODE, NVIDIA PhysX) for two reasons:

- MuJoCo uses generalized coordinates and optimization-based contact dynamics, resulting in fewer numerical instabilities,

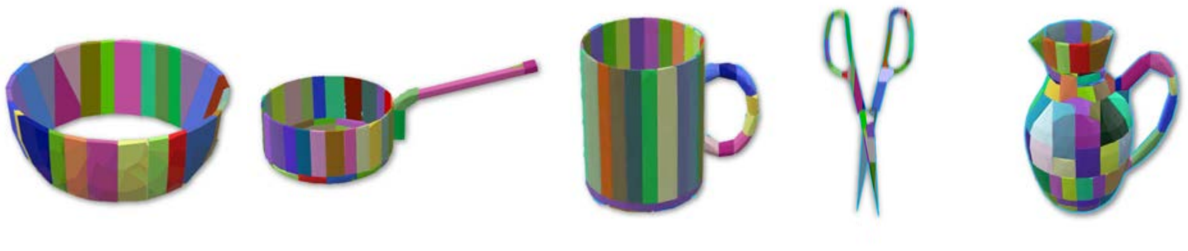


Figure 5.10: Approximate convex decomposition of some objects in our dataset.

- MuJoCo is optimized for the quality of physics as well as its speed, hence improving the quality of the physics simulation.

MuJoCo requires objects to consist of convex parts for accurate collision detection, hence all of the objects in the 3D model dataset have been decomposed into convex parts using V-HACD algorithm [146]. Volumetric Hierachical Convex Decomposition (V-HACD) algorithm creates an approximate decomposition of an object, and Figure 5.10 illustrates the results of the method on some of the objects in the dataset. Depending on the complexity of the 3D model, the number of sub-parts of an object in the dataset varies between 2-120.

The collected 3D model dataset contains 20 classes, namely, bottles, bowls, cans, boxes, cups, mugs, pans, salt and pepper shakers, plates, forks, spoons, spatulas, knives, teapots, teacups, tennis balls, dustpans, scissors, funnels and jugs. All objects in the dataset can be grabbed using the DLR-II hand, although there are limitations on how some object classes can be grasped. For example, teapots and jugs are not easy to grasp except by their handles, while small objects such as salt and pepper shakers can be approached in more creative ways. The number of objects in each class differs between 1 (dustpan) and 25 (bottles). Of the 294 total objects, 250 from all 20 classes were allocated for training, while the remaining 44 objects from 19 classes belong to the test set. Most of the objects are everyday kitchen objects. Long and thin objects such as spoons, knives, scissors, forks and spatulas are placed vertically in water cups so that it is possible to grasp them without touching the table (See Figure 5.9). In the real-world experiments, the same approach was applied to place such objects vertically, as attempting to grasp

Bottle	Bowl	Box	Can	Cup	Fork	Pan
30-70	50-400	50-500	200-400	30-330	40-80	150-450
Plate	Scissors	Shaker	Spatula	Spoon	Teacup	Teapot
40-80	50-150	100-160	40-80	40-80	150-250	500-800
Jug	Knife	Mug	Funnel	Ball	Dustpan	
80-200	50-150	250-350	40-80	50-70	100-150	

Table 5.1: Weight ranges for each object class (grams).

them while they lie on the table would be dangerous.

Realistic scales, weights and friction parameters were used for the objects in simulation. Each object in the dataset has a reference size, which is in line with the expected object sizes in the real world. Before an object is placed in a scene, it is randomly scaled by a factor in the range [0.9, 1.1]. This adds extra variance in terms of object sizes to the data collection process, while the limited scaling factors ensure that all objects remain within the grasp aperture. Similarly, during the scene generation procedure, each object instance is assigned a weight coming from an estimated distribution, obtained from real objects. The weight distribution for each class is defined as a uniform distribution within a range $[min_w, max_w]$, limited by the minimum min_w and maximum max_w weights encountered in that class among real examples. Table 5.1 shows the weight ranges for each class in *grams*. The weight ranges for the classes were set by making assumptions about the dominant materials each class can be constructed of. Such assumptions include metal pans and teapots, ceramic/porcelain mugs and teacups, plastic/wood kitchen utensils, and plastic/ceramic bowls. The bottles are assumed to be empty and cans are supposed to be filled with reasonable amounts of food.

Furthermore, the friction coefficients of the object in each scene is sampled from the range $[0.5, 1]$ in MuJoCo default units, which covers a wide range of materials from slippery surfaces (metal, porcelain) to high-friction surfaces (rubber, wood). Unlike object weights, the friction coefficient is sampled from the same distribution for every object, regardless of the class. The variability in weight and friction parameters ensure that data collection process accounts for these unobservable variations, and robustness of the grasps across different object characteristics and noise conditions becomes important. Figure 5.11

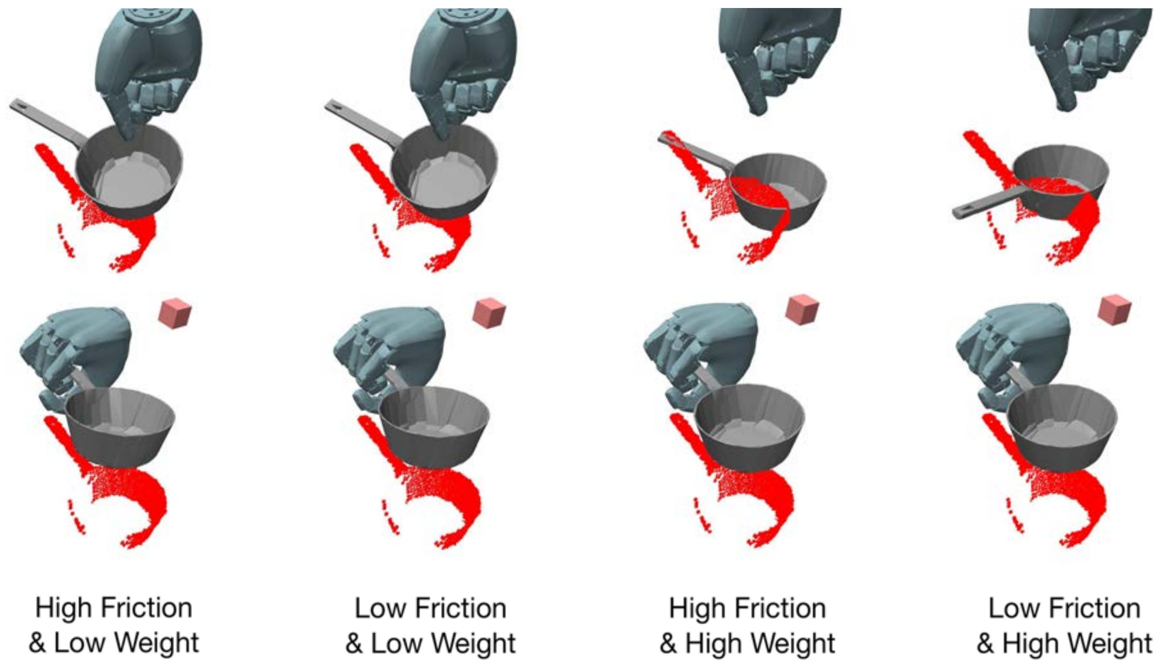

High Friction & Low Weight
Low Friction & Low Weight
High Friction & High Weight
Low Friction & High Weight

Figure 5.11: Training a robust evaluation model. (Top row) The same pinch grasp, executed on the same object, with varying friction and mass parameters. (Bottom row) A more robust power grasp, executed on the same object, with the same variation in friction and mass.

illustrates the effect of these unobservable variables on grasp success in simulation.

During the scene creation, the object is placed on the virtual table at a random pose, in order to increase the variability. Most objects are placed in a canonical upright pose, and only rotated along the gravity direction (akin to being placed on a turntable). The objects belonging to the mug and cup classes have fully random 3D rotations applied before they are placed on the table, since it is possible to grasp them in almost any setting using the robot hand. As mentioned before, the kitchen utensil classes, along with pairs of scissors, are placed vertically to avoid the inevitable collisions which would occur if they were placed horizontally on the table.

In Section 5.5, the findings of the analysis on the collected simulation data are described.

Bottle	Bowl	Box	Can	Cup	Fork	Pan
35.5 - **47.7%**	26.4 - **61.2%**	16.5 - **30.1%**	41.4 - **92.6%**	44.7 - **59.9%**	59.6 - **68.1%**	37.9 - **57.3%**
Plate	Scissors	Shaker	Spatula	Spoon	Teacup	Teapot
50.2 - **95.5%**	62.7 - **69.9%**	47.3 - **53.3%**	57.4 - **65.7%**	63.4 - **82.4%**	48.2 - **91.2%**	26.9 - **23.9%**
Jug	Knife	Mug	Funnel	Ball	Dustpan	**Total**
24.9 - **43.9%**	58.3 - **65.0%**	40.7 - **80.9%**	52.3 - **65.9%**	28.0 - **82.8%**	60.1 - **78.8%**	45.8 - **63.2%**

Table 5.2: The average and **top** grasp performances of the Generative Model (GM) on simulated, unbalanced data.

5.5 Experiments and Analysis in Simulation

Two generative-evaluative architectures, GEA1 (integrating EM1) and GEA2 (with EM2) have been tested in both simulated and real-world experiments. In this section, we analyse the performance of the networks in simulated data. The training data collected in simulation consists of 1,081,332 candidate grasps on 20,025 scenes. It has equal numbers of successes and failures, and the networks were trained for 20 epochs using the training data. The trained evaluative network was used to predict the outcomes of the grasps in the test set. The test data contains 86,130 grasps in 1,561 scenes, and consists of 42,173 successful and 43,957 failed grasp attempts. The average number of grasps stored per scene is 54, and each scene contains no more than 10 grasps per each of the 10 grasp types. Roughly a third of the candidate grasps are not considered, since they result in collisions between the hand and the virtual table.

The object classes in the dataset have different characteristics in terms of grasp affordances. Some classes are easier to grasp than others. Table 5.2 shows the success rates of the generated grasps in each class, when attempted with the grasps ranked by the Generative Model (GM) [1]. The sampled grasps perform well on a number of classes including Dustpans, Scissors, Spoons, and Mugs. Some objects can only be grasped in certain ways. To an extent, GM can identify successful grasps with its likelihood-based ranking. The advantage of the GM is evident in classes with high top-grasp and low average-grasp performances, such as Balls, Mugs and Cans. The grasps which results in collisions with the table are ignored.

[1]The dataset used in this analysis is a superset of the training data mentioned in the above paragraph, hence actual percentages may differ.

The success prediction accuracy of GEA1 is 73.8% on the test data, while GEA2 achieves 77.8%. The top grasp ranked by the Generative Model has succeeded in 1087/1561 test scenes (69.6%). In contrast, the top grasp as ranked by the the first generative-evaluative architecture GEA1 has succeeded in 1300/1561 (83.3%) of the test scenes. The second proposed method GEA2 performs even better, with a successful top grasp in 1389/1561 scenes (90%). The proposed methods show significant improvements in simulation.

	#		%	
	PS	PF	PS	PF
GTS	35701	6472	85%	15%
GTF	16075	27882	37%	63%

Table 5.3: Grasp success predictions for GEA1.

	#		%	
	PS	PF	PS	PF
GTS	35141	7032	83%	17%
GTF	12058	31899	27%	73%

Table 5.4: Grasp success predictions for GEA2.

Table 5.3 shows another result of our analysis on the simulated data for GEA1. In the table, successful grasps are listed under GTS, or “Ground Truth Success”. Similarly, failed grasps are marked as GTF, “Ground Truth Failure”. On the vertical axis, PS and PF stand for “Predicted Success” and “Predicted Failure”, respectively. They mark the grasps predicted by GEA1 method to be a success or failure. Figure 5.3 shows that the proposed method GEA1 has high recall in the success category (85%) and low recall in identifying failed grasps (63%), resulting in a higher number of false positives (GTF and PS) than false negatives (GTS and PF). In this context, recall is the ratio of the samples in a ground truth class which have been correctly identified by the network. The second network, GEA2 in Figure 5.4, substantially improves the method’s performance by improving the rate of false positives by 10% (37% to 27%). These results illustrate the need to have a good generative model rather than a random grasp generator, since the number of false positives would likely increase in the case of low-quality grasp samples.

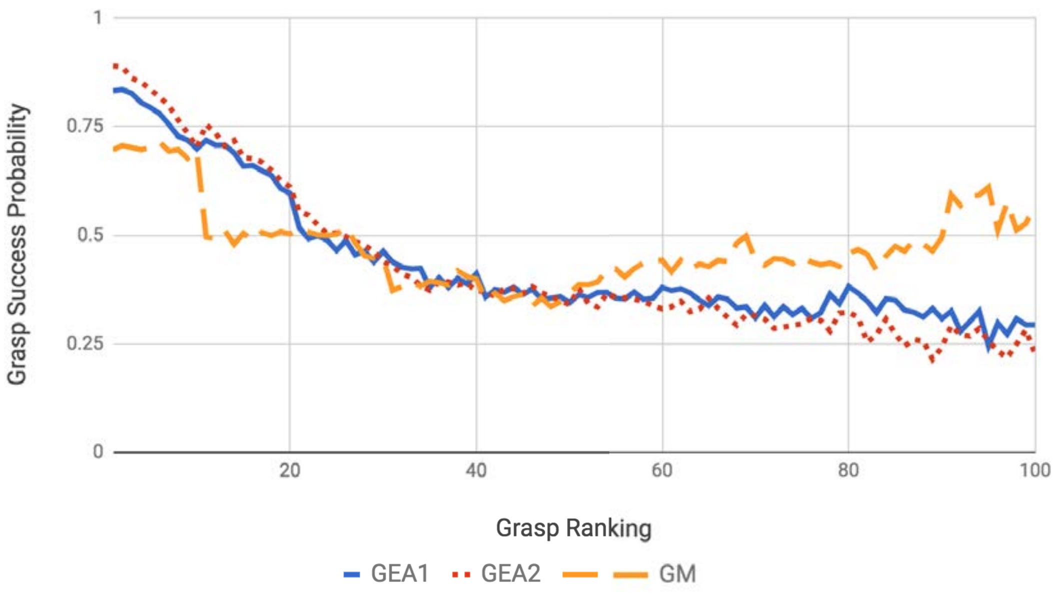


Figure 5.12: Grasp success probability vs. grasp ranking in simulation.

The desired behaviour of an evaluative method is to rank grasps based on their predicted success, monotonically from high to low. All three methods explained in this thesis, the Generative Model (GM) and the proposed generative-evaluative models (GEA1, GEA2) produce a grasp ranking. GM orders the grasps based on their likelihood, calculated using the finger joint positions and the object's point cloud, while GEA1 and GEA2 learn to predict grasp success probabilities. Figure 5.12 illustrates an analysis of the grasp success rates in simulation vs. grasp ranking, according to all three methods. The figure shows that the ranking produced by GEA2 is the best at identifying successful grasps, and the probability falls nearly monotonically as the rank increases. The ranking of GEA2 is slightly better than GEA1. The baseline method, GM, misses many successful grasps and places some of them at the end of the list.

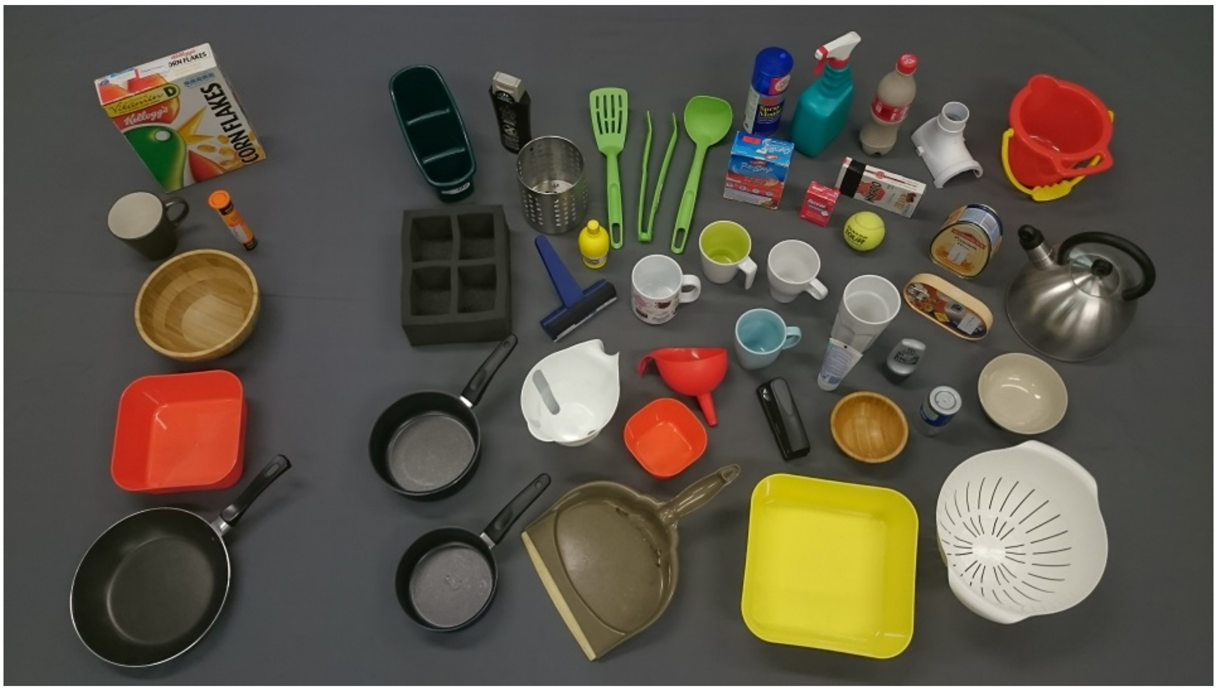


Figure 5.13: The object dataset used in real robot experiments. The training objects are given on the left column, while the test objects are in the middle and on the right.

5.6 Robot Experiments

The two proposed generative-evaluative methods (GEA1, GEA2) are compared with the baseline Generative Model (GM), since GM represents the state-of-the-art in dexterous grasping of novel objects using a single-view depth camera. Since the original paper by Kopicki et al. [105] had a 77.7% success rate, a more difficult test set was created, which contained new objects in more challenging poses. The new test set contained 40 test objects and 6 training objects, as shown in Figure 5.13. The baseline method GM was trained by demonstrating ten example grasps on the 6 training objects. Figure 5.3 illustrates the 10 training grasps, all of which were performed with the assistance of a person. 49 real scenes were constructed by placing the 40 test objects individually on the table in challenging poses. The object pose with respect to the camera greatly affects performance, since all compared algorithms rely on single-view depth images to generate grasps. Finally, 35 out of the 40 test objects are of object classes contained in the simulated data, while the rest are not.

The Generative Model (GM) and generative-evaluative architectures (GEA1, GEA2) were compared using paired trial experiments. Each model was tested on identical object-pose combinations. During the testing phase, the top *kinematically possible* grasps in all scenes were executed. It should be noted that after the grasps are ranked, those which are not possible or safe for the robot to perform are eliminated. The top-ranked grasp that passes this filtering is executed. A grasp is considered to be successful if the object is still in the robot hand, after being lifted for 5 seconds, and held stable in hand for a further 5 seconds before the automatic release. As a result of the experiments, the success rate for GM was 57.1%, for GEA1 was 77.6%, and for GEA2 was 75.5%. Table 5.5 summarises the number of successful and failed attempts for all tested algorithms. Both proposed methods outperform the generative model substantially. GEA1 has a statistically significant result with a p-value of 0.0442 and GEA2 comes close with 0.0665, where p-value is the result of a two-tailed McNemar test. The McNemar test accounts for the difference between the results of two methods. The slight performance gap between GEA1 and GEA2 is not statistically significant. The difference between GEA1 and GEA2 is likely due to the robot's calibration issues, as the experiments were performed two months apart. The end result of 12 grasps where GM and GEA1 had different outcomes are shown in Figure 5.14. A video which contains the grasp results of GM and GEA1 can be accessed here.

		GM	
		# succs	# fails
GEA1	# succs	23	15
	# fails	5	6
GEA2	# succs	23	14
	# fails	5	7

Table 5.5: Real-robot paired comparison trial results.

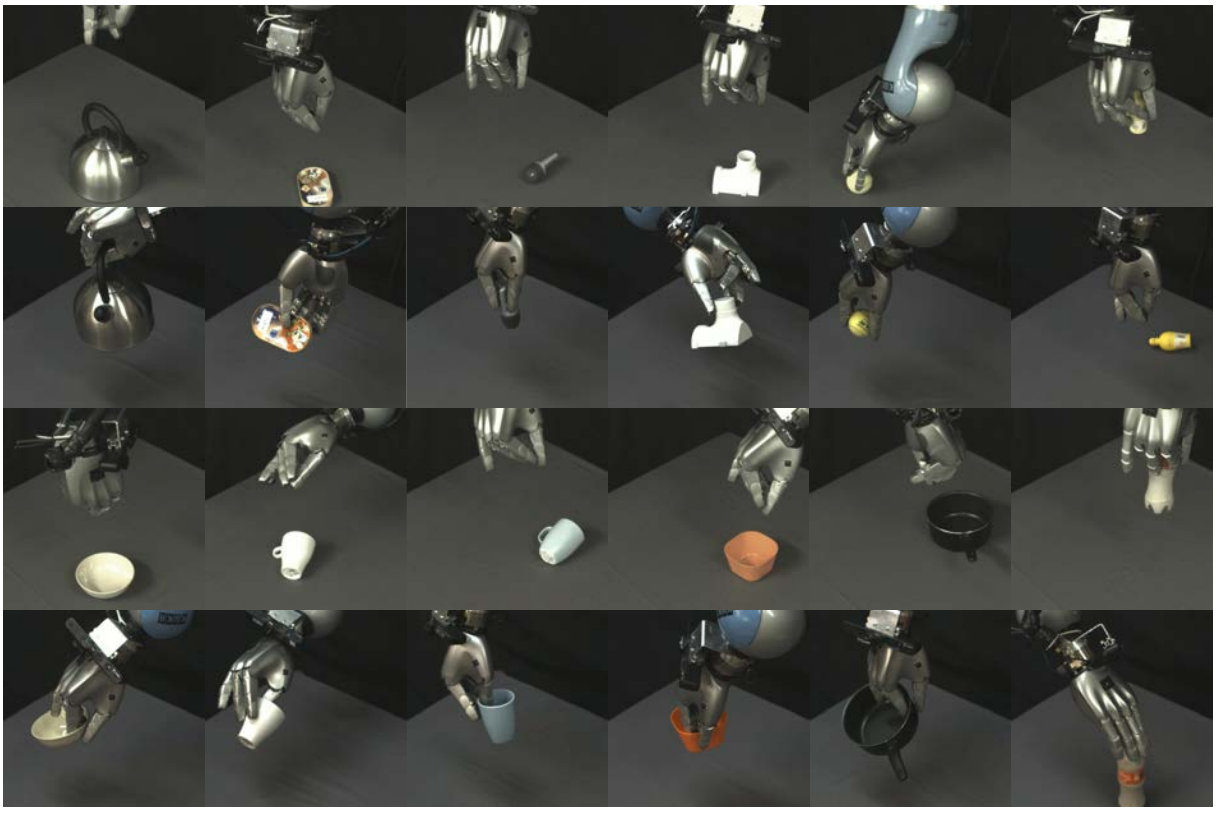


Figure 5.14: First and third row demonstrates the grasps recommended by the Generative Model (GM). Similarly, second and fourth row grasps come from the 1st Generative-Evaluative Model Architecture using the VGG-16 network (GEA1). Both methods were given identical object-pose pairs as input. In the first 5 columns, we illustrate the cases where GEA1 succeeds and GM fails. The final, sixth column shows 2 of the 5 reverse cases where GM succeeds and GEA1 fails.

5.7 Discussion

A novel architecture consisting of one learned generative and one learned evaluative model was presented. The generative model is a highly data-efficient method that learns from demonstration by a human operator. Ten example grasps were performed, along with seven different views considered per grasp, resulting in 70 grasp-view pairs. The generative model is an ensemble, containing one model per each grasp-view pair. When encountered with a test image, the generative model is used to generate sample grasps, and provides a good prior which pre-filters out improbable grasps.

During training, a randomised simulation environment is used for obtaining realistic scenes in simulation. The generative model, given the simulated depth data, creates a set of candidate grasps. These grasps are executed in simulation and each one is labeled with the recorded grasp outcome, which is either a success or a fail. The results are then used to train the evaluative deep networks which predict the success probability of a given grasp. Two evaluative networks were proposed and tested in both simulation and real environments. Both networks were trained by using more than 1M training grasps, which were cheaply obtained with the help of the simulation environment. The created artificial scenes include varying observable and hidden parameters, including object scale, friction, weight and camera distance and pose, among others. In the testing phase, the hybrid generative-evaluative architecture was used to generate a set of candidate grasps, and these grasps were ranked using an evaluative model. The top kinematically-feasible grasp was performed in every scene.

The experimental evaluation showed that the proposed methods improve the grasp success rates considerably, and statistically significant results at 0.05 were obtained. The proposed hybrid model outperforms the generative method in both simulation and real experiments, illustrating that it successfully transfers the knowledge from simulation to the real robot. A future extension of the work is to include colour images in the architecture. The discrepancies between the artificial and real colour images can be addressed by domain adaptation methods. Another interesting direction is to remove the generative method entirely, and have the network predict a series of grasps in the order of decreasing success probability.

CHAPTER 6

CONCLUSION

This thesis explored two novel deep architectures in shape analysis and grasping domains. The first one, CHOP, is a hierarchical compositional method for statistical learning of 2D shapes. The second is a new generative-evaluative method for dexterous grasping which has two competing evaluation models based on Deep Neural Networks (DNNs). Both presented frameworks have multi-layer representations and learn hierarchical features. Although compositional hierarchies and DNNs are similar in spirit, their building blocks and learning mechanisms are different. Below, a summary of the proposed work is given, followed by the key findings of this thesis. Future directions for the work are explained in Section 6.3.

6.1 Summary

First, the advantages of deep methods over flat recognition systems were established earlier in the thesis. Biological reasons and current empirical evidence suggest that deep learning is able to bridge the gap between concepts (class labels) and data (images) more successfully than flat, feature-based methods.

The Compositional Hierarchy of Parts, named CHOP, was designed for the statistical analysis of shapes. It starts with simple line-based features and learns a multi-layer feature hierarchy by recursively compressing the initial representation of the data. It employs graph-theoretic and information-theoretic tools to create a principled, graph-based framework. Learning is executed in a bottom-up fashion: Each layer is trained by compressing the data represented by the previous layer. The method performs well in shape retrieval and classification tasks in benchmark shape datasets. The analyses show that the learned vocabularies exhibit sub-linear growth as the number of objects, views and classes increase.

In the second part of the thesis, a learned generative-evaluative model for dexterous grasping was proposed. The generative model used for generating grasps is the work of Kopicki et al. [105], and it is the current state-of-the-art. Two competing DNN-based evaluative models were proposed. In order to create the training data required for the data-intensive evaluative models, a data collection setup was created to execute the generated grasps in simulation. The evaluative models learn to rank the grasps by predicting their success probabilities, after training on the simulated data. The results convincingly show that the proposed architectures improve the grasp success rates substantially. The evaluative methods successfully transfer the knowledge from simulation to real experiments.

6.2 Key Findings

Below are two direct observations which are the results of the explorations:

1. It has been demonstrated that compositional hierarchies learn compact shape representations, and are well-suited for shape retrieval and classification problems. As shown in the experimental analysis of CHOP, the learned shape models exhibit sub-linear vocabulary growth and part sharing properties.

2. A generative-evaluative architecture was proposed for dexterous grasping of previously unseen objects from a single view depth image. The experimental analysis of this method validates two key assumptions. First, an evaluative DNN improves

the grasp success rate by re-ranking the generated grasps based on their predicted success probabilities on novel objects. Second, the networks are able to transfer the knowledge from a simulation environment to the real robot.

Further discoveries can be made by comparing the two different families of algorithms at a conceptual level:

- The learned parts via a compositional hierarchical method can easily be decomposed into their sub-parts in intuitive ways. As a result, the representations are interpretable and sparse, e.g. a corner is made by two lines perpendicular to each other. DNNs lack interpretability as the networks are densely convoluted, as opposed to the sparse structures of compositional methods.
- Compositional hierarchies mostly rely on feed-forward training. This is in sharp contrast with deep networks, which are trained via a feedback mechanism (Back-propagation). As a result, DNNs can be optimised generatively or discriminatively, and often find better solutions in the solution search space, resulting in high performance.

6.3 Future Work

An interesting research direction for deep architectures is to create a hybrid approach that combines the strengths of compositional hierarchies and DNNs. Recently, Tabernik et al. have shown that the notion of compositionality can be used to reduce the number features in DNNs and improve training times [213, 212]. Another possible way to combine these methods is to use compositional techniques as means for unsupervised pre-training of deep networks. First, the common representations in the data can be discovered via a compositional hierarchy. It can be argued that these representations can be encoded as initial weights in a DNN with the same structure, and discriminative training can be used to fine-tune the DNN. It can be hypothesised that the pre-training would reduce the training and convergence times of DNNs.

The extensions of the grasping work involves going beyond an evaluative approach that ranks the generated grasps. An end-to-end learner, which takes an image as input and generates a ranked list of grasp configurations with decreasing predicted success probabilities is planned. The end result is a grasp generation network. The training procedure of this end-to-end generative network may still be coupled with an evaluative network. Multiple iterations of interleaved training (generate, collect data, evaluate) may be needed to tune the system.

LIST OF REFERENCES

[1] Cornell grasping dataset. `http://pr.cs.cornell.edu/grasping/rect_data/data.php`. Accessed: 2018-04-23.

[2] U. R. Aktas, M. Ozay, A. Leonardis, and J. L. Wyatt. A graph theoretic approach for object shape representation in compositional hierarchies using a hybrid generative-descriptive model. In *2014 European Conference on Computer Vision (ECCV)*, pages 566–581, 2014.

[3] H. Ben Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters. Generalization of human grasping for multi-fingered robot hands. In *International Conference on Intelligent Robots and Systems*, pages 2043–2050. IEEE, 2012.

[4] F. Bach, J. Mairal, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(4):791–804, 2012.

[5] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[6] X. Bai, W. Liu, and Z. Tu. Integrating contour and skeleton for shape classification. In *2009 IEEE International Conference on Computer Vision (ICCV)*, pages 360–367, Sept 2009.

[7] X. Bai, C. Rao, and X. Wang. Shape vocabulary: A robust and efficient shape representation for shape matching. *IEEE Transactions on Image Processing (TIP)*, 23(9):3935–3949, Sept 2014.

[8] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka. Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *IEEE Transactions on Robotics*, 28(4):899–910, Aug 2012.

[9] D. H. Ballard. Readings in computer vision: Issues, problems, principles, and paradigms. chapter Generalizing the Hough Transform to Detect Arbitrary Shapes, pages 714–725. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[11] Y. Bekiroglu, K. Huebner, and D. Kragic. Integrating grasp planning with online stability assessment using tactile sensing. In *2011 International Conference on Robotics and Automation (ICRA)*, pages 4750–4755. IEEE, 2011.

[12] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(4):509–522, April 2002.

[13] Rodrigo Benenson. Are we there yet? `http://rodrigob.github.io/are_we_there_yet/build/`. Accessed: 2018-03-21.

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, Aug 2013.

[15] A. Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research (IJRR)*, 14(4):319–334, August 1995.

[16] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *2000 IEEE International Conference on Robotics and Automation*, volume 1, pages 348–353 vol.1, 2000.

[17] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[18] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales. Mind the gap - robotic grasping under incomplete observation. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 686–693, May 2011.

[19] J. Bohg, J. Romero, A. Herzog, and S. Schaal. Robot arm pose estimation through pixel-wise part classification. In *IEEE International Conference on Robotics and Automation (ICRA) 2014*, pages 3143–3150, June 2014.

[20] Y-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2566. IEEE, 2010.

[21] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *CoRR*, abs/1709.07857, 2017.

[22] G. I. Boutselis, C. P. Bechlioulis, M. V. Liarokapis, and K. J. Kyriakopoulos. Task specific robust grasping for multifingered robot hands. In *IEEE/RSJ International Conference on Robotics and Automation*, pages 858–863. IEEE, 2014.

[23] A. M. Bronstein and M. M. Bronstein. Spatially-sensitive affine-invariant image descriptors. In *2010 European Conference on Computer Vision (ECCV)*, volume 6312, pages 197–208. Springer, 2010.

[24] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision (IJCV)*, 78(1):67–88, Jun 2008.

[25] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics*, 30(1):1:1–1:20, February 2011.

[26] P. A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *Case-Based Reasoning Research and Development*, pages 80–95, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[27] M. Chen, K. Q. Weinberger, F. Sha, and Y. Bengio. Marginalized denoising auto-encoders for nonlinear representations. In Tony Jebara and Eric P. Xing, editors, *Proceedings of International Conference on Machine Learning (ICML)*, pages 1476–1484. JMLR Workshop and Conference Proceedings, 2014.

[28] X. Chen, X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29*, pages 2172–2180. Curran Associates, Inc., 2016.

[29] X. Chen, R. Mottaghi, X. Lu, S. Fidler, R. Urtasun, and A. L. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '14. IEEE Computer Society, 2014.

[30] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *2007 IEEE International Conference on Computer Vision (ICCV)*, 2007.

[31] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society.

[32] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *2011 International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011.

[33] D. J. Cook and L. B. Holder. *Mining Graph Data.* John Wiley & Sons, 2006.

[34] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[35] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, Oct 2017.

[36] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893 vol. 1, June 2005.

[37] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '13, pages 1814–1821, Washington, DC, USA, 2013. IEEE Computer Society.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[39] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[40] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn. Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[41] R. Detry, N. Pugeault, and J. Piater. A probabilistic framework for 3D visual object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(10):1790–1803, 2009.

[42] R. Diankov. *Automated Construction of Robotic Manipulation Programs.* PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[43] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4), 2012.

[44] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. de Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[45] C. Dune, E. Marchand, C. Collowet, and C. Leroux. Active rough shape estimation of unknown objects. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3622–3627, Sept 2008.

[46] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687, Sept 2015.

[47] B. Epshtein and S. Ullman. Semantic hierarchies for recognizing objects and parts. In *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[48] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJRR)*, 88(2):303–338, June 2010.

[49] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis.* Wiley Publishing, 4th edition, 2009.

[50] M. Fabre-Thorpe, G. Richard, and S. Thorpe. Rapid categorization of natural images by rhesus monkeys. 9:303–8, 01 1998.

[51] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cereb Cortex*, pages 1–47, 1991.

[52] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645, Sep 2010.

[53] P. F. Felzenszwalb, D. Mcallester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[54] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 264–271, 2003.

[55] C. Ferrari and J. Canny. Planning optimal grasps. In *1992 International Conference on Robotics and Automation (ICRA)*, pages 2290–2295. IEEE, 1992.

[56] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *European Conference on Computer Vision (ECCV)*, volume 3953 of *LNCS*, pages 14–28, June 2006.

[57] S. Fidler, M. Boben, and A. Leonardis. Learning hierarchical compositional representations of object structure. In *Object categorization computer and human perspectives*, pages 196–215. Cambridge University Press, Cambridge, UK, 2009.

[58] S. Fidler, M. Boben, and A. Leonardis. Learning a hierarchical compositional shape vocabulary for multi-class object representation. *CoRR*, abs/1408.5516, 2014.

[59] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '07, pages 1–8, June 2007.

[60] D. Fischinger and M. Vincze. Empty the basket – a shape based learning approach for grasping piles of unknown objects. In *International Conference on Intelligent Robots and Systems*, pages 2051–2057. IEEE/RSJ, 2012.

[61] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[62] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67–92, Jan 1973.

[63] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[64] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(11):2188–2202, November 2011.

[65] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[66] R. B. Girshick, F. N. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 437–446. IEEE Computer Society, 2015.

[67] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

[68] A. K. Goins, R. Carpenter, W.-K. Wong, and R. Balasubramanian. Evaluating the efficacy of grasp metrics for utilization in a gaussian process-based grasp predictor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3353–3360. IEEE/RSJ, 2014.

[69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[70] I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta. Three-finger precision grasp on incomplete 3d point clouds. In *IEEE/RSJ International Conference on Robotics and Automation*, pages 5366–5373. IEEE, 2014.

[71] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2016.

[72] G. Guo, Y. Wang, T. Jiang, A. L. Yuille, F. Fang, and W. Gao. A shape reconstructability measure of object part importance with applications to object detection and localization. *International Journal of Computer Vision (IJCV)*, 108(3):241–258, July 2014.

[73] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27 – 48, 2016. Recent Developments on Deep Big Vision.

[74] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *2014 European Conference on Computer Vision (ECCV)*, 2014.

[75] L. Haibin and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(2):286–299, Feb 2007.

[76] K. Hang, J. A. Stork, F. T. Pokorny, and D. Kragic. Combinatorial optimization for hierarchical contact-level grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 381–388. IEEE, 2014.

[77] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[78] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2014 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[79] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[80] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. T. Asfour, and S. Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1–2):51–65, 2014.

[81] U. Hillenbrand and M. A. Roa. Transferring functional grasps through contact warping and local replanning. In *IEEE/RSJ International Conference on Robotics and Systems*, pages 2963–2970. IEEE, 2012.

[82] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[83] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[84] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[85] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24, 1933.

[86] P. Hough. Method and Means for Recognizing Complex Patterns. U.S. Patent 3.069.654, December 1962.

[87] K. Hsiao and T. Lozano-Perez. Imitation learning of whole-body grasps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5657–5662. IEEE, 2006.

[88] X. Hu, J. Zhang, J. Li, and B. Zhang. Sparsity-regularized hmax for visual recognition. *PLOS ONE*, 9(1):1–12, 01 2014.

[89] E. Hyttinen, D. Kragic, and R. Detry. Learning the tactile signatures of prototypical object parts for robust part-based grasping of novel objects. In *IEEE International Conference on Robotics and Automation*, pages 4927–4932. IEEE, 2015.

[90] M. Inaba, S. Kagami, F. Kanehiro, Y. Hoshino, and H. Inoue. A platform for robotics research based on the remote-brained robot approach. *The International Journal of Robotics Research*, 19(10):933–954, 2000.

[91] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *2017 Conference on Robot Learning (CoRL)*, 2017.

[92] M. Jamieson, Y. Eskin, A. Fazly, S. Stevenson, and S. J. Dickinson. Discovering hierarchical object models from captioned images. *Computer Vision and Image Understanding (CVIU)*, 116(7):842–853, 2012.

[93] S. Jayasumana, R. Hartley, M. Salzmann, L. Hongdong, and M. Harandi. Optimizing over radial kernels on compact manifolds. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3802–3809, June 2014.

[94] S. Jayasumana, M. Salzmann, H. Li, and M. Harandi. A framework for shape analysis via hilbert space embedding. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 1249–1256, 2013.

[95] E. Johns, S. Leutenegger, and A. J. Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4461–4468. IEEE, 2016.

[96] D. Kanoulas, J. Lee, D. G. Kanoulas, and N. G. Tsagarakis. Visual grasp affordance localization in point clouds using curved contact patches. *International Journal of Humanoid Robotics*, 14(1):1–21, 2017.

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

[98] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):664–676, April 2017.

[99] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on Automation Science and Engineering*, 12(2):398–409, April 2015.

[100] J. Kim, K. Iwamoto, J. J. Kuffner, Y. Ota, and N. S. Pollard. Physically based grasp quality evaluation under pose uncertainty. *IEEE Transactions on Robotics*, 29(6):1424 – 1439, 2013.

[101] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib. Grasping with application to an autonomous checkout robot. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2837–2844. IEEE, 2011.

[102] E. Kobatake and K. Tanaka. Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex. *Journal of Neurophsychiology*, pages 856 – 867, 1994.

[103] I. Kokkinos and A. L. Yuille. Inference and learning with hierarchical shape models. *International Journal of Computer Vision (IJCV)*, 93(2):201–225, 2011.

[104] G. W. Kootstra, M. Popovi, J. A. Jørgensen, K. Kuklinski, K. Miatliuk, D. Kragic, N. Krüger, et al. Enabling grasping of unknown objects through a synergistic use of edge and surface information. *The International Journal of Robotics Research (IJRR)*, 34:26–42, 2012.

[105] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt. One-shot learning and generation of dexterous grasps for novel objects. *The International Journal of Robotics Research (IJRR)*, 35(8):959–976, 2016.

[106] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[107] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105. Curran Associates, Inc., 2012.

[108] O. Kroemer, E. Ugur, E. Oztop, and J. Peters. A kernel-based approach to direct action perception. In *2012 IEEE International Conference on Robotics and Automation*, 2012.

[109] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1847–1871, Aug 2013.

[110] J. Kuffner. Cloud-enabled humanoid robots. *2010 10th IEEE-RAS International Conference on Humanois Robotics*, 2010.

[111] S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 769–776, Sept 2017.

[112] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of International Conference on Machine Learning (ICML)*, ICML '07, pages 473–480, New York, NY, USA, 2007. ACM.

[113] L. J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 424–429, Jun 2000.

[114] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[115] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989.

[116] H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area v2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *International Conference on Neural Information Processing Systems (NIPS)*, pages 873–880. Curran Associates, Inc., 2008.

[117] A. Lehmann, B. Leibe, and L. J. Van Gool. Fast prism: Branch and bound hough transform for object class detection. *International Journal of Computer Vision (IJCV)*, 94(2):175–197, 2011.

[118] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[119] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research (IJRR)*, 34(4-5):705–724, 2015.

[120] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *International Journal of Robotics Research (IJRR)*, 34(4–5):705–724, 2015.

[121] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, page 0278364917710318, 2017.

[122] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, volume 1 of *Springer Proceedings in Advanced Robotics*, pages 173–184. Springer, 2016.

[123] N. Levitt. The science of conjecture: Evidence and probability before pascal. *The Mathematical Intelligencer*, 26(1):53–55, Dec 2004.

[124] H. Li, K. Zhang, and T. Jiang. Minimum entropy clustering and applications to gene expression analysis. In *2004 IEEE Computational Systems Bioinformatics Conference*, pages 142–151, Aug 2004.

[125] L. J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2036–2043, June 2009.

[126] B. Liefeng, K. Lai, R. Xiaofeng, and D. Fox. Object recognition with hierarchical kernel descriptors. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '11, pages 1729–1736, Washington, DC, USA, 2011. IEEE Computer Society.

[127] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *2002 International Conference on Image Processing*, volume 1, pages I–900–I–903 vol.1, 2002.

[128] K.-L. Lim and H. K. Galoogahi. Shape classification using local and global features. In *2010 Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, pages 115–120, Washington, DC, USA, 2010. IEEE Computer Society.

[129] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[130] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence C. L. Zitnick. Microsoft coco: Common objects in context. In *2014 European Conference on Computer Vision (ECCV)*, pages 740–755, Cham, 2014. Springer International Publishing.

[131] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(2):286–299, Feb 2007.

[132] Z. H. Ling, S. Y. Kang, H. Zen, A. Senior, M. Schuster, X. J. Qian, H. M. Meng, and L. Deng. Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 32(3):35–52, May 2015.

[133] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani. Visual grasp planning for unknown objects using a multifingered robotic hand. *IEEE/ASME Transactions on Mechatronics*, 18(3):1050–1059, June 2013.

[134] S. Liu and S. Carpin. Global grasp planning using triangular meshes. In *IEEE International Conference on Robotics and Automation*, pages 4904–4910. IEEE, 2015.

[135] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11 – 26, 2017.

[136] Y.-H. Liu. Computing n-finger form-closure grasps on polygonal objects. *The International Journal of Robotics Research (IJRR)*, 19(2):149–158, 2000.

[137] N. K. Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5(5):552 – 563, 1995.

[138] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[139] D. G. Lowe. Towards a computational model for object recognition in it cortex. In *Biologically Motivated Computer Vision*, pages 20–31, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[140] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *International Symposium on Robotics Research*, 2017.

[141] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.

[142] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

[143] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning. *arXiv preprint arXiv:1709.06670*, 2017.

[144] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1038–1045. IEEE Computer Society, 2009.

[145] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[146] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *16th IEEE International Conference on Image Processing*, ICIP'09, pages 3465–3468, Piscataway, NJ, USA, 2009. IEEE Press.

[147] Z. C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz. General 3d modelling of novel objects from a single view. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3700–3705, Oct 2010.

[148] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[149] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[150] L. Nanni, S. Brahnam, and A. Lumini. Local phase quantization descriptor for improving shape retrieval/classification. *Pattern Recognition Letters*, 33(16):2254–2260, Dec 2012.

[151] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[152] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning (ICML)*, pages 689–696. Omnipress, 2011.

[153] V. D. Nguyen. Constructing force-closure grasps. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1368–1373, Apr 1986.

[154] P. Noriega and O. Bernier. Multicues 3d monocular upper body tracking using constrained belief propagation. In *2007 British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2007.

[155] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *International Conference on Pattern Recognition (ICPR)*, volume 1, pages 582–585 vol.1, Oct 1994.

[156] B. Ommer and J. Malik. Multi-scale object detection by clustering lines. In *2009 International Conference on Computer Vision (ICCV)*, pages 484–491, 2009.

[157] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, États-Unis, November 2013.

[158] M. Ozay, U. R. Aktas, J. L. Wyatt, and A. Leonardis. Compositional hierarchical representation of shape manifolds for classification of non-manifold shapes. In *2015 IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[159] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[160] D.I. Perrett and M.W. Oram. Neurophysiology of shape processing. *Image and Vision Computing*, 11(6):317 – 333, 1993.

[161] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE International Conference on Robotics and Automation*, pages 3406–3412. IEEE, 2016.

[162] R. Platt, R. A. Grupen, and A. H. Fagg. Learning grasp context distinctions that generalize. In *IEEE-RAS International Conference on Humanoid Robots*, pages 504–511. IEEE, 2006.

[163] N. S. Pollard. Closure and quality equivalence for efficient synthesis of grasps from examples. *The International Journal of Robotics Research (IJRR)*, 23(6):595–613, 2004.

[164] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 11(6):868–881, 1995.

[165] M. Popović, D. Kraft, L. Bodenhagen, E. Başeski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58(5):551–565, 2010.

[166] S. Qi, S. Huang, P. Wei, and S.-C. Zhu. Predicting human activities using stochastic grammar. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1173–1181, 2017.

[167] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *International Conference on Neural Information Processing Systems (NIPS)*, NIPS'06, pages 1137–1144, Cambridge, MA, USA, 2006. MIT Press.

[168] D. Raviv, Ron R. Kimmel, and A. M. Bruckstein. Graph isomorphisms and automorphisms via spectral signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1985–1993, Aug 2013.

[169] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[170] C. Redondo-Cabrera, R. Lopez-Sastre, and T. Tuytelaars. All together now: Simultaneous detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. In *2014 British Machine Vision Conference (BMVC)*. BMVA Press, 2014.

[171] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99. Curran Associates, Inc., 2015.

[172] D. Reynolds. *Gaussian Mixture Models*, pages 827–832. Springer US, Boston, MA, 2015.

[173] M. Richtsfeld and M. Zillich. Grasping unknown objects based on 2.5d range data. In *IEEE International Conference on Automation Science and Engineering*, pages 691–696. IEEE, 2008.

[174] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

[175] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, September 1978.

[176] M. Roa and R. Suarez. Grasp quality measures: Review and performance. *IEEE Robotics and Automation Letters*, 38(1):65–88, 2015.

[177] A. Rodriguez, M. T Mason, and S. Ferry. From caging to grasping. *The International Journal of Robotics Research (IJRR)*, 31(7):886–900, 2012.

[178] C. Rosales, A. Ajoudani, M. Gabiccini, and A. Bicchi. Active gathering of frictional properties from objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3982–3987, September 2014.

[179] C. Rosales, R. Suárez, M. Gabiccini, and Bicchi. On the synthesis of feasible and prehensile robotic grasps. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 550–556. IEEE, 2012.

[180] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[181] G. A. Rousselet, M. Fabre-Thorpe, and S. Thorpe. Parallel processing in high-level categorization of natural images. 5:629–30, 08 2002.

[182] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[183] R. B. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz. Perception for mobile manipulation and grasping using active stereo. In *IEEE-RAS International Conference on Humanoids*, pages 632–638. IEEE, 2009.

[184] F. Sadeghi and S. Levine. (cad)\$ˆ2\$rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016.

[185] R. R. Salakhutdinov and G. E. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research (PMLR)*, pages 412–419, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

[186] R. R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[187] J. Saut and D. Sidobre. Efficient models for grasp planning with a multi-fingered hand. *Robotics and Autonomous Systems*, 60(3):347–357, 2012.

[188] P.-A. Savalle, S. Tsogkas, G. Papandreou, and I. Kokkinos. Deformable part models with cnn features. In *European Conference on Computer Vision, Parts and Attributes Workshop*, Zurich, Switzerland, September 2014.

[189] A. Saxena, L. S. Wong, and A. Y. Ng. Learning grasp strategies with partial shape information. In *2008 Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1491–1494, 2008.

[190] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[191] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision (IJCV)*, 56(3):151–177, Feb 2004.

[192] D. Seita., F. T. Pokorny, J. Mahler, D. Kragic, M. Franklin, J. Canny, and K. Goldberg. Large-scale supervised learning of the grasp robustness of surface patch pairs. In *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 216–223. IEEE, 2016.

[193] J. Seo, S. Kim, and V. Kumar. Planar, bimanual, whole-arm grasping. In *2012 IEEE International Conference on Robotics and Automation*, pages 3271–3277, May 2012.

[194] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(3):411–426, March 2007.

[195] A. Shapiro, E. Rimon, and J. W. Burdick. On the mechanics of natural compliance in frictional contacts and its effect on grasp stiffness and stability. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1264–1269. IEEE, 2004.

[196] L. Shen, S. Wang, G. Sun, S. Jiang, and Q. Huang. Multi-level discriminative dictionary learning towards hierarchical visual categorization. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 383–390. IEEE, 2013.

[197] K. B. Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research (IJRR)*, 15(3):230–266, 1996.

[198] H.-C. Shin, H. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. J. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35:1285–1298, 2016.

[199] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S. W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(7):1125–1140, Jul 2005.

[200] Z. Si and S.-C. Zhu. Learning and-or templates for object recognition and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(9):2189–2205, 2013.

[201] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision (IJCV)*, 35(1):13–32, Nov 1999.

[202] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[203] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1470–1477, October 2003.

[204] R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[205] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *26th International Conference on Machine Learning (ICML)*, 2011.

[206] D. Song, C. Henrik Ek, K. Huebner, and D. Kragic. Task-based robot grasp planning using probabilistic inference. *IEEE Transactions on Robotics*, 31(3):546–561, 2015.

[207] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[208] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision (ICJV)*, 77(1-3):291–330, May 2008.

[209] K. B. Sun and B. J. Super. Classification of contour shapes using class segment sets. In *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '05, pages 727–733. IEEE Computer Society, 2005.

[210] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4278–4284, 2017.

[211] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[212] D. Tabernik, M. Kristan, and A. Leonardis. Spatially-adaptive filter units for deep neural networks. *CoRR*, abs/1711.11473, 2017.

[213] D. Tabernik, M. Kristan, J. L. Wyatt, and A. Leonardis. Towards deep compositional networks. In *International Conference on Pattern Recognition (ICPR)*, pages 3470–3475, Dec 2016.

[214] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3d object detection and pose estimation. In *2014 European Conference on Computer Vision (ECCV)*, pages 462–477, Cham, 2014. Springer International Publishing.

[215] A. ten Pas and R. Platt. Localizing handle-like grasp affordances in 3d point clouds. In *International Symposium on Experimental Robotics*, 2014.

[216] J. Tobin, R. Fong, A. K. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[217] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

[218] M. Trobina and A. Leonardis. Grasping arbitrarily shaped 3-d objects from a pile. In *1995 IEEE International Conference on Robotics and Automation (ICRA)*, pages 241–246. IEEE, 1995.

[219] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[220] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, January 1976.

[221] L. G. Ungerleider and J.V. Haxby. What and where in the human brain. *Current Opinion in Neurobiology*, 4:157–165, 1994.

[222] L.J.P van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9: 2579–2605, Nov 2008.

[223] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders. Kernel codebooks for scene categorization. In *2008 European Conference on Computer Vision (ECCV)*, ECCV '08, pages 696–709, Berlin, Heidelberg, 2008. Springer-Verlag.

[224] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[225] J. Varley, J. Weisz, J. Weiss, and P. Allen. Generating multi-fingered robotic grasps via deep learning. In *Inteligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4415–4420. IEEE, 2015.

[226] M. Veres, M. Moussa, and G. W. Taylor. Modeling grasp motor imagery through deep conditional generative models. *IEEE Robotics and Automation Letters*, 2(2):757–764, 2017.

[227] U. Viereck, A. ten Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using easily simulated depth images. *CoRR*, abs/1706.04652, 2017.

[228] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *2008 International Conference on Machine Learning (ICML)*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM.

[229] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511–I–518 vol.1, 2001.

[230] J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. In *2009 British Machine Vision Conference (BMVC)*, pages 1–11, 2009.

[231] Z. Wang, Z. Li, B. Wang, and H. Liu. Robot grasp detection using multimodal deep convolutional neural networks. *Advances in Mechanical Engineering*, 8(9):1–12, 2016.

[232] J. Weisz and P. K. Allen. Pose error robust grasping from contact wrench space metrics. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 557–562, May 2012.

[233] T. Wiatowski and H. Bölcskei. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, 64(3):1845–1866, March 2018.

[234] T. Wu, B. Li, and Zhu S.-C. Learning and-or model to represent context and occlusion for car detection and viewpoint estimation. *IEEE Transactions on Pattern Analysis andMachine Intelligence (TPAMI)*, 38(9):1829–1843, 2016.

[235] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang,

C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[236] Z. Xu, H. Chen, S.-C. Zhu, and J. Luo. A hierarchical compositional model for face representation and sketching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(6):955–969, June 2008.

[237] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[238] K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[239] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *2014 European Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 818–833, Cham, 2014. Springer International Publishing.

[240] H.-Z. Zhang, Y.-F. Lu, T.-K. Kang, and M.-T. Lim. B-hmax: A fast binary biologically inspired model for object recognition. *Neurocomputing*, 218:242 – 250, 2016.

[241] L. Zhang and J. C. Trinkle. The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing. In *2012 IEEE International Conference on Robotics and Automation*, pages 3805–3812, May 2012.

[242] Q. Zhang, Y. N. Wu, and S. C. Zhu. Mining and-or graphs for graph matching and object discovery. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 55–63, Dec 2015.

[243] Y. Zheng and W.-H. Qian. Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research (IJRR)*, 24(4):311–327, 2005.

[244] Y. Zhou and K. Hauser. 6dof grasp planning by optimizing a deep learning scoring function. In *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact-Turning a Problem into a Solution*, 2017.

[245] L. Zhu, Y. Chen, and A. L. Yuille. Learning a hierarchical deformable template for rapid deformable object parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(6):1029–1043, 2010.

[246] L. Zhu, Y. Chen, and A. L. Yuille. Recursive compositional models for vision: Description and review of recent work. *Journal of Mathematical Imaging and Vision*, 41(1-2):122–146, Sep 2011.

[247] S.-C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, January 2006.