3D COMPOSITIONAL HIERARCHIES FOR OBJECT CATEGORIZATION

by

VLADISLAV VADIMOVICH KRAMAREV

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
April 2017

ABSTRACT

Modern deep learning methods have de-facto become the default tool for image classification. However, application of deep learning to surface shape classification is burdened by the limitations of the existing methods, in particular, by the lack of invariance to different transformations of input data.

This thesis proposes two novel frameworks for learning a multi-layer representation of surface shape features, namely the view-based and the surface-based compositional hierarchical frameworks. The representation learned by these frameworks is a hierarchical vocabulary of surface shape features, termed parts. Parts of the first layer are pre-defined, while parts of the subsequent layers, describing spatial relations of subparts, are learned from the training data. Both frameworks exploit the learning algorithms which comprise clustering of statistical maps representing the statistics of co-occurrences of parts in the training data.

The view-based framework describes spatial relations between subparts in the camera-based reference frame. The key stage of the learning algorithm on this framework is part selection which forms the vocabulary based on multi-objective optimization, taking into account different importance measures of parts. The experiments show that this framework enables efficient category recognition on a large-scale dataset of depth images.

The main idea of the surface-based framework is the exploitation of part-based intrinsic reference frames, which are computed for parts of lower layers and inherited by parts of the subsequent layers. During learning spatial relations between subparts are described in these reference frames. During inference a part is detected in the input data when its congruent subparts are found at certain positions and orientations in each other's reference

frames, regardless of their positions and orientations in the camera-based reference frame. Since rigid body transformations do not change positions and orientations of parts in intrinsic reference frames, this approach enables efficient recognition of shapes from unseen poses. Experiments with the surface-based frameworks show that it exhibits a large discriminative power and achieves a greater robustness to rigid body transformations than advanced CNN-based methods.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Jeremy L. Wyatt. Certainly, without his support this work would never be possible. I am grateful for very productive discussions with him in which many ideas presented in this thesis were born. Jeremy was very helpful both when discussing the conceptual problems and the low-level problems, for example, details of the algorithms, the equations, experimental settings, and many other questions. I owe him much for his invaluable support both in my research and in personal matters. Finally, I want to acknowledge that on the last stage of this work Jeremy helped me by reading several chapters of this thesis and providing very useful comments.

I owe much to my parents who have been constantly supporting me in all my endeavors. I would like to say thanks to my brother Dr. Dmitry Kramarev, who gave me many useful advice, and supported me in different matters. In the last phase of this work Dmitry also helped me as a third party editor; he read several chapters of this thesis and gave me very useful advice about possible corrections.

I am very grateful to my friends and colleagues Umit Rusen Aktas, Dr. Mirela Carmia Popa, Dr. Mete Ozay, Dr. Krzysztof Walas and Dr. Dominik Belter for very interesting and productive discussions with them. It was my great pleasure to work with them. Also, I want to say thanks to Dr. Sebastian Zurek for useful discussions and for the other help he provided.

I acknowledge the contribution of Prof. Leonardis, especially in the first two years of this work when we had many constructive discussions. Many ideas behind the view-based hierarchy, presented in this thesis, appeared during my discussions with him. I also acknowledge the advice he has been giving me on the related literature, as well as other

general comments and advice.

My special thanks go to Prof. Justus Piater. Unfortunately, we had only very few discussions with Justus, however, I want to mention that he was the first person to confidently support my ideas behind the surface-based hierarchy, described in this thesis, and his support gave me the confidence I needed to proceed working in the direction I have chosen.

CONTENTS

Nomenclature **x**

List of Figures **xiv**

List of Tables **xvii**

1 Introduction **1**

1.1 Background 1

1.1.1 Challenges of Object Categorization 1

1.1.2 Category Representation 2

1.1.3 Categorization Methods Based on Artificial Neural Networks 4

1.1.4 Deep Learning Methods for 3D Shape Analysis 6

1.1.5 Compositional Hierarchical Systems 9

1.2 Motivation 14

1.3 Research Goals of this Thesis 15

1.4 Hypotheses, Challenges and Contributions 16

1.5 Definitions and Notations 18

1.5.1 Invariance 21

1.6 Structure of the Thesis 24

2 Literature Review: Representation of 2D images **25**

2.1 Handcrafted Image Features 25

2.1.1 Local Image Descriptors 26

2.1.2 Global Image Descriptors and Texture Descriptors 28
2.1.3 Limitations of the Handcrafted Image Features 29
2.2 Deep Learning in Computer Vision 30
2.2.1 Biological Motivation 30
2.2.2 Artificial Neural Networks 32
2.3 Compositional Hierarchical Architectures 35
2.3.1 Theoretical Work on Hierarchical Compositionality 36
2.3.2 And-Or Trees 38
2.3.3 Existing Hierarchical Compositional Systems 39
2.3.4 Experimental Evaluation of Compositional Hierarchical Systems . . 40

3 Literature Review: Representation of 3D Shape 43
3.1 Handcrafted Shape Features 44
3.1.1 Local Shape Descriptors 46
3.1.2 Global Shape Descriptors 51
3.2 ANN-based Methods for 3D Shape Analysis 53
3.2.1 Mapping from Manifolds to the 2D Euclidean Domain 54
3.2.2 ANNs in the 3D Euclidean Domain 56
3.2.3 CNNs based methods on manifolds and other methods 58
3.2.4 Compositional Hierarchical Methods for 3D Data 59

4 Description of the Approach 60
4.1 Design Principles 61
4.1.1 Separate Representation for Surface Shapes 61
4.1.2 Properties of parts 62
4.1.3 Redundancy of the Representation 63
4.2 Learning and Inference Algorithms 63
4.2.1 Learning and Inference Algorithms for Multiple Layers 64
4.2.2 Vocabulary Learning Pipeline for a Single Layer 66

5 A View-Based Compositional Hierarchy **69**

5.1 Introduction . . . 69

5.1.1 Camera-Based Reference Frame . . . 70

5.1.2 Description of the Input Data . . . 71

5.2 Pre-Processing of the Input Data . . . 72

5.3 First Layer . . . 74

5.3.1 Form of the First Layer Parts . . . 74

5.3.2 Quantization of Surface Orientations . . . 75

5.3.3 Inference of the First Layer . . . 76

5.4 Composition Rules . . . 78

5.4.1 Empty subparts . . . 81

5.5 Vocabulary Learning . . . 82

5.5.1 Collecting Co-Occurrence Statistics . . . 83

5.5.2 Clustering of Statistical Maps and Parameterization of Clusters . . . 84

5.5.3 Inference of Doublets from the Training Data . . . 86

5.5.4 Co-activations of Doublets . . . 87

5.6 Part Selection Problem . . . 90

5.6.1 Distance Between Parts . . . 93

5.6.2 Importance Measures of Candidate Parts . . . 94

5.6.3 Part Selection as a Multi-Objective Optimization Problem . . . 97

5.6.4 Divide and Conquer Strategy for Part Selection . . . 99

5.7 Inference of a Single Layer . . . 103

5.8 Pooling . . . 105

5.9 Object Categorization from Range Images . . . 106

5.9.1 Histogram of Compositional Parts . . . 106

5.9.2 Classification of HoCP by SVM . . . 108

5.10 Evaluation . . . 112

5.10.1 Computational Resources and Timing . . . 116

5.10.2 Absence of OR-Nodes . . . 117
5.11 Conclusion . . . 117

6 A Surface-Based Compositional Hierarchy 119

6.1 Design Principles . . . 121
6.1.1 Part-Based Reference Frames . . . 121
6.1.2 Learning and Inference from Triangulated Mesh Models . . . 122
6.2 Pre-Processing of the Input Data . . . 124
6.3 First Layer . . . 126
6.3.1 Inference of the First Layer . . . 127
6.3.2 Defining a Geodesic Circle . . . 127
6.4 Composition Rules . . . 129
6.4.1 Local Reference Frame . . . 131
6.5 Vocabulary Learning . . . 133
6.5.1 Collecting of Co-occurrence Statistics . . . 133
6.5.2 Transformation from Global to Local Reference Frame . . . 134
6.6 Clustering of the Statistical Maps . . . 139
6.6.1 Agglomerative Hierarchical Clustering . . . 140
6.6.2 Input Data for Clustering . . . 141
6.6.3 Pairwise Distance Between Entries . . . 142
6.6.4 Optimal Number of Clusters . . . 143
6.6.5 Parameterization of Clusters . . . 144
6.7 Vocabulary Learning (continuation) . . . 146
6.7.1 Inference of Doublets from the Training Data . . . 146
6.7.2 Co-activations of Doublets . . . 147
6.7.3 Part Selection (Grouping by OR-Nodes) . . . 149
6.8 Inference of a Single Layer . . . 150
6.9 Pooling . . . 152
6.10 Evaluation . . . 152

6.10.1 Evaluation Methodology 153
6.10.2 DataSet 154
6.10.3 Methods for Comparison 155
6.10.4 Evaluation Results 157
6.11 Conclusion 162

7 Conclusion and Future Work **166**
7.1 Conclusions 166
7.2 Summary 167
7.3 Future Work 169

List of References **171**

NOMENCLATURE

Δ^{sp}	Spatial relations between subparts of a vocabulary part
ϵ_g	Contrast parameter of the guided image filter
$\mathcal{B}$	A set of composition (binding) rules
$\mathcal{C}(L_n)$	Set of candidate parts of the layer L_n
$\mathcal{D}(L_n)$	Set of doublets of the layer L_n
$\mathcal{F}$	Receptive field
$\mathcal{L}$	Linkage function
$\mathcal{L}_a$	Average linkage
$\mathcal{L}_c$	Complete linkage
$\mathcal{L}_s$	Single linkage
$\mathcal{M}_{i,j}^n$	Statistical map showing co-occurrences of parts P_i^n and P_j^n
$\mathcal{P}$	Point cloud
$\mathcal{R}$	Frame of reference
$\mathcal{S}(L_n)$	Tuple of parts (shape vocabulary) of the layer L_n
$\mathcal{T}$	Dataset
$\mathcal{T}(i)$	i-th element of the dataset

$\mathcal{T}_p$	Pre-processed dataset
$\mathcal{T}_p(i)$	i-th element of the pre-processed dataset
$\mathcal{T}_{\mathcal{S}(L_n)}$	Dataset $\mathcal{T}$ expressed in terms of the shape vocabulary $\mathcal{S}(L_n)$
$\mathcal{X}_i$	i-th cluster in a statistical map
$\lvert \cdot \rvert$	Cardinality a set
μ	Centroid of the multivariate Gaussian distribution
Ω	Image domain
π	Doublet, i.e. a compositional part comprising two subparts
ρ_i	Atomic data point i , e.g. a point in a point cloud
Σ	Covariance matrix
σ_i	Average distance from cluster elements to the centroid c_i
A	Matrix describing cross-bin similarities of a histogram
B_i	Bin i of a statistical map
c_i	Centroid of the i-th cluster in a statistical map
C_i^n	Candidate part i of the layer L_n
$Cover(C_i^n \mid \mathcal{S}(L_n))$	Conditional coverage of a candidate part C_i^n, given vocabulary $\mathcal{S}(L_n)$
$Cover(C_i^n)$	Coverage of a candidate part C_i^n
D	Pairwise distance matrix
d_E	Euclidian distance
d_R	Rotational distance

d_{χ^2}	χ^2 histogram distance
d_{EMD}	Earth Mover's distance
d_{ER}	Weighted sum of the Euclidian and rotational distances
d_M	Mahalanobis distance
d_{QC}	Quadratic-Chi histogram distance
d_v	Volumetric distance, used as similarity measure between parts
F	Set of faces of the triangular mesh
f_i	Face i of the triangular mesh
FoV_h	Field of View (horizontal)
FoV_v	Field of View (vertical)
H_e	Histogram entropy
$k(T_i)$	Directional surface curvature in the direction of a tangent vector T_i
k_1, k_2	Principal curvatures
K_σ	2-dimensional Gaussian kernel with the parameter σ
L_n	Layer n of the compositional hierarchy
M_ρ	Tensor of curvature
M_{RT}	Transformation matrix
M_R	Rotation matrix
M_T	Translation matrix
N	Surface normal

P_i^n	Part i of the layer L_n
q	Quaternion
R	Part realization
r_g	Parameter of the guided image filter (radius)
T_1, T_2	Directions of the principal curvature
T_i	Vector i tangent to a surface
V	Set of vertices of the triangular mesh
v_i	Vertex i of the triangular mesh

LIST OF FIGURES

1.1 Pipeline of many approaches to surface shape analysis using deep learning 8

1.2 Example of a compositional part 10

1.3 Example of OR-nodes 12

1.4 Azimuth, elevation and distance define position of camera relative to an object's gravity centre 22

1.5 Views of the object captured with different a) camera elevations b) azimuths c) in-plane rotations 23

1.6 Illustration for view invariance 23

3.1 Illustration for the intrinsic and extrinsic surface properties 46

4.1 Scheme of the learning and inference algorithms 65

5.1 Several part of different layers of the view-based hierarchy 70

5.2 Camera based frame of reference (black) and image frame of reference (blue) 71

5.3 Washington RGB-D dataset 72

5.4 Quantization of surface normals 76

5.5 Quantization of surface normals to 1-d bins 77

5.6 illustration of composition rules 79

5.7 Parts of the layer L_2 learned under composition rules $\mathcal{B}$. Empty cells are shown with red crosses 81

5.8 Parts of the layer L_3 learned under composition rules $\mathcal{B}$. Empty cells are shown with red dots 81

5.9 Example illustrating usage of empty cells 82
5.10 Parts with empty subparts typically appear in the locations where the surface ends of occludes itself 83
5.11 (a) Parameterized statistical map $\mathcal{M}^1_{41,41}(x, y, z)$ depicting co-occurrences of realizations of the part P^1_{41} (b) Doublets formed from this statistical map 85
5.12 Illustration for different co-activation cases 90
5.13 Distance between two candidate parts 93
5.14 Histogram for candidate parts with: a) high entropy, b) moderate entropy, c) low entropy. 96
5.15 Explanation for coverage and conditional coverage of candidate parts . . . 97
5.16 Histogram of Compositional Parts 107
5.17 Illustration of the Earth Mover's Distance measure 110
5.18 Beeswarm plots illustrating performance of different settings 114
5.19 Object categorization results using different kernels 115
5.20 Object categorization results using parts of different layers 116

6.1 Examples of parts of different layers of the surface-based hierarchy 120
6.2 Pre-processing of the triangulated mesh model 126
6.3 Illustration for geodesic circles 129
6.4 Comparison of the composition rules of the view-based and the surface-based compositional hierarchies 130
6.5 Computation of the curvature tensor using surface points and surface normals 132
6.6 Transformation from the global to local reference frame 136
6.7 Illustration for clustering of statistical a map 143
6.8 Results of clustering of the statistical maps 145
6.9 Examples of deformable object categories. 155
6.10 Depth image colorized by a) surface normals, b) Principal curvatures . . . 157
6.11 Illustration for the radial HoCP 158

6.12 Beeswarm plot illustrating the accuracy of the view-based system on the new dataset. 160
6.13 Performance of different methods under changes of view 162
6.14 Relative performance of different methods under changes of view. Recognition performance of each method from the front view is assumed to be 1.0 163
6.15 Performance of different methods under in-plane rotations 164
6.16 Relative performance of different methods under in-plane rotations. Recognition result of each method without in-plane rotation is assumed to be 1.0 165

LIST OF TABLES

5.1 Results of different methods on Washington RGB-D dataset (depth channel) (*) - Methods pre-trained on ImageNet. 113

6.1 Results achieved by different methods on the new dataset. Leave-one-out cross validation is used. 158

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
AOT	And-OR Tree
BoW	Bag of words
CNN	Convolutional Neural Network
DBI	Davies – Bouldin index
DBN	Deep Belief Network
DCM	Direction Cosine Matrix
DoG	Difference of Gaussian
EMD	Earth Mover's distance
FMA	Fast Marching Algorithm
FoV	Field of View
GDM	Geodesic Distance Matrix
GLOH	Gradient Location and Orientation Histogram
HeatSD	Heat Shape Descriptor
HHA	Horizontal disparity, Height above ground, Angle with gravity
HKS	Heat Kernel Signature
HoCP	Histogram of Compositional Parts
HOG	Histogram of Oriented Gradients
HONV	Histogram of Oriented Normal Vectors
ILSVRC	Imagenet large scale visual recognition challenge
ISC	Intrinsic Shape Context

LBP	Local Binary Patterns
LRF	Local Reference Frame
MDL	Minimal Description Length
MLE	Maximum Likelihood Estimation
PacMan	**P**robabilistic **a**nd **c**ompositional Representation for object **man**ipulation
PCA	Principal Component Analysis
PSD	Positive Semi-Definite
RBM	Restricted Boltzmann Machine
RGB-D	Red-Green-Blue-Depth (4 dimensional image)
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SHREC	Shape Retrieval Contest
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
TD	Temperature Distributor
WKS	Wave Kernel Signature
w.r.t.	with respect to

CHAPTER 1

INTRODUCTION

1.1 Background

One of the problems which have been intensively studied by the computer vision community over a long time is *object categorization*, in which objects in images are grouped into categories for some specific purpose. A very large number of object categorization methods using different visual aspects of objects, such as size, shape, color, texture, and motion are proposed in the literature.

1.1.1 Challenges of Object Categorization

Object categorization is a very difficult task due to several reasons. The first reason is that usually there is large intra-category variability, i.e. objects of the same category may exhibit dramatically different geometric and photometric properties. Intra-category variability of shape is particularly large for non-rigid (deformable) objects, for example, objects with articulated parts. On the other hand, inter-category variability may be very small, since objects of different categories may have very similar shape and appearance, making them almost indistinguishable from certain viewpoints.

The second reason is that every single object may appear in images in an astronomically large number of ways due to a large range of its possible relative positions and

orientations relative to a camera. The most difficult scenario is object categorization under the *generic viewpoint assumption* [1], according to which objects are not assumed to be in a certain position and orientation relative to a camera.

The third reason is that the number of existing objects and object categories is large. There exist different estimates which vary depending on the estimation methodology. For instance, according to Biederman's [2] estimation, there exist approximately ∼30,000 geometrically distinguishable objects which can be classified into approximately ∼3,000 categories.

The fourth reason is that there exists a number of difficulties related to the imaging process. Camera settings and illumination conditions, under which objects are captured, may vary significantly; moreover, all image acquisition methods impose different image degradations, such as noise, image blurring, etc. Additional challenges come from the fact that in real world cluttered scenes objects may partially occlude each other.

However, despite all these challenges, human vision systems deal with the massively complicated problem of object categorization seemingly effortlessly, accommodating a very large number of object categories and performing very fast [3] and robust category recognition. In contrast, for the computer vision community object categorization is still an open research topic attracting considerable attention, as evidenced by a large number of papers dedicated to this topic appearing annually in the mainstream journals and conferences.

1.1.2 Category Representation

There are two fundamental questions which have to be addressed when designing a computer vision system for object categorization. The first question relates to a type of *representation*, i.e. how category models should be efficiently represented in memory. There are several desired properties of a category representation, for instance, its generalization capabilities, geometric and photometric invariance, scalability, moderate storage demands, ability to facilitate fast inference and matching, etc. Generalization is the prop-

erty of a category representation meaning that description of categories should be general enough to enable correct category recognition for novel objects, i.e. those objects that were not used when developing the representation. Scalability means the ability of an object categorization system to handle a growing amount of data, and its sublinear growth when the number of categories increases. An *invariance* is a property of a computer vision system meaning that the output of this system remains unchanged or changes insignificantly under certain transformations of the input data, e.g. illumination changes, viewpoint changes, shape deformations, etc. It is hard to develop a representation having all these properties simultaneously, however, for some specific task settings some of these properties may become more important than the other ones.

The second fundamental question is how the representation should be acquired. In some of the recent computer vision papers (e.g. [4, 5, 6]) different approaches for developing a category representation are broadly classified into two large classes. The first class of approaches assumes category modelling based on *handcrafted* image or shape features, for instance, histograms, shape signatures, filter banks, co-occurrence matrices, etc. Approaches of the second class performs *multi-layer learning* (also termed *deep learning*) of image and shape features, and use these learned features for category modeling[1].

A huge amount of handcrafted features exploiting different visual aspects of objects, such as 2D contour shape, 3D surface shape, colour, texture, and motion, have been proposed in the literature. SIFT [7], HOG [8], spin images [9] and HKS [10] are examples of very popular hand-crafted features. Some of the handcrafted image features, for instance, SIFT, and some of its modifications (e.g. PCA-SIFT [11], GLOH [12]) have become particularly widely used, since they combine a large discriminative power and an invariance to different types of geometric and photometric transformations of the input data, such as in-plane rotations, changes of scale, illumination changes and (up to a certain extent) out-plane rotations [7, 12]. Another example is HKS which is the popularly employed handcrafted surface shape descriptor invariant to rigid body transformations and

[1]Note, that such a splitting is not very strict anyways, as there exist approaches that to some extent involve both handcrafting and multi-layer learning. More details will be discussed in Chapters 2 and 3

to isometric deformations of shapes. Such a combination of useful properties makes these handcrafted features applicable for a large range of computer vision and shape analysis tasks, including object categorization, detection, tracking, image referencing, shape retrieval, etc. The object categorization systems based on handcrafted image features have been dominant over a long period, however, their progress substantially slowed down in 2010-2012 [13]. The introduction of the large scale category recognition challenges, for instance, the annual ILSVRC[1] challenge [14] based on ImageNet [15], demonstrated that it is very difficult or impossible to achieve further rapid progress purely with hand-crafted image features.

Another common way of acquiring category representations, which has become particularly popular in the last few years, assumes multi-layer learning (also termed *deep learning*) of image or shape features from a training set. The key idea behind deep learning is the existence of multiple abstraction layers, where features of higher layers are defined in terms of features of the previous layers, thus enabling the description of category models using multiple levels of features of increasing complexity. Broadly speaking, the paradigm of deep learning suggests that a complicated object or category model should be described in terms of the less complicated models, which, in turn, can be represented in terms of even simpler models, and so on, while features at the lowest abstraction layer should encode primitive image features (e.g. edge segments or textons [16]). Thereby, the existence of multiple abstraction layers helps to bridge a large semantic gap between primitive image features and complicated object or category models, by recursively expressing different mid-level visual cues at multiple hierarchical levels.

1.1.3 Categorization Methods Based on Artificial Neural Networks

Most of the recent advances in the area of deep learning methods for object categorization deal with artificial neural networks (ANNs), for instance, convolutional neural networks

[1]Imagenet Large Scale Visual Recognition Challenge

(CNNs), deep belief networks (DBNs), autoencoders, etc. CNNs are the most widely used in computer vision type of ANNs, since this architecture is particularly well-adapted for feature learning from the image domain, and for classification from images. Each CNN contains a number of *convolutional layers* each of which comprises a set of feature maps that respond to certain structures in input images. A feature map can be considered as a stencil which is being convolved with an input image. Convolutional layers are usually interspersed with *pooling layers* purposed to simplify the information in the output of convolutional layers. Pooling can be described as a form of non-linear downsampling, e.g. by replacing 3×3 node regions with a single node on the next layer. The purpose of pooling layers is to progressively reduce the spatial resolution of the representation by reducing the number of parameters and computations in higher layers of a CNN. Nodes of convolutional and pooling layers take input values only from a certain local neighbourhood of a square form, termed *receptive field*, so that these nodes respond only to local image structures, located within the receptive field. In contrast, higher layers of CNNs are *fully connected*, i.e. they take inputs from the whole image domain, thus responding to certain spatial configurations of features in the whole image. Thereby CNNs combine the power of local and global feature representations.

The discriminative power of modern ANN-based methods has been thoroughly demonstrated in the literature. In fact, these methods outperform the methods based on handcrafted image features by a large margin [17], especially for large-scale category recognition [15]. However, despite the outstanding performance, CNN-based architectures have certain limitations. For instance, they are not very robust to different types of transformations such as viewpoint changes and, especially, in-plane rotations. That means the recognition ability of CNN-base methods goes down significantly when testing objects are orientated (w.r.t. a camera) differently than objects from the training set. For instance, one of the experiments, described in Chapter 6 of this thesis, shows that the CNN of Krizhevsky *et al.* [17] working with depth images achieves 81.2% category recognition accuracy on a certain dataset in the scenario when testing and training objects are

approximately aligned. However, if the testing objects are rotated in plane only by 30 degrees (thus becoming unaligned with the training objects), the recognition accuracy of the system falls to 29.3%, while if the testing objects are rotated by 60 degrees, the recognition accuracy goes down to 13.7%[1].

There are several fundamental reasons for such a behavior, one of the most important of which is that the convolution operations, which are applied on multiple layers of CNNs, are in general case not rotationally-invariant[2]. This is why objects rotated by a certain angle start producing dramatically different responses to feature maps, thus activating different features on multiple hierarchical layers. The commonly taken way of addressing this issue is *augmenting* the training dataset by rotating the training images/models multiple times; thus increasing the training set by at least one or two orders of magnitude which is very computationally expensive for large datasets. Effectively, this means that each shape and appearance feature is learned at multiple orientations, and the ability of CNN-based systems to recognize an object at a certain orientation depends on the presence of similar objects under similar orientation in the training set[3].

1.1.4 Deep Learning Methods for 3D Shape Analysis

The recent advent of reliable and cheap depth sensors, such as Microsoft Kinect and PrimeSence Carmine, and a rapid growth of 3D shape models repositories (e.g. Google Warehouse) increased the interest in 3D shape analysis problems, such as object categorization based on surface shape features, 3D shape retrieval, etc. However, a much smaller number of ANN-based methods working with 3D data (e.g. depth images, triangulated mesh models or point clouds) have been proposed in the literature, compared to

[1]Note that these numbers may vary significantly for different CNN architectures and different datasets. Also note that there exist some techniques (e.g. *colorization* of the depth channel), mitigating this problem and making CNNs slightly more robust to different geometric transformations. More details will be presented in Chapter 6

[2]Except for the case when filter kernels are chosen to be rotationally invariant, e.g. the Gaussian kernels

[3]Note, that the similar problems appear also in other types of ANNs, for instance in convolutional deep belief networks, which also require augmenting the training set by rotating objects multiple times [18].

the number of ANN-based methods for 2D data (i.e. RGB or grayscale images).

Apparently, the reason for this is that CNNs, which are the main type of ANNs used in computer vision, are specially designed to learn features from the 2D image domain, i.e. a rectangular or square area in the 2D Euclidean space, regularly sampled by a pixel grid. Consequently, it is a very natural idea to apply CNN-based systems for classification from RGB or grayscale images, i.e. the type of data such systems are designed for. On the other hand, the direct application of CNNs to different surface shape analysis problems is not straightforward since surfaces typically represent non-Euclidean manifolds, on which there is no shift invariance, and therefore convolution does not exist in a classical sense [19, 20, 21].

Due to this problem, most of the deep learning approaches to 3D shape analysis problems use different types of mapping of surface shapes to the 2D Euclidean domain, i.e. onto the image plane, and apply CNNs after that, as illustrated in Figure 1.1. Some authors [22] use computer graphics techniques to render images from 3D models, and apply the CNN for classifying these rendered images. Sinha *et al.* [21] use parameterization of 3D surface models to produce multi-channel images of a special type, called *geometry images*, and apply the CNN to classify these images, while Shi *et al.* [23] proposed to use the projective geometry to render panoramic images, and apply the CNN classifier to them. Many authors, e.g. [24, 25]), use depth images, representing partial views of 3D shape models, and process these depth images by CNNs in the same ways as if they were ordinary grayscale images. However, the disadvantage of such an approach is that rich surface geometry information can not be efficiently utilized in this case. Consequently, some authors (e.g. [13, 26]) propose the methods involving *colorization* of depth images, i.e. converting them to multi-channel images, where channels encode different geometric characteristics of a surface (e.g. surface normals, horizontal disparity, etc.) and then apply CNNs to classify colorized images. In general, the CNN-based approaches working with colorized depth images achieve substantially better results than the approaches using raw depth images.

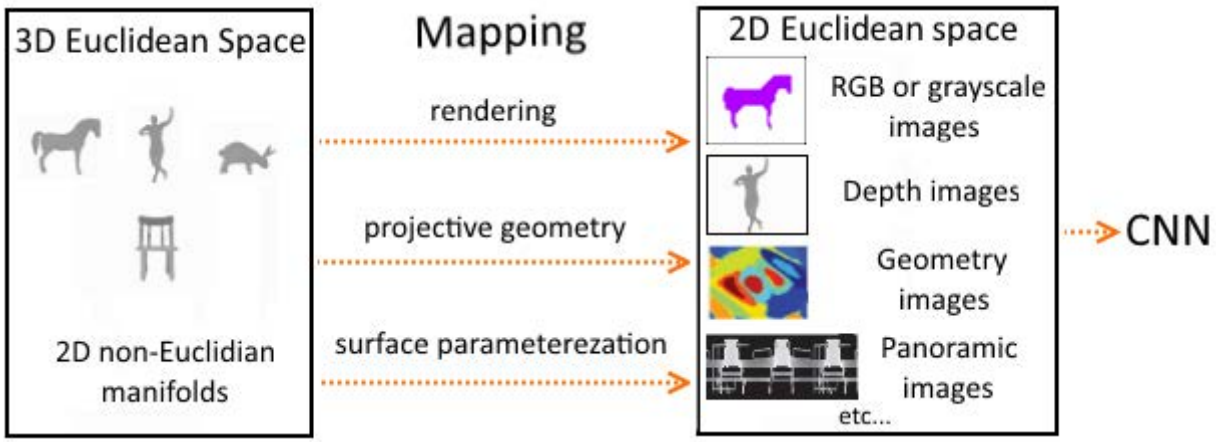


Figure 1.1: Pipeline of many approaches to surface shape analysis using deep learning

Broadly speaking, all presented approaches do not apply deep learning in the data domain, i.e. on manifolds embedded in Euclidean 3D space, but rather map the input data to the 2D Euclidean domain, where CNNs can be efficiently used, and apply CNNs in this domain. Apparently, there are several disadvantages of such approaches. First, any of these mappings induce certain errors. Second, since CNNs are specially designed for the 2D domain, they are not based on any special design principles that would facilitate efficient exploitation of very rich surface geometry information. This is evidenced by the fact that the colorization techniques, by only incorporating very simple geometric features to the CNN input substantially improve the categorization results [26]. This may be an indication that CNNs cannot automatically efficiently learn such features from raw depth data and, presumably, the colorization techniques only mitigate, but not fully solve this problem. Third, due to a non rotationally invariant nature of CNNs (discussed in Subsection 1.1.3) any system exploiting them assumes that each input shape model should be presented in the training set at multiple orientations, otherwise it will be very likely to fail at recognizing objects from radically novel viewing angles and/or in-plane rotations.

There are several recently proposed frameworks [18, 4] for 3D shape analysis, which perform voxelization of 3D shapes, computing 3D occupancy grids, and then apply 3D convolutional neural networks or 3D convolutional deep belief networks in the 3D Eu-

clidean domain. However, these approaches have severe limitations, such as very low spatial resolution of voxel grids (e.g. $30 \times 30 \times 30$ voxels in [18]), and lack of an invariance to different types of data transformations, such as rigid body transformations and isometric deformations of non-rigid objects [21]. There are very few attempts [20, 27] to adapt the CNNs to learn features in the data domain, i.e. directly from surfaces, but these methods are limited since they can only learn local shape features, therefore they can not utilize the power of CNNs to automatically learn hierarchical features of increasing size.

1.1.5 Compositional Hierarchical Systems

Except for the methods based on ANNs, there are other types of multi-layer frameworks proposed in the computer vision literature. For instance, there exists a class of deep learning method referred as *compositional hierarchies*. Many compositional hierarchical methods [28, 29, 30, 31, 32, 33, 34] appeared between 2005-2010, i.e. when the methods based on handcrafted features were still dominantly employed, while ANNs were not yet demonstrating outstanding results. This situation encouraged the researchers to investigate different types of multi-layer frameworks that are not based on ANNs. Note, that some of the compositional hierarchical methods were further developed between 2013-2016 [35, 36, 37, 38, 39, 40], i.e. in the period of dominance of ANN-based methods.

Broadly speaking, most of the compositional hierarchical frameworks are based on similar principles which are sometimes referred as principles of *hierarchical compositionality*. There are theoretical papers proposing a mathematical formulation of hierarchical compositionality, and a formal description of these principles [28, 41]. However, in this introductory chapter I give only a high-level explanation of the main principles of hierarchical compositionality by demonstrating some relevant examples. The main reason for such a choice is that most of these ideas are formulated and implemented quite differently by the different authors.

The hierarchical compositionality assumes the existence of very simple and easily detectable image features at the first layer. Features of all the subsequent layers represent

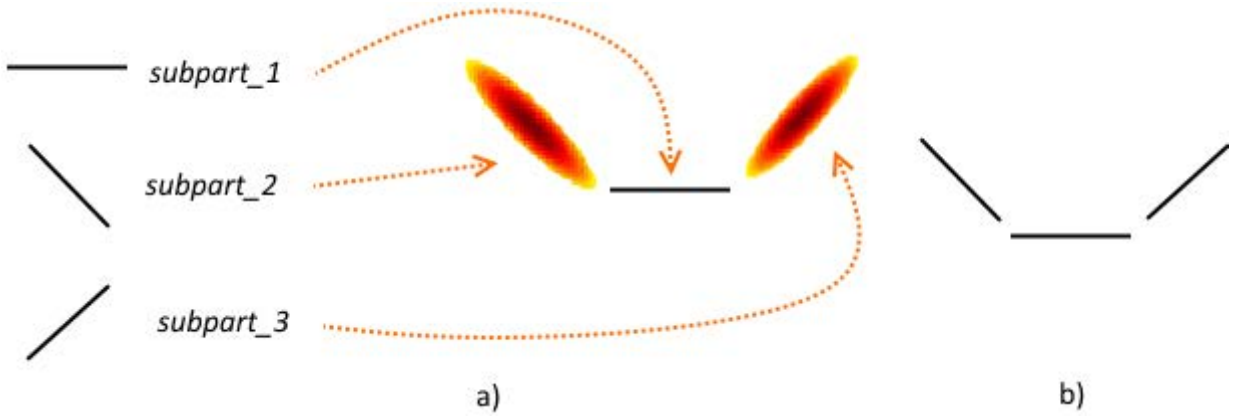


Figure 1.2: An example of a compositional part, comprising three subparts. a) Three subparts are composed into a compositional part. Distributions of possible relative positions of subparts are parameterized by Gaussian distributions. b) Resulting compositional part.

compositions or *compositional parts*, meaning that each of them is **composed** of two or more subparts, i.e. features from the previous layer. The most important characteristic of a compositional part is the description of relative positions of its subparts. This description is represented by a distribution (rather than by a single offset vector) making compositional parts non-rigid, i.e. tolerating some variance in relative positions of subparts. An example illustrating how a compositional part may be composed of three subparts is shown in Figure 1.2.

One of the most important principles of hierarchical compositionality is *reusability* (also termed *part-sharing* or *shareability*) of compositional parts, meaning that the same compositional part may be used to describe objects of different categories. In fact, many compositional parts represent some shape or appearance features that are not specific to a particular object category and therefore, they can be used to describe multiple object categories. Part reusability is a very important property since it makes the representation more compact, facilitates scalability of the representation, and speeds up the inference process by reducing the number of matching operations [42].

Another principle of hierarchical compositionality is termed *executive-summary* [41]. This and/or similar principles are differently formulated by different authors, however, the meaning is that a state of each compositional parts is only described coarsely (as a

short summary) while a more detailed description is kept in the states of its subparts. For instance, a state of a high-level part, representing a human category model may be a short summary of the following type: "Subpart (e.g. representing a body), **and** subpart (head) in a certain relative position[1] from the body, **and** two subparts (arms) in certain relative positions from the body, **and** two subparts (legs) in certain relative positions from the body". According to the executive-summary principle, such a non-detailed description is sufficient for each compositional part across all layers, since more details can be extracted from the states of subparts. For instance, the state of the subpart representing a human head may involve a similarly structured description, for example: "Subpart (e.g. representing a nose), **and** two subparts (eyes) at certain relative positions from the nose, **and** subpart (mouth) at a certain relative position from the nose, etc." In turn, the states of each of these subparts contain similarly structured descriptions. At the second layer of the hierarchy compositional parts are also described in a similar manner, but they are composed of the primitive parts of the first layer.

The next principle of hierarchical compositionality is the existence of certain switching variables (often referred as *OR-nodes*) on different hierarchical layers, showing the alternative configurations. For example, a human body may be quite differently shaped, arms may be straight or bent, a mouth can be open or closed, etc. Obviously, since compositional parts are non-rigid, certain shape variations may be represented by a single compositional part. However, in some cases, shape variations may be too large to be modeled by a single compositional part. This means that a state of a compositional part should also include an indication, that certain subparts may be represented by several different parts from the previous layer. Taking this into account, the description of a state of a compositional part representing a human model may look as follows: "$\Big[$subpart (e.g. representing slim body) **OR** subpart (fit body) **OR** subpart (another body type)$\Big]$, **and** $\Big[$subpart (bald head) **OR** subpart (head with a hat)$\Big]$ at a certain relative position from the body, **and**, etc.". Figure 1.3 shows an example how alternative subparts representing

[1]Remember that relative positions of subparts are described by distributions, not by single vectors

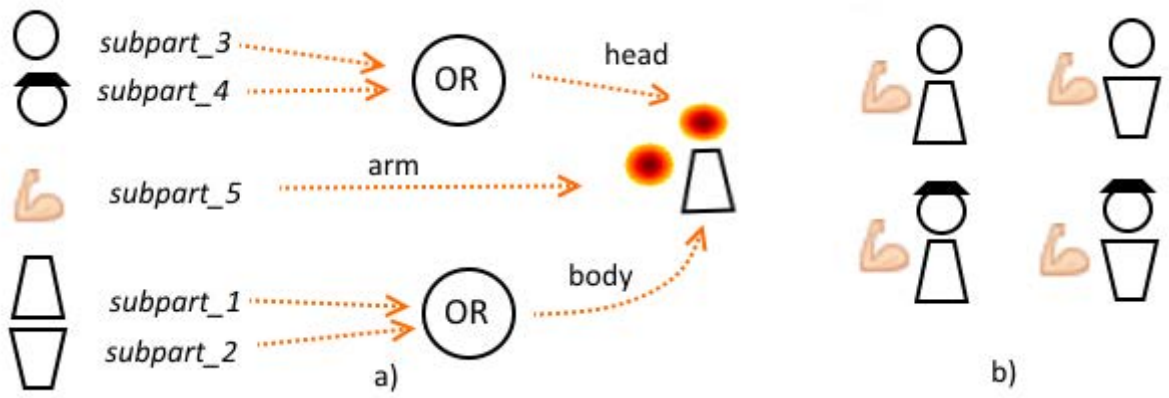


Figure 1.3: An example illustrating how OR-nodes may used to encode alternative configurations of subparts. a) There are two subparts representing different body models, and two subparts representing different head models. b) Resulting compositional parts.

different body models and different head models are composed into a single compositional part of the next layer via OR-nodes.

In general, a non-rigid nature of compositional parts, the executive summary principle and the existence of switching variables are purposed to increase the generalization capability of compositional hierarchical systems, especially when dealing with object categories having a large shape and appearance variability, for instance, categories of non-rigid objects with articulated subparts.

Right after presenting these examples, which I used to illustrate some of the principles, it is very important to make clear that actually a part-to-subpart decomposition in the existing compositional hierarchical systems does not necessarily coincide with semantic parts of objects. In contrast, such a decomposition may be done by a compositional hierarchical system very differently from how humans would mentally do it [2]. For instance, in real compositional hierarchical systems the part-to-subpart decomposition of a human model may be done very differently from the examples I just presented.

In fact, there exists a very large number of possible ways how to recursively decompose a complicated shape to subparts, as well as a very large number of possible ways of composing simpler shape features to more complicated ones[1]. Therefore, in order to re-

[1]Note that different compositional hierarchical systems go both ways either by recursively decomposing complex parts into simpler subparts, or starting with primitive features learn more and more complex compositions. More details will be presented in Chapter 2

duce this ambiguity, each compositional hierarchical system has a certain set of *grammar rules*, also termed *composition rules*, or *binding rules* dictating how exactly the compositions/decompositions should be performed.

So far we considered five principles of hierarchical compositionality which are: (i) the non-rigid nature of compositional parts, (ii) part reusability, (iii) the executive-summary, (iv) the existence of switching variables, and (v) the existence of composition rules. However, it is important to mention, that despite a certain conceptual similarity of the existing compositional hierarchical frameworks, they formulate and implement the aforementioned principles in very different ways. These frameworks are designed for different purposes such as object categorization, recognition of letters and digits, image parsing and/or segmentation face representation, multi-view object detection, pose estimation, etc. They use different types of primitive features at the lowest level, different composition/decomposition rules and different ways of describing spatial relations between subparts. These frameworks also substantially differ on the level of the mathematical formulation, and on the level of implementation, data structures, etc.[1]

Note that most of the compositional hierarchical systems are designed for 2D images and there exist only very few systems working with 3D data (e.g. [43]). However, these systems work only with a limited number of shapes (e.g. cars, hand-made objects) and, according to the best of my knowledge, there are no compositional hierarchical methods for 3D data that are sufficiently well generalizable to be applied to datasets comprising large numbers of object categories. Also note that the problem of the geometric invariance of compositional hierarchical systems attracted relatively little attention so far. Only a few authors [44, 45, 39] addressed the problem of rotational invariance of compositional hierarchical systems, however, these systems have mainly been applied to specific types of data, e.g. images of handwritten digits.[2].

The rest of the introductory chapter is organized as follows. Section 1.2 describes

[1]Also, according to the best of my knowledge, there is no software, such as libraries, toolboxes, etc. commonly used across multiple compositional hierarchical systems, therefore, compositional hierarchical systems are typically developed from scratch independently from each other.

[2]More details will be provided in Chapter 2

the motivation of the research work presented in this thesis. Section 1.3 specifies the research goals of this thesis. Section 1.4 describes the main hypotheses, challenges, and contributions of this thesis, while Section 1.5 presents the definitions and the notations which are used in the subsequent chapters of the thesis. Finally, Section 1.6 outlines the structure of the remainder of the thesis.

1.2 Motivation

We can make several important conclusions from the previous section. Regarding neural networks the following conclusions can be made. First, despite the outstanding performance of CNNs, which are the main type of ANNs used in computer vision, they cannot extrapolate their description of geometric relationships of features to radically novel viewpoints and, in particular, to novel in-plane rotations. Second, there exists a much smaller number of ANN-based methods for 3D data, than for 2D data, since there are certain problems in applying CNNs in other domains than in the 2D Euclidean domain. Third, the ANN-based methods specially designed for these domains have strong limitations, for instance, low spatial resolution, lack of invariance to different data transformations, etc.

Several conclusions can also be made about compositional hierarchical systems. First, these systems are currently not at the level of performance of ANN-based methods, however, they have some interesting theoretical properties reported by different authors. Second, the principles of hierarchical compositionality are quite flexible, and can be easily complemented by some additional principles. Finally, we notice that most of the compositional hierarchical systems have been designed for the 2D domain and there have been only few attempts to develop 3D compositional hierarchical systems, however, these system have severe limitations.

Many researchers, e.g. Hinton [46], Biederman [2] and others, have pointed out that human visual systems are very good at mental extrapolations, such that after seeing a new shape once people can recognize it from a different viewpoint and/or in-plane rotation.

The necessity of an object and category representation which is as invariant as possible to geometric transformations is advocated in the classical computer vision literature [2, 47]. These facts suggest that the deep learning area may potentially largely benefit from introducing special principles that would facilitate robustness of deep learning methods to different types of data transformations.

Since there are only few deep learning methods specially designed for 3D data and these methods have severe limitations, it is necessary to develop a novel deep learning system which would be robust to transformations of different types, in particular, to in-plane rotations and view changes. On the other hand, since convolution operations, which are applied on multiple layers of CNNs are not rotationally-invariant by definition, we assume, that such a deep learning method should be based on another mathematical ground. Thus arise two theoretically and practically interesting questions i.e. (i) whether it is possible to develop a deep learning architecture working with 3D data that would be substantially more robust (compared to CNN-based systems) to in-plane rotations and view changes, and (ii) on which theoretical principles such an architecture can be developed.

We assume that the principles of hierarchical compositionality provide such an opportunity. However, since there are only very few compositional hierarchical methods for 3D data (also having strong limitations), it is necessary first to transfer the ideas of hierarchical compositionality to the 3D domain, and develop a deep learning framework for surface shape features based on these principles. After that it is necessary to propose some new principles, that would facilitate the geometric invariance of this system, and to modify the framework according to these principles.

1.3 Research Goals of this Thesis

There are two main research goals of this thesis.

The **first goal** is to transfer the existing principles of hierarchical compositionality

to the 3D domain, by developing a novel framework for multi-layer learning of surface shape features based on these principles. The focus of this work should be in the domain change from 2D to 3D, therefore, we do not aim to introduce novel principles, as long as they are not needed for the domain change. The developed framework should be applied to object categorization on the Washington RGB-D dataset [48], on which many modern deep learning methods have been tested, and thereby compared to other methods.

The **second goal** is to revise and complement the principles of hierarchical compositionality by introducing special principles facilitating robustness to view changes and in-plane rotations. Then another framework for multi-layer learning of surface shape features should be developed on the basis of the revised principles. Our target is to develop a framework having a large discriminative power and facilitating greater robustness to view changes and in-plane rotations than advanced CNN-based methods. To confirm these properties a novel framework should be applied to object categorization in two scenarios, namely (i) when training and testing objects are aligned, and (ii) when recognition from novel (unseen) views and in-plane rotations is performed. A comparison with advanced CNN-based approaches in these experimental settings should be performed.

1.4 Hypotheses, Challenges and Contributions

There are two **main hypotheses of this thesis**.

- The principles of hierarchical compositionality can be used as a basis for a novel deep learning framework capable of learning surface shape features which would enable efficient category recognition on a large dataset of depth images comprising multiple object categories.
- The principles of hierarchical compositionality, after being revised and complemented, may serve as a basis for a deep learning framework capable of learning surface shape features demonstrating a large discriminative power and a greater robustness to view changes and in-plane rotations than CNN-based approaches.

There are two **main challenges** of this work.

- The first challenge is that the properties of 2D data, for which most of the existing compositional hierarchical systems were developed, are radically different from the properties of 3D data, therefore, a significant work has to be done for adaptation of the principles of hierarchical compositionality to the 3D domain. Consequently, most of the ingredients of a 3D compositional hierarchical system (e.g. composition rules, part selection methods, etc.) should be defined differently than in 2D compositional hierarchical systems.

- The second challenge is the need to develop a novel complicated deep learning system from scratch, due to absence of the special software (e.g. libraries, tooboxes) implementing the standard functionality of compositional hierarchical systems. Note that the competing ANN-based methods are to a large extent based on the existing mature software, for instance, Caffe [49], Torch [50], Theano [51], providing a very rich functionality for artificial neural networks and supporting massively parallel computations on GPUs.

Major contributions of this PhD thesis include:

- Development of the novel deep learning framework for surface shape features based on the principles of hierarchical compositionality. At the time when this system was finished (by the end of 2014), its result on the Washington dataset [48] (80.4%) was very close to the state-of-the-art level (which was around 81%). Later, however, the performance of this system has been surpassed by more recent ANN-based approaches, which achieved substantially better results (e.g. [26, 52]) on this dataset.

- Development of the novel deep learning framework for surface shape features, based on the principles of hierarchical compositionality, and on the novel principle of hierarchical part-based reference frames. Surface shape features learned by this framework demonstrate a large discriminative power, and substantially greater robustness

to in-plane rotations compared to the advanced ANN-based systems, and slightly greater robustness to large viewpoint changes (around 45-60 degrees).

Minor contributions of this PhD thesis include:

- The thesis proposes a novel protocol for evaluating the robustness of computer vision methods to view changes and in-plane rotations. According to this protocol training and testing objects are first aligned, and then the camera position is gradually changed to produce novel views and in-plane rotations. The methods are then compared in category recognition from the unseen views.

- The novel dataset comprising 20 categories of rigid and 52 categories of deformable objects is proposed. Objects in the dataset are manually aligned, and the depth images representing the front view of each objects are rendered.

- The thesis proposes to use the advanced affinity measures for histograms (namely the EMD and Quadratic-chi) within SVM for classification of histograms of compositional parts. In particular, we demonstrated how to build a cross-bin similarity matrix using the measure of geometric similarity of compositional parts.

1.5 Definitions and Notations

In this section I explain the terms and the notation used in the following chapters of this thesis. Note, that some of the details vary for both the compositional hierarchies proposed in this thesis, i.e. the view based and the surface based ones. That is why in this section a general description of the terms is provided, while more details for each hierarchy are specified (if necessary) in the corresponding chapters.

The representation of surface shape features proposed in this thesis is called a *compositional hierarchical shape vocabulary*. This representation contains several *layers* (denoted L_n, where $(n \geqslant 1)$), each of which represents a tuple[1] of *parts*, i.e. surface shape models.

[1] finite ordered list of elements

A tuple of a layer L_n is termed *shape vocabulary* of the layer L_n and denoted $\mathcal{S}(L_n)$. The notation $\mid \mathcal{S}(L_n) \mid$ is used to denote number of parts in the vocabulary of the layer L_n.

Parts from the vocabulary of the first layer L_1 represent pre-defined shape features. These parts are atomic, i.e. they are not assumed to be composed of any simpler elements. In this work I made a choice to use planar surface patches as parts of the layer L_1, however, in general, there is no obstacle for choosing other types of atomic parts. In contrast to the atomic parts of the layer L_1, parts of each subsequent layer $L_n, (n \geqslant 2)$ are *compositions of subparts* (i.e. parts of the previous layer L_{n-1}) statistically learned from the training data. Note, that in this thesis I will use the term *parts* for both atomic parts of the first layer and compositional parts of the subsequent layers. I use the notation P_i^n to denote i-th part of the layer L_n.

Spatial configuration Δ^{sp} is a parameterized distribution of possible relative positions and (for the surface based compositional hierarchy only) relative orientations of a subpart w.r.t. another subpart of a compositional part. Note, that the actual type of parametrization is chosen differently for both the compositional hierarchies presented in this thesis, that is why more details will be provided in the corresponding chapters.

Assume, we have k parts in the vocabulary of the layer $L_n, (n \geqslant 1)$, i.e., $\mid \mathcal{S}(L_n) \mid = k$. Also assume we learned t spatial configurations of L_n parts from the training data, and we denote the l-th spatial configuration as Δ_l^{sp}, where $1 \leq l \leq t$. Then a *doublet* π_i^{n+1} of the layer L_{n+1} $(n \geqslant 1)$ is defined as follows:

$$\pi_i^{n+1} = \left(P_c^n, (P_j^n, \Delta_l^{sp})\right), \quad 1 \leq j \leq k,\ 1 \leq l \leq t,\ 1 \leq c \leq k \tag{1.1}$$

i.e., doublet is a compositional part, comprising two subparts, namely the *central subpart* P_c^n and another subpart P_j^n accompanied by the description of its relative position[1] Δ_l^{sp} w.r.t. the central subpart P_c^n.

After defining a doublet, we can formally define a compositional part, comprising any number of subparts. Part P_i^{n+1} of the layer L_{n+1} may be defined using the following *part*

[1] and relative orientation, in case of a surface based compositional hierarchy

equation:

$$P_i^{n+1} = \left(P_c^n,\, (P_{j_1}^n, \Delta_{l_1}^{sp}),\, (P_{j_2}^n, \Delta_{l_2}^{sp}),\, ...\, , (P_{j_m}^n, \Delta_{l_m}^{sp})\right), \quad 1 \leq c \leq k, \tag{1.2}$$

where m is the number of subparts, while all indices j are natural numbers from the interval $1 \leq j \leq k$, all indices l are natural numbers from the interval $1 \leq l \leq t$. This equation means that each part P_i^{n+1} comprises the central subpart P_c^n and m other subparts P_j^n with descriptions Δ_l^{sp} of their relative positions with respect to the central subpart.

A *part realization*, or *part activation* (denoted R) is an instance of a vocabulary part found in the input data, e.g. in a range image. Intuitively, it can be illustrated using the following example. Assume there is a segment (centered in the point ρ) of a range image. If a surface patch represented by this segment approximately fits a shape model, represented by a vocabulary part P_i^n, then we say that there is a realization (or activation) R_k of the part P_i^n in the point ρ, or, in other words, that part P_i^n is *activated* in the point ρ. This segment of the range image is then termed *support region* for the part realization R_k.

Learning of a layer $L_n, (n \geqslant 2)$ is a process of filling the vocabulary $\mathcal{S}(L_n)$ of this layer by compositional parts representing the most statistically representative configurations of parts of the previous layer L_{n-1}. *Inference* is a process of matching of the vocabulary parts against the input data, i.e. inference can be understood as a search of parts realizations in the input data. Both learning and inference procedures for a layer $L_n, (n \geqslant 2)$ are performed within a local neighborhood (termed *receptive field*, $\mathcal{F}$) around each part realization of the previous layer L_{n-1}. Receptive fields at higher layers of the hierarchy span larger surface areas, than at lower layers.

Composition rules (denoted $\mathcal{B}$), also called *binding rules*, is the set of syntactic rules under which subparts are composed into parts of the next layer. Composition rules should address the following questions: (Q1) which parts are allowed and not allowed be composed into parts of the next layer, (Q2) what is the minimal and maximal number of

subparts a compositional part can contain, (Q3) what is the range of possible spatial relations between subparts allowing composing them into a part of the next layer. Composition rules may either remain the same for all vocabulary parts at all layers of the compositional hierarchy, or be changed depending on the layer of the hierarchy and the type of shape represented by a vocabulary part.

1.5.1 Invariance

An *invariance* is a property of a system (or a function) meaning that the output of this system (or a function) remains unchanged or changes insignificantly under transformations of certain types applied to the input data. The term invariance may be used to describe a computer vision system, meaning that the output of this system, for example, the predicted object category labels, remain the same or almost the same under different transformations of the input data. This term can also be used to characterize a shape or appearance descriptor, i.e. a function that typically outputs a multi-dimensional vector describing certain properties of a shape or an image. An invariance of a descriptor means that the resulting multi-dimensional vector remains unchanged under certain transformations of the input data.

Several types of an invariance are considered in the computer vision literature, for instance, *photometric invariance*, i.e. invariance w.r.t. changes of lighting conditions, brightness or contrast changes, etc., and *geometric invariance*, for example, invariance to view changes, rotational or scale invariance. To explain different types of geometric invariance let us assume that the input objects are in fixed positions, i.e. they are not moved and not rotated, while the camera is changing its position with respect to the object such that camera's optical axis is always pointing to the object centre of mass. In this case the camera position can be specified by three parameters: *distance* from the camera to the object centre of mass, *azimuth* and *elevation*, as shown in Figure 1.4. We say that camera's azimuth, and elevation define a *view*. For each particular view the camera can also be *rotated* around its optical axis resulting in *in-plane rotations* of the

objects in the images. Examples of images captured under different camera elevations, azimuths and rotations are shown in Figure 1.5.

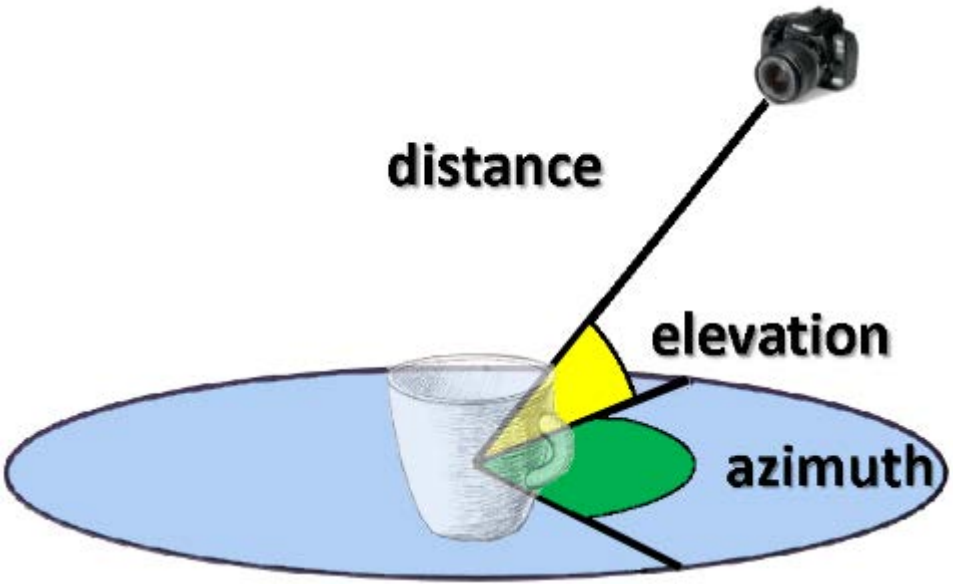


Figure 1.4: Azimuth, elevation and distance define position of camera relative to an object's gravity centre

The *rotational invariance* of a system (or function) means that its output remains the same under in-plane rotations in range from 0 to 360°. Similarly, *scale invariance* means the output remains unchanged under changes of the distance from the camera to the object, resulting in changes of the objects sizes in images. *Translational invariance* means that the output remains the same when objects are shifted (translated) in the plane parallel to the image plane[1].

Generally, the view invariance of a vision system means that its output remains unchanged or changes insignificantly under changes of view, i.e. changes of azimuth and elevation. It is, however, a bit more tricky to define view invariance of shape or appearance descriptors, since certain view changes affect visibility of their support regions [2]. What is actually quite often meant by a view invariance of a descriptor is that its output (i.e. a multidimensional vector) remains unchanged or changes insignificantly under

[1]Provided that all these transformations do not substantially change visibility of input objects, e.g. when objects are partially or fully moved outside the camera frame, or outside the range of a depth sensor, etc.

[2]Support region of a descriptor is the segment of the input data using which this descriptor is computed

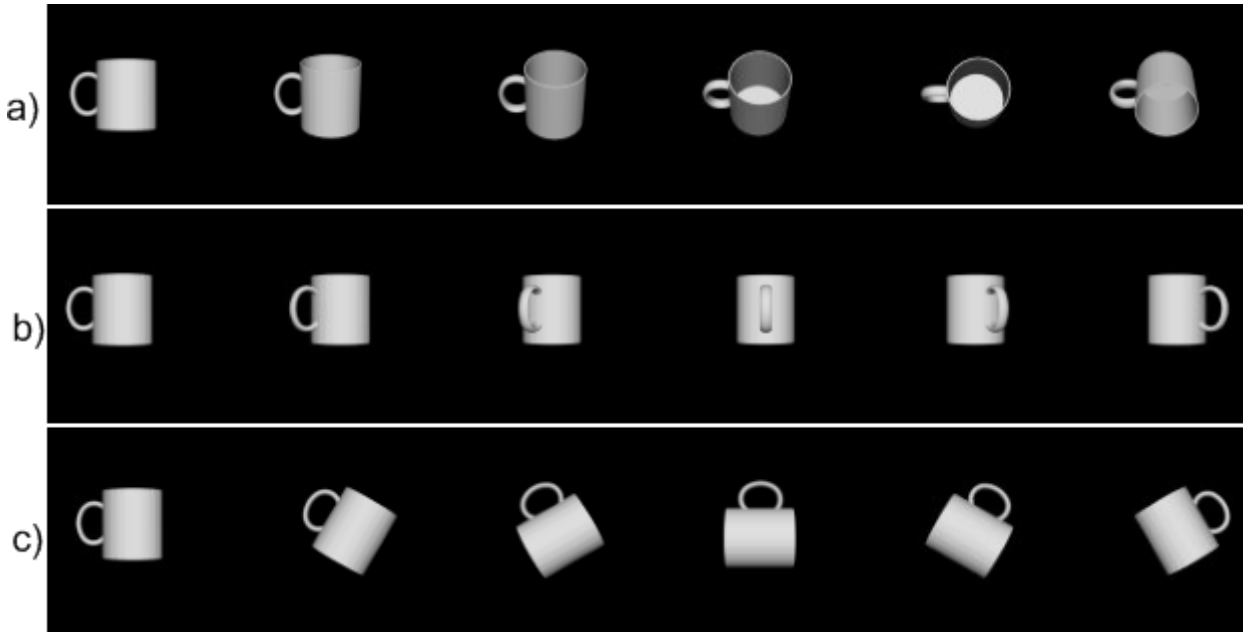


Figure 1.5: Views of the object captured with different a) camera elevations b) azimuths c) in-plane rotations

those view changes that do not affect visibility of its support region [53, 2][1]. For example, if a shape descriptor has a support region on a side of a cylinder (Figure 1.6 (a)), its support region remains visible under transformations shown in Figure 1.6 (c)-(e), but not in Figure 1.6 (b).

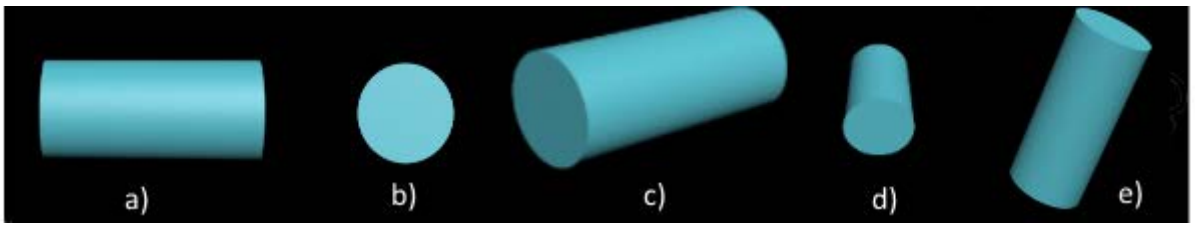


Figure 1.6: Illustration for view invariance

Another type of geometric invariance, which is quite often discussed in the computer vision and shape analysis literature is invariance to the deformations of non-rigid shapes. In particular, there exists a large class of shape descriptors that is invariant to the *isometric deformations* of shapes, i.e. those deformations that do not change geodesic distances between points on a surface. Isometrically invariant shape descriptors are particularly important for the analysis of deformable objects with articulated parts.

[1]This is termed by Besl as an **invariance when visible** [53]

1.6 Structure of the Thesis

The rest of this thesis is organized as follows. Chapter 2 overviews the literature on 2D representations. It includes an overview of handcrafted image features and deep learning methods, including a detailed overview of compositional hierarchical methods. Chapter 3 describes the literature on features used in 3D shape analysis. This chapter has a similar structure, i.e. it starts from the overview of the handcrafted shape features, and then describes the deep learning methods used for 3D shape analysis. Chapter 4 describes common principles of both the compositional hierarchical systems described in this thesis, and provides a high-level description of the learning and inference pipelines. Chapter 5 describes the view-based compositional hierarchical system and the experiments on object categorization using this system, while Chapter 6 describes the surface-based compositional hierarchical system and the results of its experimental evaluation.

CHAPTER 2

LITERATURE REVIEW: REPRESENTATION OF 2D IMAGES

This chapter overviews the existing features for 2D images, i.e. RGB, grayscale or binary images, and the object categorization approaches exploiting these features, and analyses properties of these features and approaches. The approaches to object categorization can be broadly classified into two large classes, namely the approaches mainly exploiting handcrafted image features, and deep learning approaches. In this chapter I follow this classification and partition the literature review correspondingly. Section 2.1 is dedicated to the most popularly employed handcrafted image features, their properties, and the methods exploiting these features. Section 2.2 is dedicated to deep learning methods, especially the methods based on ANNs. Section 2.3 is dedicated to compositional hierarchical systems, which can be considered as a sub-class of deep learning methods, however, since they are the main subject of research of this thesis, the related literature is reviewed in the separate section.

2.1 Handcrafted Image Features

This section overviews several types of hand-designed image features described in the literature, such as local image descriptors, global image descriptors, and texture descriptors. It is also shown that the computer vision systems using ensembles of different handcrafted

image features usually achieve better performance compared to the ones using features of a single type.

2.1.1 Local Image Descriptors

Local image descriptors are often used to describe local neighborhoods of image interest points, e.g. Harris points [54] or saliency points [55]. Each local image descriptor represents a vector in a multi-dimensional space characterizing the local image structure in a neighborhood of an interest point. Methods based on handcrafted local image descriptors have been intensively used for different computer vision tasks, such as instance and category recognition [56, 57, 58, 59], localization [60], texture recognition [61], object tracking [62], image registration [63] and many other tasks. There exists a large number of discriminative local descriptors proposed in the literature, for instance, SIFT [7], SURF [64], PCA-SIFT [11], GLOH [12], etc.

There are several desired properties making local image descriptors suitable for different image processing and computer vision tasks, for instance, their *discriminative power*, *robustness* to various types of image degradations, *invariance* to different photometric and geometric transformations, compactness, and computational efficiency. For example, the scale-invariant feature transform (SIFT) which is one of the most intensively used local image descriptors, is proved to be (i) very discriminative, (ii) robust to noise and image blurring, (iii) invariant to in-plane rotations, (iv) to a certain extent invariant to scale and viewpoint changes, (v) robust to illumination changes [7, 12, 54, 65].

Let us now briefly consider the key principles on which local descriptors can be built to achieve the aforementioned properties. For example, SIFT descriptor represents a block-wise histogram of image gradients extracted from a neighborhood of an image interest point. There are several techniques employed when building this histogram. First, the most appropriate scale for each interest point is found by convolving the original image with Gaussian filters with increasing parameter σ, and searching for the extrema of the Difference of Gaussian (DoG). This technique helps to achieve a certain degree of scale

invariance. Second, once the appropriate scale is found, the dominant gradient orientation in the neighborhood is computed. Histograms of gradients are then made relative to this dominant orientation, thus providing rotational invariance of the descriptor. A neighborhood of an image interest point is split into 16 bins, and the histograms of gradients are built for each bin and then stitched together into the single 128-dimensional vector, having a large discriminative power. There are multiple papers proposing further improvements of SIFT, for instance, PCA-SIFT [11], GLOH [12], etc.

There exist different types of object categorization approaches based on handcrafted local image descriptors. Some approaches, commonly referred as "*bag-of-feature*" or "*bag-of-words (BoW)*" [66, 67], completely disregard information about positions of local features in images and spatial relations between them. BoW methods usually store a large collection (termed *visual vocabulary*) of clustered local image features and discriminative image patches which are often termed *visual words*. Given such a visual vocabulary each object category can be represented by a sparse histogram showing occurrence counts of visual words in images depicting objects of this category. There are several severe drawbacks of such methods, for instance, that (i) spatial context is not considered, i.e. only occurrences of visual words in an image do matter in this case, while the information about their positions and spatial relations between them is disregarded, (ii) a large number of matching operations is required, as each detected image feature usually has to be matched against each visual word, (iii) visual words represent local features only, while the description of different mid-level and high-level visual cues is missing.

There exists, however, a number of more powerful object categorization approaches based on local image features going beyond naïve BoW strategy. The authors of these approaches introduce additional frameworks or data structures for modeling spatial distributions of local images descriptors, and/or spatial relations between them [68, 69, 56, 57, 70, 58]. These frameworks are usually purposed to provide a certain description of the global structure, i.e. a model of possible valid configurations of local descriptors in images representing object categories. For instance, the Implicit Shape Model, proposed

by Leibe *et al.* [56] starts with the learning of a vocabulary of visual words (termed *codebook*) and then learns the shape models specifying where in objects the visual words can appear consistently with each other.

2.1.2 Global Image Descriptors and Texture Descriptors

Except for local image descriptors, we have discussed in the previous subsection, there exists a number of global image features which describe either an image as whole or a large image segment. The simplest global descriptors represent greyscale or colour histograms of raw pixel data [71, 72]. There are global descriptors that exploit different geometric characteristics of objects, for instance, area, perimeter, and compactness or other characteristics, such as moments [73].

One of the most popularly employed global descriptors is called Histogram of Oriented Gradients (HoG) [8]. In this method either the entire image domain or a large image segment representing a region of interest is partitioned into several sub-regions; then orientational histograms of image gradients are computed for each sub-region separately and then stitched together into a single descriptor. This descriptor is quite discriminative and to some extent robust to occlusions and illumination changes [74, 75].

There exists a large class of handcrafted image descriptors dealing with texture of objects. A huge number of texture descriptors are proposed in the image processing and computer vision literature. There are texture descriptors based on statistics of the first order (e.g., moments, entropy), statistics of the second order (e.g. based on co-occurrence matrices); there are also texture descriptors exploiting signal processing methods (filter banks, Fourier or wavelet transforms etc.), geometrical methods (e.g. Voronoi Diagrams), fractals and many other methods and techniques [76, 77, 78]. One of the widely used texture descriptors is called Local Binary Patterns (LBP). LBPs enable scale and rotationally invariant texture descriptors which have been utilized for many different computer vision tasks, e.g. object categorization, face detection and recognition, etc. [79, 80, 81].

Note that handcrafted image features of different types are often used in ensembles for

solving different computer vision tasks. For instance, Wang *et al.* [75] combined HOG and LBP for objects detection with the handling of occlusions, while Zhang *et al.* [82] used the same combination of features for object localization. Lisin *et al.* [83] used a collection of different local and global features for object category recognition. It has been shown that ensembles of different handcrafted features often achieve better results compared to the results obtained using only a single type of image feature.

2.1.3 Limitations of the Handcrafted Image Features

In general, handcrafting of image features requires good domain expertise, creativity, and solid engineering and practical skills. In the previous two subsections it is demonstrated that there exists a huge number of hand-designed image features of various types. These features address different visual aspects of objects, such as shape, colour, and texture. They are built on absolutely different heuristics and exploit different mathematical tools. And yet the existing handcrafted image features are quite far from perfection.

It has been noted that the progress of the methods based on handcrafted image features substantially slowed down around 2010-2012 [13]. The apparent reason for this is that in general the complexity and the variability of image structures is too large to be fully captured by handcrafted image features of any type and handcrafted object or category models. The number of existing objects and object categories is very large, while every single object may appear in images in near-infinite number of ways due to a variability of possible relative positions and orientations of this object relative to a camera, changes of illumination, articulations (for non-rigid objects), occlusions, etc. All these reasons explain that the overall complexity of the object categorization problem is **enormously large** and, therefore, it is very hard to manually design image features that are sufficiently well generalized to work in multiple different scenarios.

The introduction of the large-scale object categorization datasets [15] and challenges [14] made it quite difficult to achieve further rapid progress with handcrafted image features and hand-designed frameworks exploiting them.

2.2 Deep Learning in Computer Vision

While the progress of the methods exploiting handcrafted image features substantially slowed down in 2010-2012, deep learning methods became the most popularly employed tool for a large variety of computer vision tasks. Deep learning is a subfield of machine learning represented by a number of approaches that learn a multiple layer representation of the input data, i.e. a hierarchy of features, where high-level abstractions are defined in terms of lower-level ones and relations between them. The main reason for the tremendous success of the deep learning methods in computer vision is that these methods successfully bridge a large semantic gap between local features and global high-level abstractions by statistically learning the mid-level cues representing structural dependencies of features on multiple processing layers. Inference in deep learning methods is also performed on multiple hierarchical levels which help to gradually restrict an otherwise computationally prohibitive search space. For instance, in CNNs inference is represented as a sequence of convolution and pooling operations which are done on multiple processing layers.

The work of Krizhevsky *et al.* [17], published in 2012 demonstrated that their CNN-based framework (called AlexNet) outperformed the existing methods based on handcrafted features by a large margin in the large-scale object category recognition problem [15]. This publication is sometimes considered to be one of the most important milestones, which ultimately changed the main focus of attention of the computer vision community from methods based on handcrafted features to deep learning methods.

2.2.1 Biological Motivation

The concept of deep learning is supported by findings about biological vision systems, showing that the seemingly effortless performance and speed [3] of biological vision systems becomes possible due to their hierarchical multi-layer architecture. A large number of papers dedicated to the biological visual systems describe a hierarchical organization and functional mechanisms of primates vision systems, for instance, [84, 85, 86, 87, 88,

89]. Kruger *et al.* provide a good review [90] of the related literature, describe the functional principles and structures that underlie the primate visual cortex and summarize the principles that may be used in computer vision systems. Some of their findings can be described as follows:

- A large part of the primate neocortex (up to 55 percent) deals with the vision pipeline which has a hierarchical organization comprising six layers;
- Neurons in the lower layers respond to primitive features (e.g. edge orientation, disparity, motion) over small local regions of visual space, termed receptive field. The information about activations of these neurons is then passed to neurons of the higher layers which respond to more complex features;
- Receptive fields cover larger and larger regions of the visual space for the higher layers;
- Visual system of primates contains separate, though having multiple inter-dependencies, channels that process different types of visual information such as colour, contour, motion, texture, and 3D shape information.

The organization of biological vision systems exhibits several computational advantages. Since the layers are built on top of each other, there exists shareability (reusability) of the elements of the lower layers, which is a very important block facilitating generalization capabilities of vision systems. The existence of separate channels for different visual aspects (such as colour, texture, motion, etc.) leads to a more compact representation and helps to avoid a combinatorial explosion of an integrated representation. The existence of separate channels also facilitates the ability of robust inference when the information from some channels is missing.

2.2.2 Artificial Neural Networks

Despite the recent increase in popularity of the deep learning methods, they first appeared a long time ago, for instance, Fukushima's Neocognitron [91] was proposed in the 1980s. Currently artificial neural networks represent an extremely fast-growing area, such that many new publications on ANNs appear every month. That is why there exists a large number of variations of ANN-based architectures and many of them are branched from a few original parent architectures. The existence of multiple software tools, such as Caffe [49], Torch [50] and Theano [51] providing a very rich functionality for artificial neural networks and supporting massively parallel computations on GPUs facilitates rapid development of new systems thus causing very fast progress in the field.

The existing ANN-based methods can be split into two large categories: supervised and unsupervised. The unsupervised methods are represented by autoencoders, restricted Boltzmann machines (RBMs), and deep belief networks (DBNs), while supervised methods are represented by the convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Autoencoders [92, 93, 94] are unsupervised neural networks which have input and output layers with the same number of nodes, and several hidden layers between them. The main purpose of autoencoders is to reconstruct the high-dimensional input data using low-dimensional codes, which are computed in the central hidden layer of the network. Autoencoders can be used for dimensionality reduction, image denoising [93] and for other tasks. In 2012 Le *et al.* [94] proposed a 9-layer autoencoder interchanging three times the local filtering, pooling, and local contrast normalization. Although their architecture was trained in an unsupervised manner on unlabeled images, and therefore the features were not tuned for category label prediction, the authors applied the features to the object categorization problem and reported nearly 70% accuracy improvement over the previous state-of-the-art methods based on handcrafted image features (namely 15.8% vs. the previous best result [95] demonstrating 9.3 % accuracy).

Restricted Boltzmann machines (RBMs) [96] are generative stochastic artificial neural

networks that learn a probability distribution of the input variables. RBMs comprise a layer of visible (binary or real-valued) units and a layer of binary hidden units with no intra-layer unit connections. A deep belief network is a generative neural network with multiple layers of hidden units. Units of the adjacent layers are fully connected, but there are no connections between units of the same layer. That is why DBNs are often considered as stacks of restricted Boltzmann machines where each subnetwork's hidden layer is used as a visible layer for the next layer. Deep belief nets became particularly popular after Hinton's publication [97] in which he proposed a fast greedy learning algorithm, in which the layers are learned sequentially from the lowest to highest and using the previous layers activations as inputs for the next layer. In addition, another algorithm for fine-tuning of a DBN can be applied in order to increase its discriminative power. One of the most attractive properties of the DBN-based architectures is that, since they are generative models, they can be used to generate new samples for the learned distributions, i.e. they can be applied not only for recognizing images but also for generating them. For instance, a DBN trained on a dataset of images of handwritten digits can be used to generate images looking as plausible handwritten digits [98]. Lee *et al.* [99] proposed a convolutional deep belief network which, in contrast to standard DBNs, has a property of translational invariance (which is achieved by using convolutional operations) and allows applying DBNs to images of large size, by exploiting max-pooling layers which are purposed to downsample the representation at higher layers. Different architectures based on DBNs are also used in 3D shape analysis, which will be discussed in Chapter 3.

The most commonly used in computer vision type of ANNs is convolutional neural networks. Convolutional neural networks have been applied for many computer vision applications, such as object detection and segmentation [13], object categorization [17, 100, 101, 102], object localization [100], scene classification [103], fusion of 2D features and surface features [6], etc.

CNNs typically comprise several interchanged convolutional and pooling layers, followed by two or three fully connected layers taking input weights from the whole image

domain. One of the most influential CNN-based architectures is AlexNet of Krizhevsky *et al.* [17], which is also implemented in different software libraries, such as CaffeNet [49] and Theano [51], and from there serves as a basis for many other recent CNN-based methods [13, 26, 100, 102]. AlexNet comprizes five convolutional layers, most of which are followed by max-pooling layers. The first convolutional layer contains $3 \times 224 \times 224$ neurons, which are activated by the pixel values of an input 224×224 RGB image. If input images are larger than 224×224, they have to be downsampled and/or cropped before undergoing AlexNet. The first convolutional layer has relatively large receptive fields of size 11×11 nodes and 4 nodes distances (called *strides*) between the neighbouring receptive fields[1]. Higher layers of AlexNet are represented by three fully connected layers with 4096 units each, and the output *softmax* layer comprising 1,000 units[2]. The output of the softmax layer is a set of 1,000 positive numbers which sum up to 1 and can be considered as a probability distribution of the predicted category labels.

Many subsequent papers from different authors proposed modifications of different parameters of this architecture. Zeiler and Fergus [101] used smaller receptive fields at the first convolutional layer 7×7 and decreased strides to 2 nodes. Simonyan and Zisserman [100] further decreased the receptive field at the first convolutional layer to the size 3×3 nodes and made 1-node strides (thus creating receptive fields centered in each node) and studied an influence of the depth of the network, i.e. the number of convolutional and pooling layers.

Some authors use CNNs pre-trained on ImageNet for object categorization on other datasets. There are two main ways how this can be done. The first way does not assume re-training of a CNN; it involves only inference of features from different layers from novel images and then the usage of linear SVM or other classifier for recognition. The experiments show that the features learned by CNNs on ImageNet are general enough to be successfully used for object categorization on other datasets[3]. For instance, Schwarz

[1]Receptive fields in this case are partially overlapped, tiling the input image.

[2]Corresponding to the number of object categories in the ILSVRC challenge [14].

[3]Except of the features of the higher fully connected layers and the softmax layer, which are very category and data-specific.

[102] used the features extracted from the CNN pre-trained on ImageNet for categorization of the Washington RGB-D dataset [48] and achieved the state-of-the-art results. Another commonly used technique is *fine-tuning* of the system in which the lower layers are taken from a pre-trained network without changes, while the higher layers of the network are re-trained on a new dataset. This approach is commonly used because it is much faster to re-train only higher layers and because the features of lower layers learned from ImageNet are proved [104] to be sufficiently general, i.e. not specific to particular datasets or object categories.

2.3 Compositional Hierarchical Architectures

In this Section the literature on hierarchical compositionality and existing 2D compositional hierarchical systems is reviewed. Most of the existing compositional hierarchical architectures are built on the set of common design principles (outlined in Section 1.1.5). They usually have primitive image features at the lowest level, more complex compositional parts representing structural dependencies of subparts at the middle levels, and abstracted object or image models at the top levels. Compositional parts are usually non-rigid, i.e. they tolerate some variance of relative positions of subpart. Description of the lower layer parts is commonly more precise, while parts of the higher layers are described more coarsely. Many authors report common properties of the compositional hierarchical systems, such as reusability of parts across different objects and categories [105, 106, 35, 36, 31, 107, 108, 42] and sublinear growth with the number of images/categories used for training [109, 35, 107, 42].

Having emphasized that most of the compositional hierarchical architectures are common in spirit, as they mainly follow similar general principles, it has to be pointed out that most of them substantially differ from each other in mathematical formulation, representation, architecture and the actual mathematical means used in the learning and/or inference phases. That is why it is very hard or impossible to present a commonly ac-

cepted mathematical theory employed across multiple compositional hierarchical systems. In fact, different researchers and research groups often introduce different terminology and notation, propose different mathematical formulations of the problem and set up their own compositional hierarchical systems. It should also be mentioned that there are no software engines, such as toolboxes or libraries, commonly used across different compositional hierarchical systems, therefore these systems are usually developed from scratch independently from each other.

These are the reasons why the following literature review mainly focuses on a comparison of different compositional hierarchical systems on a conceptual level, and on the level of the most important design principles and properties of these systems, having in mind that actual mathematical formulations and solutions are often very different anyways. The rest of this Section is organized as follows. Subsection 2.3.1 describes the theoretical work in the area of hierarchical compositionality and related areas, Section 2.3.2 describes computer vision systems based on stochastic grammar formulated as And-Or graphs. Though these systems are not usually considered as compositional hierarchical ones, they are closely related. Section 2.3.3 compares existing compositional hierarchical systems based on different criteria, while Section 2.3.4 analyses the level of experimental evaluation of the existing compositional hierarchical systems.

2.3.1 Theoretical Work on Hierarchical Compositionality

There exist several theoretical works describing the principles of hierarchical compositionality and closely related topics. Geman, *et al.* [28] propose a mathematical formulation for hierarchical compositionality. They justify the approach where objects are represented by a hierarchical library of compositional parts statistically learned from the training set. The authors show the importance of statistical learning of the mid-level features (e.g., arcs, junctions) which are in turn used as building blocks for higher level parts. Composition rules are defined as a set of syntactic rules under which subparts are composed to form parts of the next layer. Recursive application of the composition rules should lead to

a set of recognizable shapes. The authors suggest that the *redundancy* of a representation, i.e. the possibility to have multiple explanations for the same data may be a good idea since the existence of multiple explanations provides higher chances of finding the correct solution through several alternative inference paths. The authors also show connection of their vocabulary learning approach with the Rissanen's Minimal Description Length (MDL) principle [110].

Theoretical works of Potter [44] and Huang [45] propose making a scale and rotation invariant compositional hierarchical system by introducing relative coordinate systems (in contrast to the image-based "absolute" coordinate system). However, they demonstrate the performance of their methods only for handwritten digit and letter recognition systems, while the complexity of real-world images (and 3D structures) is substantially larger; that is why introducing relative coordinate systems for real world images may be a very difficult task. The recent work of Mete Ozay *et al.* [39] proposed another way of achieving of scale and rotational invariance by defining a shape manifold within a hierarchical compositional system.

The recent work of Yuille [41] published in 2016 summarizes the main principles of hierarchical compositionality and the properties of the existing compositional hierarchical systems. He provides a complexity analysis of the learning and inference phases in compositional models and analyzes the applicability of compositional hierarchical systems to parallel computing. This work of Yuille is interesting from the theoretical point of view, however, it is primarily focused on summary and performance analysis of the existing hierarchical compositional frameworks, and its purpose is not to propose novel compositional hierarchical frameworks.

The theoretical work of Guo *et al.* [111] formulates a *measure of importance* for 2D object shape parts. The importance of each local part is measured based on its contribution to the perception and recognition of the global object shape. The authors compute a measure of global shape reconstructability given local contour fragments. Even though the results of this work are not used in this thesis directly, we study and experimentally

evaluate different measures of part importance which are employed in the *part selection* procedure where decisions about promoting the compositions to the higher layers are made (see Chapter 5 for more details).

2.3.2 And-Or Trees

In this subsection the computer vision architectures based on *And-Or Trees (AOT)*, also referred to as *And-Or graphs*) are discussed, since they are closely related to the compositional hierarchical architectures. The term *And-Or Tree* was first introduced in 1984 by Pearl [112] in the area of AI, however, a number of computer vision algorithms based on AOTs were developed much later (since 2006) by H. Chen *et al.* [113], Y. Chen *et al.* [114], S. C. Zhu and D. Mumford [115], L. Lin *et al.* [116, 117], Z. Xu *et al.* [32], Z. Si *et al.* [35], Y. Lu et al [118], Li et al. [119], W. Hu *et al.* [43], Wu *et al.* [120]. AOT is a stochastic image grammar that provides reconfigurable templates for representing valid configurations of image features. *And-nodes* represent compositions of subparts with certain spatial relations, while *Or-nodes* serve as switching variables that choose between multiple possible sub-configurations of objects. Thus And-Or trees are capable of describing different spatial configurations of models with a large shape or appearance variability. Methods based on And-Or trees have been applied to different computer vision tasks including object categorization, tracking, detection, segmentation and object/image parsing. In some works, e.g. [116] it is shown that And-Or trees can represent generative models, stochastic sampling from which enables generation of new category instances.

Many compositional hierarchical methods do not use And-Or Trees directly, but follow similar principles or use similar data structures for hierarchical representation (e.g. similar tree-based structures are used by Zhu *et al.* [33, 121]). The compositional hierarchical systems proposed in this thesis uses OR-nodes as a switching variable between different spatial arrangements of subparts forming similar shapes on each layer, except the first one. More details are presented in Chapters 4, 5, and 6.

2.3.3 Existing Hierarchical Compositional Systems

In this subsection we compare different compositional hierarchical and related architectures based on different criteria.

Application of the compositional hierarchical systems. Compositional hierarchical systems have been used for many different computer vision tasks, such as object categorization [122, 105, 31, 30, 123, 124, 107, 117, 125, 126, 127, 36, 108, 42, 38], object detection [105, 128, 114, 34, 108, 35, 119, 42, 120], localization [109], recognition of letters and digits [44, 45, 106], image parsing and/or segmentation [115, 114, 33, 118, 37], face representation [32], domain adaptation [37], multi-view object detection and pose estimation [121, 38]. In general, the large variety of the problems addressed by the compositional hierarchical systems explains a vast range of possible design choices.

Generative and discriminative models. Most of the compositional hierarchical frameworks are generative models [105, 106, 116, 35, 125, 31, 42, 36]. Many authors advocate unsupervised learning of the shape vocabularies [129, 31, 130, 33, 121, 30, 35, 42, 37], while there are some approaches adding discriminative parts to the vocabulary [40].

First layer features. There exists a very large number of choices for the first layer features in compositional hierarchical models. Invariant discriminative local descriptors (mainly SIFT) are used as the first layer of some frameworks [125, 105, 126]. Small edge segments (edglets) are the most common choice, made in [87, 131, 128, 132, 31, 123, 30, 33, 36, 107, 42, 37]. Kokkinos and Yuille use two types of the first layer features, namely edge and ridge segments [109]. Hu *et al.* use edge segments and voxels [43]. Ommer *et al.* exploit a codebook of quantized local feature histograms built from salient image regions (Harris keypoints) [122, 108]. Small image patches are used by [133, 29, 129, 127]. Lin *et al.* [117] have a set of generalized geometric primitives such as circle, ellipse, rectangle, triangle, etc. at the first layer. Huang *et al.* [130] define a relatively large filter bank at the first layer. Finally, Si *et al.* use four types of atomic features: edge, flatness, texture, and colour features [35].

Number of layers. There exist several shallow hierarchical architectures involving two or three layers [87, 126, 105, 125]. However, most of the compositional hierarchies contain more than three layers.

Bottom-up vs top-down learning and inference. The most common learning/inference method is *bottom-up* [31, 129, 34, 121, 36, 42], starting from the detection of atomic features in the data and then proceeding through multiple layers towards high-level abstractions. There are *top-down* methods, e.g. [127, 134, 108, 109], performing recursive decomposition of complicated shapes into subparts. Some authors combine both top-down and bottom-up phases [30, 114, 33, 117, 123, 37], sometimes iterating between bottom-up and top-down phases several times.

The problem of the **geometrical invariance** of compositional hierarchical systems attracted relatively little attention so far. Several authors addressed the problem of rotational invariance of compositional hierarchies [44, 45, 37, 39]. Some authors [129, 126, 34] demonstrated a certain robustness of their systems to viewpoint changes, while [121, 38] evaluated their systems on multiple views of objects and demonstrated part-sharing across different views of the same objects. Ozay *et al.* [39] used shape manifolds to achieve scale and rotational invariance of the compositional hierarchy. Another way of achieving robustness to scale changes proposed in the literature is to downsample the testing images several times before they undergo the inference procedure, and then perform inference at several discrete scales [42].

Several authors show relations of their vocabulary-forming rules to Rissanen's MDL principle [28, 32, 36].

2.3.4 Experimental Evaluation of Compositional Hierarchical Systems

In this subsection of the reported properties of the existing compositional hierarchical systems are outlined.

Compositional hierarchical methods have been evaluated on a large number of datasets.

For instance, LHI animal faces dataset containing 2200 images of 20 categories was used by [117]. The Weizmann horse dataset comprising 328 images is employed by [114, 33, 121, 35]. Methods [35, 33, 121] use the Pascal VOC 2007 dataset containing 9,963 images. Caltech 101, comprising 101 categories and more than 50 images per category, is the most popular dataset used for evaluation of the compositional hierarchical methods used by [130, 33, 135, 117, 134, 124, 42]. However, we have to mention that the Pascal VOC 2007 and the Caltech 101 datasets are often used partially, i.e. only subsets of images are used for training. The UIUC car dataset includes 1050 training images and is used by [109]. The dataset of handwritten digits MNIST, comprising 60,000 images, is used in [130]. Geman *et al.* [106] use a small dataset of 458 car license plate images. [107, 109] perform experiments on ETH dataset. Ommer *et al.* [122] take 700 images, while Sudderth *et al.* evaluate their method on a dataset of 16 categories comprising 30 objects per category. Savarese *et al.* [126] work with images from 10 categories. Fidler *et al.* [31] use 3200 training images.

The researchers report multiple properties of compositional hierarchical systems, and these properties typically vary depending on the purpose of a system and the design decisions made. *Reusability* of parts across different objects and categories is reported by [131, 105, 106, 35, 36, 31, 107, 42]. Part sharing between different views of the same object is reported by [121, 38]. Sublinear growth with the number of images/categories used for training (also termed scalability) is shown by [131, 107, 109, 35, 42]. Most of the methods perform batch learning on a full training set, while some methods [131, 107] use sequential learning of categories such that new parts are included in the vocabulary only in case new training examples cannot be expressed by the existing vocabulary parts. Compactness of the representation is reported by [35, 31].

Fast inference, achieved by efficient indexing schemes, is highlighted in [31, 135, 36]. It is worth mentioning, that the idea of using indexing to reduce the number of matching operations during inference is not novel and has been employed in computer vision for a long time [136, 137]. However, compositional hierarchical architectures allow implementa-

tion of very efficient indexing schemes, which gradually and coherently restrict the search space on multiple hierarchical layers.

There are several conclusions which can be made. First, the compositional hierarchical methods have been evaluated on relatively small datasets which are at least three orders of magnitude smaller than the current state-of-the-art datasets, e.g. ImageNet [15]. Second, according to the best of our knowledge, there are no papers presenting the results of direct experimental comparison of compositional hierarchical systems against modern state-of-the-art ANN-based systems. In general, it is obvious that currently the compositional hierarchical methods are not at the level of performance of modern ANN-based methods in the object categorization task. It is also quite hard to compare (on basis of the literature review) many other properties of compositional hierarchies against the corresponding properties of ANN-based architectures. For instance, such properties as scalability, inference time, storage demands, etc., should rather be compared when both systems are evaluated on datasets of a comparable size, however, the literature review does not provide enough information for such a comparison.

CHAPTER 3

LITERATURE REVIEW: REPRESENTATION OF 3D SHAPE

There is no universally accepted way of representing 3D data. The most common data types for representing surfaces are depth images, point clouds, and triangulated mesh models, while 3D occupancy grids [138] are used to represent the volumetric data. Note, that other ways of representing 3D data may be used for some specific purpose, for instance, octrees and kd-trees are often used within different computational algorithms to speedup computations and for other purposes [139, 140]. This chapter reviews the main types of 3D shape features proposed for the object categorization and for the related problems, for example, 3D shape retrieval. Note that the volumetric representation is much less frequently used than the surface representation, that in why this chapter is mainly dedicated to surface shape features.

Handcrafted shape features have been dominant in 3D shape analysis for a long time, and only very recently (approximately since 2015) the situation started changing as a number of novel deep learning frameworks, specially designed for 3D data, appeared. However, in contrast to the 2D domain where the deep learning methods already demonstrated their superiority over the methods based on handcrafted image features in most of the mainstream challenges (e.g. the annual ILSVRC challenge [14]), the situation in the 3D domain is quite different. On the one hand, a very large number of proposed handcrafted surface shape descriptors demonstrate many useful properties, such as a dis-

criminative power and an invariance to rigid body transformations, scale changes and to isometric deformations of shapes. On the other hand, the deep learning methods for 3D shape analysis are still quite far from perfection and they have severe limitations, which will be analyzed in this chapter.

It is also worth mentioning, that in contrast to the 2D domain, where very large datasets [15] comprising millions of RGB images are used for training and evaluation of the state-of-the-art methods, much smaller datasets are used in the 3D shape analysis for these purposes. For example, the ModelNet [18] dataset of rigid objects, which is very popularly employed for evaluation of the 3D deep learning methods, contains around 10,000 models split into 40 categories. The existing datasets of deformable objects are even smaller, for example, SHREC'15 dataset [141] comprises 1200 models split into 60 categories. As for the datasets of depth images, the Washington RGB-D dataset [48] comprising 300 objects[1] split into 51 categories is still very intensively used. That means, currently in the 3D domain it is not yet possible to run the experiments of the same scale, as it is done in the 2D domain.

The rest of the chapter is organized as follows. Section 3.1 is dedicated to handcrafted shape features, namely local and global descriptors, while Section 3.2 describes the deep learning methods used in 3D shape analysis.

3.1 Handcrafted Shape Features

Handcrafting of shape descriptors is a very application-dependent and data-dependent process, that is why the required properties of the shape descriptors may vary substantially. The most commonly addressed properties of shape descriptors are their discriminative power, robustness to data degradations, invariance to rigid body transformations, i.e. translations and rotations, scale invariance and an invariance to isometric deformations of shapes.

[1]However, it has many views for each object

A huge number of handcrafted shape descriptors have been proposed, and these descriptors are based on very different heuristics and exploit non-similar mathematical tools. Different authors propose multiple ways of classifying the existing shape descriptors, for example, splitting them into local and global shape descriptors, view-based and model-based once, extrinsic and intrinsic ones, etc.[1] To organize this section I used a splitting into local and global shape descriptors, however, before going to the corresponding subsections I shortly explain what is meant by the other terms.

The view-based descriptors (e.g. [142]) are those that measure certain shape characteristics relative to a position of a camera, for example, encoding statistics of orientations of surface normals relative to a viewpoint direction. The Histogram of Oriented Normal Vectors (HONV) proposed by Tang *et al.* [142] assumes that each surface normal $\vec{n}$ originating at a point ρ increases a score of a histogram bin, chosen according to the position of ρ and orientation of $\vec{n}$ relative to the camera-based reference frame. Though such descriptors are usually quite easy to compute, and they can be effectively used in some applications, their general drawback is that they are much less robust to viewpoint changes and camera in-plane rotations than the model-based descriptors (e.g. [143, 144]), which are computed using the shape model properties only (e.g. geometric or statistical properties), and do not depend on the model's position relative to a camera.

In the literature, the surface shape descriptors are quite often split into *intrinsic* and *extrinsic* ones[2]. Surfaces typically represent 2D manifolds embedded in the 3D Euclidean space. The intrinsic descriptors depend on manifold properties only and do not depend on how this manifold is embedded in the 3D space, while the extrinsic descriptors depend on embedding of the manifold to the 3D Euclidean space. For example, the intrinsic descriptors may be based on geodesic distances between points on a manifold [145], as these distances remain unchanged both for rigid body transformations and isometric deformations of a shape, making these descriptors useful for different shape analysis applications,

[1] Obviously, due to a very large variability of the existing shape descriptors any type of splitting is not precise anyways, as some descriptors may be assigned to both classes or to neither of them.

[2] These terms originally came from the differential geometry of surfaces

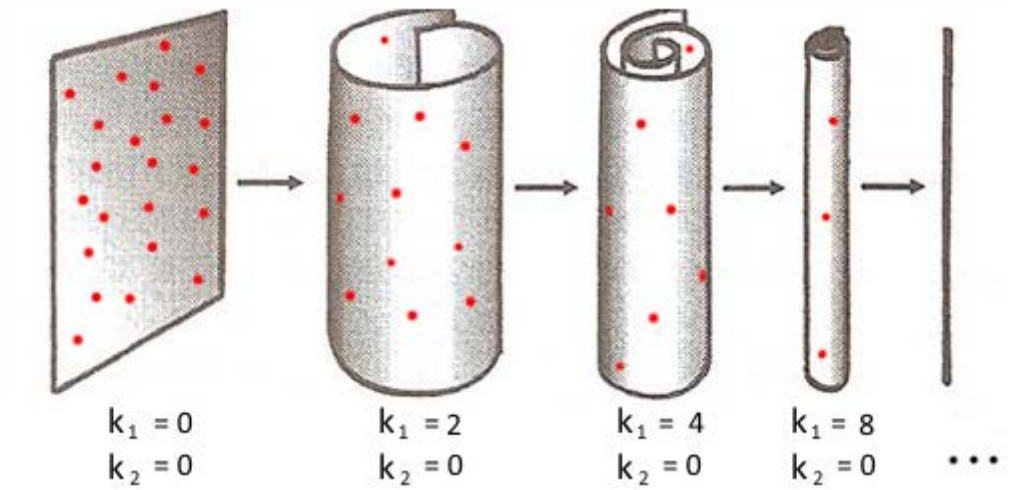


Figure 3.1: Intrinsic properties, e.g. geodesic distances between points on a manifold are independent of embedding of the manifold in the 3D space. In contrast, the extrinsic properties, e.g. principal curvatures k_1 and k_2 depend on the actual type of embedding

including analysis of deformable shapes. To illustrate this, let us consider the example shown in Figure 3.1. Assume, we have a manifold and a set of points defined on it. When we change the embedding of this manifold in the 3D space, the intrinsic characteristics, for instance, pairwise geodesic distances between points remain unchanged, while the extrinsic properties, e.g. principal curvatures k_1 and k_2 are being changed[1].

3.1.1 Local Shape Descriptors

Early approaches for 3D shape analysis did not exploit any specially designed local shape descriptors, instead they mainly used local geometrical surface characteristics, for example, taken from the differential geometry of surfaces. For instance, Medioni and Nevatia [146] used principal curvatures and Gaussian curvature for surface description, Vemuri *et al.* [147] used principal curvatures as well, while Besl *et al.* [53] exploited a relatively large number of geometrical surface characteristics, for instance, mean and Gaussian curvatures, principal curvatures and other characteristics based on surface partial derivatives and on the coefficients of the first and second fundamental forms of a surface. Though some of

[1]Note, however, that the Gaussian curvature $K = k_1 k_2$ which is a product of the two principal curvatures k_1 and k_2, neither of which is an intrinsic surface property, is an intrinsic property. In the illustrated example, Gaussian curvature $K = 0$ for any type of embedding.

these characteristics (e.g. principal curvatures) are still successfully employed within the recent state-of-the-art methods [21] the approaches based only on them did not have a sufficient discriminative power, that is why a large number of more discriminative shape descriptors have been proposed in the next decades.

In general, local shape descriptors are purposed to assign to a given point on a shape a number or a multi-dimensional vector representing local shape properties around this point. Some approaches involve computation of shape descriptors for each point of a shape, while other approaches search for special characteristic points (also termed *key-points*) on a shape, and compute the shape descriptors only in neighborhoods of these points. Some of the keypoint detection algorithms used in shape analysis extend the corresponding 2D algorithms, for example, Harris-3D [148] or 2.5D SIFT [149], while most of the algorithms are specifically developed for the 3D domain, for instance, the algorithms exploiting local curvature maxima [150], eigenvalues of the covariance matrix of the neighboring points [151], spectra of the Laplace-Beltrami operator [152], etc.

As for the local shape descriptors themselves, few of them also represent adaptations of the successful 2D descriptors to the 3D domain, for instance, depth-SIFT [153] or SURF [154], however, it is much more common to design specific descriptors for the 3D shape domain. All local shape descriptors, designed for the 3D domain, can be broadly split into several groups according to the mathematical tools they employ, for instance, to histogram-based, signature-based, and transform-based, kernel-based and other types of descriptors.

The first group of the local shape descriptors is **histogram-based** descriptors. Many local shape descriptors represent histograms of surface points[1] in a local neighbourhood. Johnson and Herbert [9, 155] introduced the two-dimensional histograms, called *spin images*, describing surface locations around a given point ρ on a surface. The support region of a spin image at this point is a cylinder of a pre-defined radius and height centered in ρ such that the axis of the cylinder is aligned with the surface normal $\vec{n}$ at the point ρ.

[1]Remember that surfaces are typically represented by point clouds or depth images, both of which basically represent a set of points on a surface

This support region is divided linearly into radial and vertical segments, thus forming a two-dimensional histogram. The score of each bin is computed by counting the points that fall within this bin. Since the input data may be represented by an irregular point cloud, each point included in the corresponding bin is weighted by the inverse of that points density estimate. Spin images have been employed in multiple applications, and have been for a long time considered to be a benchmark against which many other descriptors (e.g. [156, 157, 158, 159, 160, 161]) were compared. Different variations of spin images have been proposed to improve their discriminative power and robustness to geometric transformations such as scale and viewpoint changes [162, 163].

Rusu *et al.* proposed descriptors called Point Feature Histograms [143, 164] characterizing the local surface geometry around a given point ρ, by encoding both positions of points on a surface around ρ and orientations of surface normals at these points. Another key difference from the spin images is that *local reference frames* (LRFs) based on the Darboux frame[1] are defined for each point to partition the neighborhood of this point to histogram bins. In general, usage of local reference frames helps to align the underlying data making the descriptors invariant to rigid body transformations, such that those surfaces that differ from each other only by translation and rotation are characterized by the same descriptor values. Zhong *et al.* [144] also use a histogram of points in a local neighborhood of a given surface point, but they define the histogram bins using a LRF made of eigenvectors of the covariance matrix of the neighboring points. Kokkinos *et al.* [165] proposed a histogram-based descriptor called Intrinsic Shape Context (ISC) representing a local histogram defined on an intrinsic local polar coordinate system. ISC is not designed for any particular surface characteristic, instead, it can be applied to any scalar or vector field representing a local geometric property of a shape. Other examples of histogram-based local shape descriptors are [157, 166, 167, 161].

The second group of local shape descriptors is **signature-based** descriptors. In contrast to the histogram-based descriptors which aggregate certain geometric measures over

[1]Darboux frame represents a surface normal at a given point and two directions of principal curvatures

all points from the local neighborhood of a keypoint ρ, the signature-based descriptors compute geometric measures individually for each point of a subset of the neighborhood. There are different ways how a subset can be obtained. For instance, Stein and Medioni [168] define geodesic circles around a keypoint ρ and compute the signature encoding angular distances between the surface normal at the point ρ, and surface normals of all points on the geodesic circle. Sun *et al.* [169] also defined geodesic circles around a point ρ and then project these geodesic circles to a tangent plane of a surface at the ρ, thus obtaining curved lines on a tangent plane, which they call *points fingerprint* descriptor. Castellani *et al.* [159] defined a spiral pathway around the keypoint ρ and build a signature extracting different surface characteristics (e.g. principal curvatures) for all points along this pathway. Other examples of signature-based local shape descriptors are [156, 158, 160].

The third large group of local shape descriptors is **transformation-based** ones. These descriptors transform the local shape neighborhood of a keypoint from the spatial domain to other domains, for instance, to the spectral domain, and then compute shape descriptors by encoding the information in the transformed domain. For example, Hu and Hua [152] applied the Laplace-Beltrami operator to local surface patches and built a discriminative descriptor for shape matching and retrieval using the spectrum of this operator. Properties of the Laplace-Beltrami operator made this descriptor invariant both to rigid body transformations, and to isometric deformations of the shape. Knopp *et al.* [170] proposed voxelization of a mesh model and applied Haar wavelet transform to voxelized 3D images. Then the responses to the wavelet transform are used to define a LRF and build the descriptor. Another example of transformation-based descriptors is Optimal Spectral Descriptor proposed by Litman and Bronstein [171].

The next class of local shape descriptors is **kernel-based** ones. Bo *et al.* [172] introduce kernel descriptors for shape analysis, proposing three types of kernel-based descriptors for depth images and point clouds, namely size kernel descriptor, kernel PCA descriptor, and spin kernel descriptor. Sun *et al.* [10] proposed a Heat Kernel Signature (HKS)

descriptor based on a physical model of heat diffusion on a manifold, which is described by the *heat equation*. As the *heat kernel* represents a fundamental solution to the heat equation, the authors defined a diagonal heat kernel $h_t(\rho, \rho)$, also called *autodiffusivity function*, which can be physically interpreted as an amount of heat remaining at the point ρ after time t. The heat kernel signature is defined as $HKS(\rho) = (h_{t_1}(\rho, \rho), h_{t_2}(\rho, \rho), ..., h_{t_n}(\rho, \rho))$, where t_i shows different diffusion times and n is a number of time samples. HKS has become a very popular descriptor for many applications, because it is fully intrinsic, and therefore invariant to both rigid body transformations and to isometric shape deformations, it is dense, i.e. can be computed at each point on a shape, scale invariant (in the modification proposed by Bronstein *et al.* [173]), robust to noise and computationally efficient. Another example of the very popularly employed kernel-based descriptor is Wave Kernel Signature (WKS) proposed by Aubry *et al.* [174] which is based on the physical model of a quantum particle on a manifold described by the Schrodinger equation.

There exists a relatively large class of intrinsic isometrically invariant shape descriptors which are **based on geodesic distances** between points on a surface. Such approaches involve defining a set of points on a surface, measuring pairwise geodesic distances between these points (for example, using the fast marching algorithm [175]), and creating a *geodesic distance matrix* (GDM) based on these distances. Various descriptors computed from GDM have been proposed, for instance, the histogram comprising all matrix elements, largest singular values of GDM, largest eigenvalues of GDM, singular value decomposition of GDM, and many other descriptors [145, 176, 177, 178, 179]. Note, however, that the GDM-based descriptors can be both local, i.e. when geodesic distances of points located within a certain neighborhood are computed, and global, where the points used for GDM computation are distributed across the whole shape[1].

Concluding the subsection on the local shape descriptors we have to emphasize that this overview is quite far from covering all existing descriptors because in general, the handcrafted shape features is not the subject of the research of this thesis. A more detailed

[1]Having mentioned that, we will not describe GDM-based descriptors again in the next subsection dedicated to the global shape descriptors.

overview of local shape descriptors is presented in [180], while a very thorough overview of shape features (both local and global) applicable for non-rigid 3D shape analysis can be found in [181].

From the presented examples of the local shape descriptors, we can make several important conclusions. First of all, though some successful descriptors used for 2D image analysis have been adapted for the 3D domain, it is much more common to design special descriptors for this domain, since such descriptors are purposed to exploit a very rich surface geometry information. Second, the necessity of geometrically invariant shape descriptors has been widely emphasized by many authors. The first principle way of achieving invariance is to build the descriptors on top of the invariant shape characteristics [169, 145, 176, 177, 10, 173, 174, 165, 179], while the second way is to define a data-driven local reference frame (or at least one local axes) for aligning the data before computing shape descriptors [168, 155, 169, 157, 158, 143, 144, 167, 160, 166, 170].

3.1.2 Global Shape Descriptors

In contrast to the local shape descriptors describing the local shape properties, the global descriptors describe the entire 3D object, therefore such descriptors are much less suitable for recognition of a partially visible object, which is an important task in computer vision. Global descriptors are therefore more commonly used in 3D shape retrieval, than in computer vision, that is why in this we provide only a very short overview of those methods and recommend the surveys of 3D shape retrieval methods [182, 181, 183] for further reading.

Simplest global shape features address different geometric and statistical characteristic of shapes, such as the volume to surface ratio, statistical moments [184, 185]. Corney *et al.* [186] introduce convex-hull based indices, for example, the ratio of the object surface area and the surface area of its convex hull, the ratio of the convex hull volume and the object volume, etc. Elad *et al.* [187] used statistical moments as global shape features.

Histogram based descriptors are quite popularly employed in shape analysis. Fang *et*

al. [188] propose a global shape descriptor called Temperature Distributor (TD), representing a one-dimensional histogram. This histogram is computed in several steps. At the first step, a unit heat is applied to a point on a surface. Then the diffusion, driven by the heat equation is computed and the distribution of surface point temperatures is measured after a fixed diffusion time. Then this procedure is repeated for each surface point, and the average temperature distribution is encoded by the TD descriptor. Later Fang and his colleagues [189] proposed a Heat Shape Descriptor (HeatSD) describing probability distribution of HKS on a given shape, representing a two-dimensional histogram aggregating values of HKS descriptors at multiple time scales. For instance, if each HKS has n time samples, and the number of histogram bins is n_b then the HeatSD will represent the $n \times n_b$ histogram. Therefore, in contrast to the TD, the HeatSD is a multi-scale descriptor, providing a local-to-global description a shape. Both these descriptors are fully intrinsic and therefore invariant to rigid body transformations and to isometric deformations of shape.

Signature-based methods are also quite commonly employed as global shape descriptors. Osada's Shape Distributions [190] is a method for computing shape signatures for arbitrary 3D models, in which random points on a surface are first defined, and then different measures based on distance, angle, area and volume between those points are computed. Li *et al.* [191] use Monte Carlo sampling to extract distinctive shape signatures from 3D models.

Transform-based method are represented by Fourier-based descriptors [192], Kazhdan's Spherical Harmonic Descriptor [193], 3D Zernike moments [194], wavelet-based descriptors [195], and the descriptors based on the Laplace-Beltrami operator, for example, using the spectrum (eigenvalues) of this operator [196], and shape embedding using eigenvalues and eigenfunctions of the Laplace-Beltrami operator [197].

Tabia *et al.* [198] proposed a Covariance Descriptor for 3D shape retrieval, which exploits covariance matrices of local shape features. An interesting branch of global shape descriptors approaches involves estimation of object skeletons or medial axes [199, 200];

the matching is then done by establishing the correspondence between the skeletons.

3.2 ANN-based Methods for 3D Shape Analysis

It is shown in Chapter 2, that the convolutional neural networks achieve a break-through performance in object categorization from 2D images as well as in other problems such as object detection, segmentation, etc., and therefore have *de facto* become a default tool for solving many problems. Such results are possible not only because CNNs successfully learn structural dependencies of features on multiple hierarchical layers, thus bridging a semantic gap between low-level image features and high-level abstractions (object or category models) but also because CNNs are specially adapted for learning and inference from the image domain, which represents a rectangular or square area in the 2D Euclidean space, regularly sampled by a pixel grid.

Many important design principles on which the CNNs are based assume the existence of the domain of the type. For instance, this domain is naturally partitioned to the evenly spaced local receptive fields of a square form in which convolution operations are applied. Convolutional layers are usually coupled with pooling layers which are purposed to simplify outputs of convolutional layers by reducing the spatial resolution, typically by replacing each square region (e.g. 2×2 nodes) on the previous layers by a single node on the next layer. After several convolutional and pooling layers the output, which by this time has a low spatial resolution, is transferred to one or more fully connected layers, whose nodes take input values from the whole image domain, and therefore purposed to respond to the global image structures.

It has been pointed out that direct application of classical CNN architectures and similar methods for 3D shape analysis is a difficult task due to multiple reasons. First of all, surfaces shapes typically represent non-Euclidean manifolds embedded in the 3D Euclidean space, on which there is no shift invariance, and therefore convolution does not exist in a classical sense [19, 21]. There is a number of other problems, for example, how to

partition a manifold to evenly spaced receptive fields, and how to re-define pooling layers in absence of a regular sampling, which is naturally provided by a pixel grid in 2D image domain, but maybe not easy to define for non-regular surface shapes. Generally, due to these difficulties the adaptation of the CNNs to the surface shape analysis requires re-defining of many important ingredients of this architecture. Consequently, there is still no consensus on how these problems should be solved, that is why the researchers nowadays follow dramatically different ways, each of which, however, has certain limitations.

3.2.1 Mapping from Manifolds to the 2D Euclidean Domain

All the approaches described in this subsection do not apply deep learning methods in the input data domain, i.e. on surface manifolds, but rather exploit different ways of mapping the input data to the 2D Euclidean domain, where CNNs and similar architectures are very efficient, and solve the problem in this domain. There are multiple ways how this mapping can be done, for instance, using the computer graphics rendering techniques [22], projective geometry [23, 201], parameterization of a surface shape [21], etc.

Su *et al.* exploit the Phong shading model [202] to render grayscale images depicting each input 3D shape model from multiple viewpoints. Then they use these images as an input for the CNN, also introducing a special layer (termed *view-pooling*) which takes input weights from multiple views thus performing a multi-view category recognition. Shi *et al.* [23] use the projective geometry, i.e. they define a cylinder projection around the object's principal axis, to compute a *panoramic image* of this object, and then apply CNN to the panoramic images. They also introduce a special layer, termed *row-wise pooling layer* making the representation invariant to rotations about the principle axes (though not invariant to rotations about two other axes)[1].

Another way of mapping surface shapes to the 2D domain is to produce depth images representing partial views of objects. Depth images can be computed using projective

[1]It should be mentioned that principle axis are quite often unstable [182, 203], especially for non-rigid shape models with articulated parts, that is why the methods using object alignment by PCA may be unstable for certain types of the input data.

geometry, or captured by special depth sensors, such as Kinect of PrimeSense Carmine. Zhu *et al.* and Feng *et al.* [25, 204] use depth images as an input for the autoencoders, while Socher *et al.* [24] apply the architecture combining the CNN and the Recurrent Neural Network (RNN) to learn the category representation from depth images. In these cases, the depth images are processed by the deep learning algorithm in the same way as if they were ordinary grayscale images, i.e. no special techniques purposed to exploit the surface geometry are introduced. A general problem of such approaches is that in this case a very rich surface geometry information can not be very fully and efficiently utilized.

In an attempt to mitigate this problem, several authors proposed *colorization* of depth images, i.e. a conversion of a depth image to a multi-dimensional image (usually three-dimensional one), where each channel encodes certain geometric characteristics of a shape. For instance, Girshick *et al.* [13] proposed HHA colorization, in which the **h**orizontal disparity, the **h**eight above ground and the **a**ngle with gravity are computed. Shwartz *et al.* [102] colorized the depth images by computing the distances of each point from the object's center, and then applying different color palettes to convert these distances to RGB values. [205] Bo *et al.* propose colorization of depth images using x, y and z components of surface normals at each point, while Eitel *et al.* [26] normalize depth values to the range from zero to 255 and then use the jet colormap to convert the normalized depth values to RGB values. All these colorization techniques lead to a sufficient improvements of the object categorization accuracy relatively to the processing of raw depth data by CNNs.

Another way of mapping surfaces to the 2D Euclidean domain is to use surface parameterization. Sinha *et al.* [21] applied two types of surface parameterization, namely authalic and conformal ones, to associate each point on a manifold to the corresponding point in the 2D image domain. They compute certain surface characteristics for each point on a manifold, i.e. HKS for deformable objects, and the principal curvatures for the rigid ones, and use the chosen surface parameterization to project these characteristics from a manifold to the 2D image of a special type, called *geometry image*. After that CNNs are used to classify these geometry images.

In general, all the approaches presented in this subsection have several disadvantages. Firstly, all presented mappings from one domain to another, i.e. rendering, projective geometry and surface parameterization, unavoidably induce certain errors. Secondly, since CNNs are specially designed for the 2D domain, they are not based on any special design principles that would facilitate efficient exploitation of a very rich surface geometry information. This is evidenced by the fact that the colorization techniques, by only incorporating very simple geometric features to the CNN input substantially improve the object categorization results. This may indicate that CNNs cannot automatically efficiently learn such features from the raw depth data, and, presumably, the proposed colorization techniques only mitigate, but not fully solve this problem. Thirdly, due to a non-invariant nature of standard CNNs (discussed in Subsection 1.1.3), any system exploiting them generally assumes that each input shape model should be presented in the training set at multiple orientations (i.e. rotated multiple times around all three axis, which is very computationally inefficient), otherwise the CNN-based systems will be be very likely to fail to recognize objects from radically new viewing angles and in-plane rotations.

3.2.2 ANNs in the 3D Euclidean Domain

Except for the deep learning methods that map surfaces to the 2D Euclidean domain there exist a few recently proposed (in 2015-2016) methods working in the 3D Euclidean domain. They first perform transformation of surfaces to the 3D domain, by voxelization of the input data i.e. computing the 3D occupancy grids for each input shape, and then applying the deep learning methods on these grids.

Wu *et al.* [18] proposed a representation (called 3D ShapeNet) of a probability distribution of binary variables on a 3D occupancy grid (of size $30 \times 30 \times 30$) using a convolutional deep belief network. Binary variables take the value 1 when a voxel intersects the surface, and the value 0 to represent the empty space. The representation, they proposed is generative, and therefore can be successfully applied not only for object category recognition

but also for shape completion given a partial view of the data.

Maturana [4] proposed supervised discriminatively trained 3D convolutional neural network (called VoxNet) taking 3D occupancy grids (of size $32 \times 32 \times 32$) as the input. In contrast to 3D ShapeNet, VoxNet employ different types of occupancy grids, i.e. the binary grid and the density grid which are used to differentiate the empty and the unknown space. VoxNet outperforms the 3D ShapeNet on the ModelNet dataset [18] for object categorization, yet providing a much more compact representation and performing much faster learning and inference. Garcia *et al.* [5] also proposed a supervised 3D convolutional neural network (called PointNet) using 3D density occupancy grids, counting the number of data points of an input point cloud that reside in each voxel. They substantially increased the spatial resolution ($60 \times 60 \times 60$ voxels). Sedaghat and Thomas Brox [206] further developed the idea of 3D convolutional networks working on 3D voxel grids, proposing the ORION architecture, which is ORIentation-boosted vOxel Net. Their main contribution is that they changed the CNN's cost function by adding the term measuring loss on orientation to enforce the system to output not only category labels but also object orientations.

The idea of extending CNNs from the 2D square image domain to the 3D cube voxel grid to categorize 3D shapes looks very promising, and apparently, the methods following this path will be further developed in the nearest future. However, the methods proposed so far have substantial drawbacks, such as a very low spatial resolution, and lack of invariance to different types of transformations. The first drawback is explained by the third extra dimension, leading to larger memory demands (compared to the 2D images) and larger computational overhead. As for the invariance, these architectures are not invariant to rotations[1], which can be only compensated by augmenting the training data, i.e. representing each training model at multiple orientations. Another drawback, pointed out by [21] is that these methods are not well suited for recognition of deformable objects, and they often fail to recognize isometric deformations of shapes, for instance, deformation

[1]Due to the same reasons as 2D CNNs, i.e. mainly because the convolution operation is not rotationally invariant

of a standing person to a sitting one. This happens because the volumetric occupancy grid for both of these human poses will be radically different, while the 3D CNNs do not have any special mechanism for exploitation of the intrinsic shape properties that remain unchanged under the transformations of this type.

In conclusion, we should stress that these methods are yet not at the level of performance of the methods using mapping of surface shapes to the 2D domain (e.g. [22, 21]), discussed in the subsection 3.2.1.

3.2.3 CNNs based methods on manifolds and other methods

There exists another branch of deep learning methods proposed by Masci and his colleagues [19, 20, 27] that attempt to apply CNNs directly on non-Euclidean manifolds, without mapping the data to the 2D or 3D Euclidean space. The authors re-define the most important of the CNN's ingredients to make this architecture applicable to the non-Euclidean domain. They defined a notion of *geodesic convolution* which can be understood as "correlation with surface template", and build the convolutional layers using this notion. In [20] the geodesic convolution is defined on basis of the patch operator [165], while [27] defines it using vertex frequency analysis framework [207]. Also the authors re-defined pooling layers introducing the *angular max-pooling* layer working on a local polar-coordinate reference frame. However, the main restriction of their architecture is that it learns only local shape descriptors, therefore the power of CNNs to automatically learn hierarchical features of increasing size (spanning the whole input image at the top fully connected layers) is not fully utilized by their architecture.

A few more ways of applying deep learning methods for 3D shape analysis have been proposed. Bruna *et al.* [208] proposed a spectral CNN, in which convolutional layers are applied in the Fourier domain. Fang *et al.* [189] proposed a 3d deep shape descriptor which is learned using an autoencoder. Their method can be described as follows. At the first stage they compute HKS local descriptors for each point on a shape and then use them to compute a HeatSD global descriptor (described in Section 3.1.2) for this shape. After

HeatSD histograms are computed for each shape of the dataset, the deep shape descriptor is learned from them using a neural network, with a cost function minimizing the intra-category variance and maximizing inter-category margin. That means the processing pipeline looks as follows: shape models → local handcrafted shape descriptors (HKS) → global handcrafted shape descriptor (HeatSD) → neural network → deep shape descriptor. Since their method is based on intrinsic descriptors, the deep shape descriptor is invariant to isometric transformations and rigid body transformations.

3.2.4 Compositional Hierarchical Methods for 3D Data

There are very few 3D systems that are to some extent similar to our work, and they are not closely related to the approach presented in this thesis. Hu and Zhu [43] proposed the AND-OR tree-based representation of surface shape features and contour features for learning car templates. They use the learned templates for simultaneous object detection, localization and pose estimation. They applied their method only to car detection in images.

Wessel and Klein [209] presented a framework for decomposing of 3D objects into sections which can be represented by planes, spheres, cylinders cones and tori. They introduced a probabilistic framework modeling the spatial arrangements between these shape primitives. However, their method deals with the simplest shapes (mainly hand-made objects) and hence is not suitable for general multi-class category detection.

CHAPTER 4

DESCRIPTION OF THE APPROACH

Both the compositional hierarchical frameworks proposed in this thesis, i.e. the view-based framework, presented in Chapter 5 and the surface-based one, described in Chapter 6, have many similarities, for example, they have shared design principles and similar learning and inference pipelines. The main purpose of this chapter is to describe the common design principles behind both compositional hierarchies and to provide a high-level description of the learning and inference algorithms, while the next two chapters will specify the details for each hierarchy, and present the results of their experimental evaluation.

Note that a high-level description of our system looks very similar to the description of the compositional hierarchical system of Fidler *et al.* [42], from which we borrowed some ideas, for example vocabulary learning based on clustering of statistical maps. However, since their system is designed for 2D contour features, while our frameworks works with 3D data, both these frameworks are very different on the level of details. Nevertheless, it is important to emphasize that the descriptions provided in Section 4.2 are to a large extent similar to the work of Fidler *et al.* [42].

This chapter is organized as follows. Section 4.1 describes the most important design principles of the frameworks proposed in this thesis, while Section 4.2 provides a high-level description of the learning and inference algorithms used in both frameworks.

4.1 Design Principles

4.1.1 Separate Representation for Surface Shapes

This thesis advocates the approach assuming that surface shape representation should be based on different principles than the representation(s) of other visual modalities, such as 2D contour shape, colour and texture, and, therefore, it should be developed separately. The main argument for such a choice is that, in general, a representation specially developed for a certain modality can much more efficiently exploit specific properties of this modality, that is why it may be more efficient than the unified representation for different modalities. For instance, a representation of surface shape features may have many advantages by intensively using very rich surface geometry information.

In general, there is no consensus point of view to this question. As shown in Chapter 3, the handcrafted 3D shape descriptors are mainly built on different principles than the 2D image descriptors, since they exploit various surface characteristics, such as surface normals, principal curvatures, geodesic distances between points on a surface, etc. In other words, most of the 3D shape descriptors are specially designed for the 3D domain and exploit properties and characteristics of this domain. On the other hand, many deep learning methods use the same representation and the same learning and inference algorithms for different visual modalities, for example, there exist multiple CNN-based systems, which are applied to RGB-D images, thus processing the surface shape and the appearance features within the same framework. There are, however, several deep learning approaches specially designed for 3D shape features (e.g. 3DCNNs, geodesic CNNs).

In the previous chapters (Sections 1.1.4 and 3.2) it was pointed that the CNN-based systems working with the RGB-D data cannot efficiently process raw depth data since these systems are not based on any special principles that would facilitate efficient exploitation of surface geometry. That is why a deep representation specially developed for surface shape modality should to a certain extend bridge this gap.

4.1.2 Properties of parts

Parts of the first layer. In this thesis the planar surface patches of a certain size are used as the first layer features, since they are very simple (primitive) and allow easy detection tests from different types of input data, such as depth images, point clouds and triangulated mesh models. In general, the choice of the first layer features is a very difficult problem. The literature review shows that a large variety of the first layer features have been used in different deep learning systems. Some authors use very simple features (e.g. edge segments), while others exploit the advanced hand-crafted descriptors (e.g. SIFT of HKS) at the lowest layers of their deep learning systems. In this thesis I advocate the approach where no hand-crafted features are used at any stage, and all low-level, mid-level and high-level shape features are statistically learned from the training data. Therefore, the first layer vocabulary should contain only the simplest features, which are later used as building blocks for more complex features on the subsequent layers. Planar surface patches are both very simple and easily detectable in input data, therefore they are used at the first layer of both compositional hierarchies.

Generalization capability of parts. The key property of each vocabulary part is that it is purposed to represent a range of shapes with similar geometric properties rather than a single particular shape. This property facilitates a certain generalization, in which similar surface shapes are represented by a single vocabulary part. For the atomic parts of the first layer, which represent planar surface patches, such a generalization is achieved by tolerating some deviations from the planarity, i.e. each "nearly planar" surface patch of a certain size is represented by a vocabulary part of the layer L_1. For compositional parts of layers $L_n, (n \geqslant 2)$, the generalization is achieved by the way how compositions are built, i.e. a range of possible relative positions of subparts is described using the parameterized distributions, which make all compositional parts tolerating a certain degree of shape variability. In this case each vocabulary part represents a shape model generalizing over a range of similar shapes, thus enabling **recognition of novel shapes** from within this range. Additionally, such a flexibility is purposed to make the representation more robust

to data degradations and distortions (e.g. caused by the projection).

4.1.3 Redundancy of the Representation

The key property of both representations proposed in this thesis is their redundancy, in which we allow: (i) the existence of part realizations with strongly overlapped support regions, (ii) the existence of activations of multiple vocabulary parts in the same data point. It is sometimes not possible to reliably infer certain structures or estimate local reference frames at the lower layers of the hierarchy, where receptive fields are small and therefore different properties of the input data (sampling, quantization) as well as data degradations (e.g. noise) may have significant influence. That is why in some locations inference may fail or produce the wrong output; the local reference frames may also be estimated wrongly. The proposed approach to tackling this problem is to allow the inference algorithm to produce multiple hypotheses (i.e. realizations of two or three different vocabulary parts at the same point) in a case of uncertainty, each of which is confirmed or rejected during inference of the next layers. Having multiple part realizations with strongly overlapped support regions helps to compensate for failures of the inference procedure in the nearby locations. The obvious drawback of the redundancy is an additional computational overhead, caused by the increasing amount of part realizations, however, the experiments presented in this thesis shows that redundancy can significantly increase the object categorization rate, since it increases the chances to find the correct solution through multiple alternative inference paths.

4.2 Learning and Inference Algorithms

This section provides an overview of the learning and inference algorithms used in both the compositional hierarchical frameworks presented in this thesis, i.e. of the view-based (Chapter 5) and the surface-based (Chapter 6) frameworks.

4.2.1 Learning and Inference Algorithms for Multiple Layers

This subsection outlines the algorithm for learning of the hierarchical compositional shape vocabulary of all layers $L_n, (n \geqslant 2)$ from a training dataset, and the algorithm for inference of parts of all layers $L_n, (n \geqslant 1)$ from a given dataset[1]. For describing steps of the algorithms the following notation is used:

$$Verbal\ description\ :\ \left[input_1, input_2, ..., input_l\right] \rightarrow \left[output_1, output_2, ...output_m\right],$$

meaning that this step takes l input arguments and produces m outputs. If there is only one input argument or only one output then the brackets are not used. The learning algorithm is presented in Algorithm 1, while the inference algorithm is described in Algorithm 2.

Algorithm 1: Vocabulary learning algorithm for multiple layers

Data: Training dataset $\mathcal{T}$
Pre-defined vocabulary of the first layer $\mathcal{S}(L_1)$
A set of composition rules $\mathcal{B}$
Result: Vocabulary $\{\mathcal{S}(L_n)\}_{2 \leq n \leq \mathcal{N}}$ of layers from L_2 to $L_{\mathcal{N}}$

1 Pre-process the training dataset: $\mathcal{T} \rightarrow \mathcal{T}_p$;
2 Perform **inference** of L_1 parts: $\left[\mathcal{T}_p, \mathcal{S}(L_1)\right] \rightarrow \mathcal{T}_{\mathcal{S}(L_1)}$;
3 **for** $n = 2$ **to** $\mathcal{N}$ **do**
4 | **Learn** vocabulary of the layer L_n : $\left[\mathcal{T}_{\mathcal{S}(L_{n-1})}, \mathcal{B}\right] \rightarrow \mathcal{S}(L_n)$;
5 | Perform **inference** of the layer L_n: $\left[\mathcal{T}_{\mathcal{S}(L_{n-1})}, \mathcal{S}(L_n), \mathcal{B}\right] \rightarrow \mathcal{T}_{\mathcal{S}(L_n)}$;
6 | Perform **pooling**: $\mathcal{T}_{\mathcal{S}(L_n)} \rightarrow \mathcal{T}_{\mathcal{S}(L_n)}$;
7 **end**

Both presented algorithms have three inputs. The first input is a dataset $\mathcal{T}$, which is a tuple (of length k) of 3D shape models, represented as depth images, point clouds or triangulated mesh models. We denote i-th shape model as $\mathcal{T}(i)$. The second input is a shape vocabulary. The learning algorithm takes only a pre-defined vocabulary of the first layer, while the inference algorithm takes the vocabulary of all layers as an input. The third parameter taken by both algorithms is a pre-defined set of composition rules

[1] which may be a testing set, where only inference of all layers if required

Algorithm 2: Algorithm for inference of multiple layers

Data: Dataset $\mathcal{T}$
Vocabulary $\{\mathcal{S}(L_n)\}_{1 \leq n \leq \mathcal{N}}$ of layers from L_1 to $L_{\mathcal{N}}$
A set of composition rules $\mathcal{B}$

Result: The dataset represented in terms the shape vocabulary $\{\mathcal{T}_{\mathcal{S}(L_n)}\}_{1 \leq n \leq \mathcal{N}}$

1 Pre-process the dataset: $\mathcal{T} \rightarrow \mathcal{T}_p$;
2 Perform **inference** of L_1 parts: $\left[\mathcal{T}_p, \mathcal{S}(L_1)\right] \rightarrow \mathcal{T}_{\mathcal{S}(L_1)}$;
3 **for** $n = 2$ **to** $\mathcal{N}$ **do**
4 | Perform **inference** of the layer L_n: $\left[\mathcal{T}_{\mathcal{S}(L_{n-1})}, \mathcal{S}(L_n), \mathcal{B}\right] \rightarrow \mathcal{T}_{\mathcal{S}(L_n)}$;
5 | Perform **pooling**: $\mathcal{T}_{\mathcal{S}(L_n)} \rightarrow \mathcal{T}_{\mathcal{S}(L_n)}$ (optional);
6 **end**

$\mathcal{B}$ as an input. The composition rules define legitimate bindings of subparts into parts of the next layer. Actually, the choice of the first layer vocabulary and the choice of composition rules eventually define the set of shapes which are learnable and recognizable by a compositional hierarchical system.

The described learning algorithm comprises five sub-algorithms, while the inference algorithm consists of four sub-algorithms, as visualized in Figure 4.1. Note, that four out of five sub-algorithms of both algorithms are equivalent.

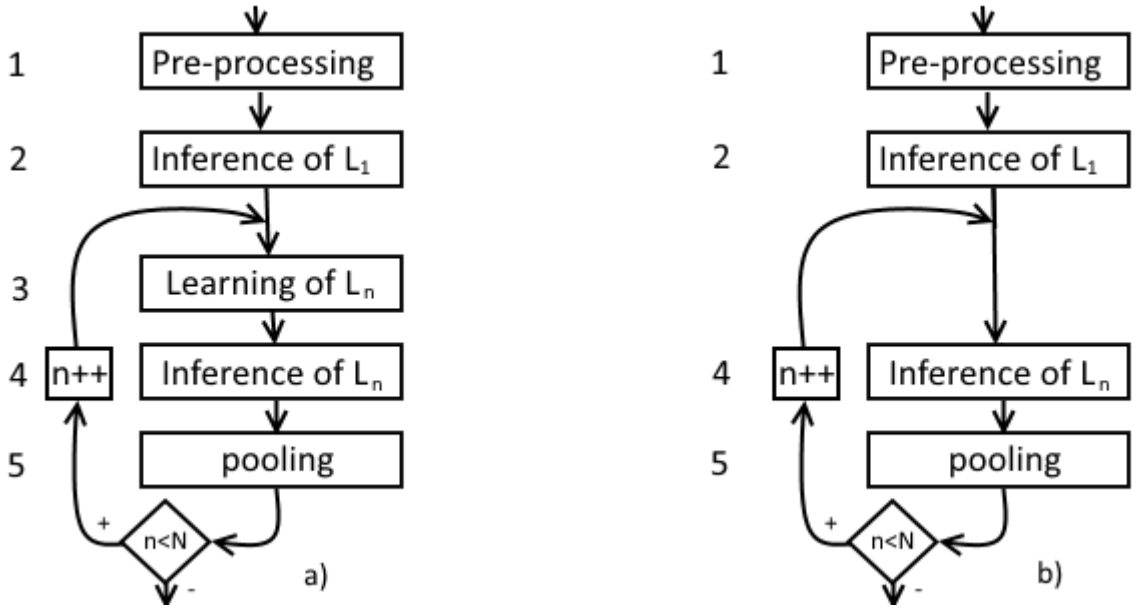


Figure 4.1: Scheme of the learning and inference algorithms. a) Learning of multiple layers (Algorithm 1). b) Inference of multiple layers (Algorithm 2). Both algorithms comprise 4 equivalent sub-algorithms.

Both learning and inference algorithms first perform *pre-processing* of the input data. The pre-processed dataset is denoted as $\mathcal{T}_p$, while the i-th pre-processed shape model is denoted as $\mathcal{T}_p(i)$. After pre-processing of the input data both algorithms perform *inference of the first layer* parts from the pre-processed data. To denote the results of the inference procedure of a layer $L_n, (n \geqslant 1)$ we use the notation $\mathcal{T}_{\mathcal{S}(L_n)}$, meaning "the dataset $\mathcal{T}$ expressed in terms of the shape vocabulary $\mathcal{S}(L_n)$". More precisely, $\mathcal{T}_{\mathcal{S}(L_n)}$ is a list of length k, where i-th element of this list is a set of part realizations of the layer L_n found in i-th shape model.

After inference of the first layer parts, the learning algorithm (Algorithm 1) performs three steps for each subsequent layer: it learns the vocabulary of this layer, infers the learned vocabulary from the training data, and performs pooling. In contrast, the inference algorithm (Algorithm 2) performs only inference and pooling for each subsequent layer.

Note, that the pre-processing algorithm (blocks 1 in Figure 4.1), the algorithm for inference of the first layer parts (blocks 2), the algorithm for inference of a single layer (block 4) and the pooling algorithm (blocks 5) are different for both compositional hierarchies, therefore, they will be described in details in the corresponding chapters. However, the learning algorithm for a single layer (block 4), has the same pipeline for both compositional hierarchies, that is why this pipeline is presented in the next subsections in order to avoid repetition in the next two chapters.

4.2.2 Vocabulary Learning Pipeline for a Single Layer

In Subsection 4.2.1 the outlines of the learning and inference algorithms for multiple layers were presented. This subsection describes the vocabulary learning procedure for a single layer $L_n, (n \geqslant 2)$, i.e. actually, this subsection expands the line 4 of Algorithm 1. The high-level overview of the learning algorithm for a single layer is presented in Algorithm 3.

The vocabulary learning algorithm for a single layer $L_n, (n \geqslant 2)$ involves several steps.

Algorithm 3: Vocabulary learning algorithm for a single layer $L_n, (n \geqslant 2)$

Data: Training set $\mathcal{T}_{\mathcal{S}(L_{n-1})}$ represented in terms of the vocabulary $\mathcal{S}(L_{n-1})$
Set of composition rules $\mathcal{B}$

Result: Vocabulary $\mathcal{S}(L_n)$ of the layer L_n

1 Collect statistical maps using co-occurrence statistics: $\left[\mathcal{T}_{\mathcal{S}(L_{n-1})}, \mathcal{B}\right] \rightarrow \{\mathcal{M}_{i,j}^{n-1}\}_{i,j}$;

2 Find and parameterize clusters in statistical maps: $\{\mathcal{M}_{i,j}^{n-1}\}_{i,j} \rightarrow \{\Delta_t^{sp}\}_t$;

3 Form a set $\mathcal{D}(L_n)$ of doublets of the layer L_n: $\{\Delta_t^{sp}\}_t \rightarrow \mathcal{D}(L_n)$;

4 Perform inference of doublets from the training set: $\left[\mathcal{T}_{\mathcal{S}(L_{n-1})}, \mathcal{D}(L_n), \mathcal{B}\right] \rightarrow \mathcal{T}_{\mathcal{D}(L_n)}$;

5 Form a set of candidate parts $\mathcal{C}(L_n)$ of the layer L_n: $\mathcal{T}_{\mathcal{D}(L_n)} \rightarrow \mathcal{C}(L_n)$;

6 Perform part selection: $\mathcal{C}(L_n) \rightarrow \mathcal{S}(L_n)$

At the first step, the statistics of co-occurrences of the L_{n-1} part realizations in the training data is collected and represented in a form of *statistical maps*. Statistical maps are built for each pair of parts of the previous layer, i.e. if $\mid \mathcal{S}(L_{n-1}) \mid = x$, then x^2 statistical maps are built when learning the layer L_n. A statistical map $\mathcal{M}_{i,j}^{n-1}$ is a function that approximates the probability to observe activations of the part P_j^{n-1} at a certain relative position[1] from activations of the part P_i^{n-1} . After statistical maps are built on basis of the co-occurrence statistics, they are clustered such that cluster centres represent the most frequently observed (in the training data) spatial configurations of part realizations of the layer L_{n-1}. After that, these clusters are parameterized, and a set of parameterized distributions is $\{\Delta_t^{sp}\}_t$ is built. Note that this set contains the parameterized distributions obtained from clustering of all statistical maps $\{\mathcal{M}_{i,j}^{n-1}\}_{i,j}$.

Each parameterized distribution allows making a doublet of the layer L_n. For instance, if a parameterized distribution Δ_m^{sp} is received from clustering of the statistical map $\mathcal{M}_{i,j}$, the resulting doublet will be $\pi_m^n = \left(P_i^{n-1}, (P_j^{n-1}, \Delta_m^{sp})\right)$. This notation means that the doublet, π_m^n contains the central subpart P_i^{n-1} and the subpart P_j^{n-1} at the relative position described by the distribution Δ_m^{sp} from the central subparts. When a set of doublets $\mathcal{D}(L_n)$ is complete, doublets are inferred from the training data. After that, the statistics of co-activations of doublets within the same receptive field are collected, and

[1] and certain relative orientation in case of the surface-based compositional hierarchy

the most frequently observed co-activations of doublets are used to form a set of *candidate parts* $\mathcal{C}(L_n)$ of the layer L_n. The set of candidate parts is typically very large, that is why it is compressed by the part selection procedure, in which only some of the candidate parts, chosen according different importance measures, are finally included to the $\mathcal{S}(L_n)$. Part selection concludes the learning procedure, and the selected subset $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$ forms the vocabulary of the layer L_n, which is the output of the algorithm.

CHAPTER 5

A VIEW-BASED COMPOSITIONAL HIERARCHY

5.1 Introduction

This chapter describes the view-based hierarchical compositional representation of surface shapes and the framework for learning and inference of the view-based compositional hierarchical shape vocabulary. The chapter also presents the results of the experimental evaluation of this framework.

In Chapter 4 we provided a high-level description of the learning and inference pipelines used in both the compositional hierarchies proposed in this thesis, i.e. the view-based compositional hierarchy presented in this chapter and the surface-based one, described in Chapter 6. However, most of the details, for instance, the description of the sub-algorithms and the equations, were not presented there, since these details are different for both compositional hierarchies. This chapter provides a detailed description of all algorithms of the view-based compositional hierarchical framework.

Note that some of the algorithms presented in this chapter are to a certain extent similar to the algorithms used in the compositional hierarchical system of Fidler *et al.* [42], from which we borrowed some ideas. The major similarity is that Fidler *et. al* also use vocabulary learning based on clustering of statistical maps, and two-pass learning for each layer, i.e. in the first pass they learn doublets, and then build the vocabulary using the statistics of co-activations of doublets. However, except for these similarities,

our algorithms are different from the algorithms of Fidler *et. al.* Also notice that since their system is designed for 2D contour features, while our frameworks works with 3D data, even conceptually similar algorithms would differ on the level of details[1]. Figure 5.1 shows some parts of different layers of the view-based hierarchy.

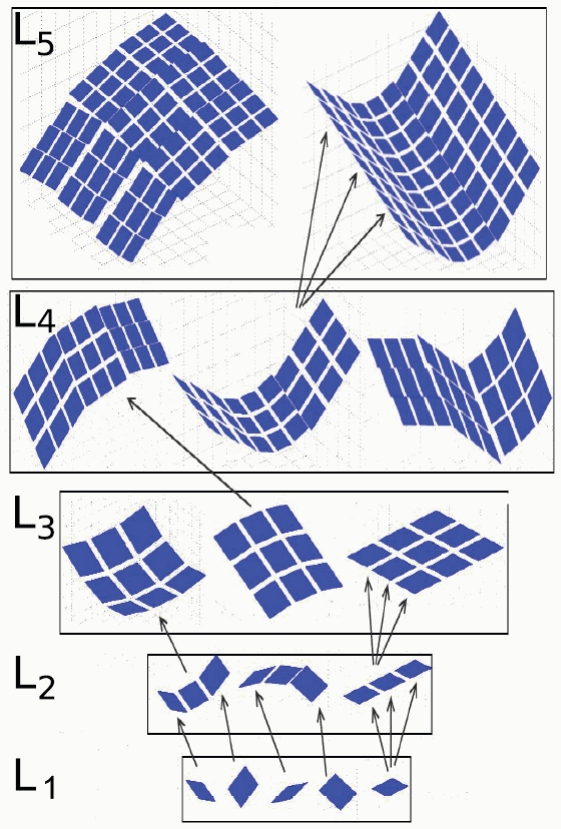


Figure 5.1: Several part of different layers of the view-based hierarchy

5.1.1 Camera-Based Reference Frame

The main property of the view-based representation is that **each vocabulary part represents a shape model of a certain orientation** relative to the camera-based reference frame. The camera-based reference frame is defined as illustrated in Figure 5.2. The Z axis, which is the camera's optical axis, points from the camera origin through the centre of the image plane towards the scene. Axes X and Y (up-vector) are parallel to the coor-

[1]Also note that we did not use any program codes of Fidler *et al.* and the system presented in this chapter has been developed from scratch

dinate axes of the image based reference frame. The origin of the image based reference frame is in the centre of the image.

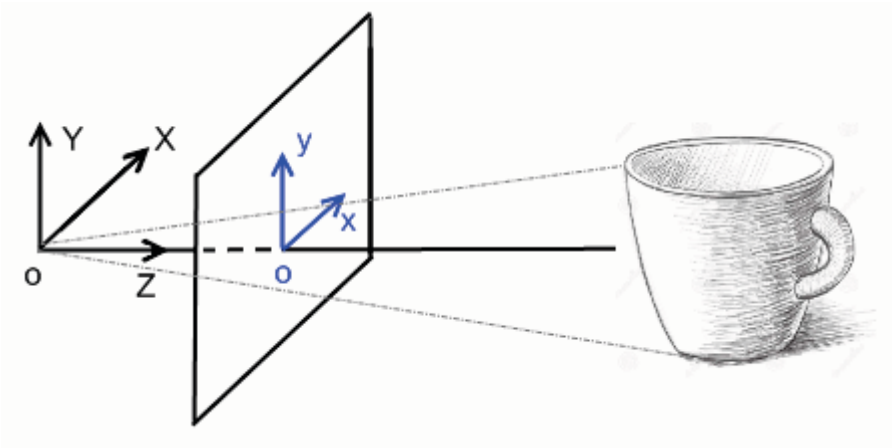


Figure 5.2: Camera based frame of reference (black) and image frame of reference (blue)

5.1.2 Description of the Input Data

The view-based compositional hierarchical system learns the vocabulary from a dataset of range images. All experiments described in this chapter were performed using the Washington RGB-D dataset [48]. The dataset comprises RGB and range images of 300 objects split into 51 categories. Multiple views (around 200) of each object are presented in the dataset. The input images are segmented, i.e. masks separating objects from the background are provided for each image. Some models from the Washington RGB-D dataset are shown in Figure 5.3.

Notice, that since this thesis proposes a representation of surface shape features, we use only the depth channel and do not use the RGB data.

The rest of the chapter is organized as follows. Section 5.2 outlines the pre-processing of the input data, while Section 5.3 describes the first layer of the compositional hierarchy, and explains how to infer the first layer parts from the pre-processed input data. Section 5.4 introduces the composition rules that are used on different layers of the compositional hierarchy. Section 5.5 describes all steps of the vocabulary learning procedure for a single layer $L_n, (n \geqslant 2)$, except for the part selection step, which is discussed in a separate section

Figure 5.3: Some shape models from the Washington RGB-D dataset. Picture source: the author's website https://rgbd-dataset.cs.washington.edu/

(Section 5.6). Section 5.7 describes inference from a single layer $L_n, (n \geqslant 2)$, while Section 6.9 describes the pooling algorithm. Section 5.9 presents our approach to object categorization from range images based on the compositional hierarchical shape vocabulary. Finally, Section 5.10 presents the result of experimental evaluation of the system.

5.2 Pre-Processing of the Input Data

Before undergoing learning and inference the input data needs pre-processing due to the following reasons. First, it is necessary to minimize the influence of noise and other data degradations, which are unavoidable during the image acquisition process. Second, since depth images typically represent shapes under a certain projection[1], the reconstruction of the original shapes is required. Note, that since the properties of the data, such as sampling, quantization, level of noise, type of projection, camera settings, etc., vary significantly across different datasets, there are no universally suitable procedures and parameters for pre-processing. That is why this section describes only the most frequently

[1] typically the perspective projection

required pre-processing steps and recommends the suitable parameters[1].

Assume the input data is represented by a list of range images $\mathcal{T}$. At the first stage we **pre-smooth** each input image $I = \mathcal{T}(i)$, $(i \in \mathbb{N})$ with a two-dimensional Gaussian kernel K_σ, i.e. compute $I_\sigma = K_\sigma * I$, where the parameter σ is chosen from the range from 0.5 to 2.0 depending on the noise level. If the input data is presented as a set of RGB-D images of textureless objects, the pre-smoothing of the depth channel can be done in a more sophisticated way, using the *guided image filtering* [210], where different filtering kernels for different locations of the depth image are used. These kernels are computed using the corresponding areas of the RGB image (which serves as the *guidance image*) in order to facilitate smoothing within flat regions and prevent smoothing across image edges. The parameter of the guided image filter r_g should be chosen from the range from 4 to 8, while the parameter ϵ_g can be from the range from 0.2^2 to 0.4^2.

On the second stage we convert each pre-smoothed image I_σ to a point cloud $\mathcal{P}$, i.e. for each pixel $\rho = (x, y) \in \Omega$ from the image domain $\Omega \subset \mathbb{Z}^2$ we compute the corresponding 3D point $p = (p_x, p_y, p_z)^T$. This pre-processing step is done to minimize influence of the projection and to enable the learning and inference algorithm to deal with the original 3D shape not distorted by the projection. If the image size is $w \times h$ pixels, and the camera's horizontal and vertical fields of view and are FoV_h and FoV_v, then this conversion can be done using the following equations:

$$p_x = 2\, I_\sigma(\rho) \left(\frac{x}{w} - 0.5 \right) \tan\left(\frac{FoV_h}{2} \right), \tag{5.1}$$

$$p_y = 2\, I_\sigma(\rho) \left(\frac{y}{h} - 0.5 \right) \tan\left(\frac{FoV_v}{2} \right), \tag{5.2}$$

$$p_z = I_\sigma(\rho). \tag{5.3}$$

After this conversion, each resulting point cloud $\mathcal{P}$ is included to the pre-processed dataset $\mathcal{T}_p$. Note, that after pre-processing both learning and inference algorithms do not

[1] having emphasized, that some additional or alternative pre-processing steps and parameters may be needed.

deal with depth images any more, and all subsequent computations are performed using these point clouds.

5.3 First Layer

In Chapter 4 we explained the decision to use small planar surface patches as the vocabulary parts of the first layer of both view-based and surface-based compositional hierarchies. Since in the view-based compositional hierarchy each part should represent a shape model of a certain orientations relative to the camera-based reference frame, the vocabulary of the first layer should comprise differently orientated planar surface patches.

5.3.1 Form of the First Layer Parts

It is important to make a decision about the form of the planar surface patches representing the first layer parts, i.e. decide whether they should be circular, square or having some other forms.

We considered two alternative ways of defining the form of the planar surface patches, representing the L_1 parts, and each of these ways has its own pros and cons. The first way is to define the L_1 vocabulary such that all parts represent differently orientated equally sized (in 3D space) planar surface patches. However, in this case the orthographic projections of these surface patches on the XY plane will have different forms and cover different areas. This can make the inference procedure more complicated, since in this case different vocabulary parts will have differently sized support regions on the XY plane. The second way is to define a set of differently orientated surface patches of different sizes, such that their orthographic projections on the XY plane are equally sized, and therefore cover equal areas, which can make the learning and inference procedures substantially easier. On this basis we decided to follow the second path and define the L_1 parts such that their orthographic projections on the XY plane are squares of the same size. Note, that such an approach is very similar to the CNN-based methods applied to depth images,

in which the image domain is tiled by the equally sized receptive fields of the square form.

5.3.2 Quantization of Surface Orientations

The next important task is to quantize orientations of planar surface patches to form the L_1 vocabulary. To this end we define $n_b \times n_b = n_b^2$ orientational bins, meaning that the vocabulary of the first layer contains n_b^2 parts. Notice, that for the experiments presented in this chapter we used $n_b = 9$, i.e. we defined 9×9 orientational bins. This number was chosen experimentally targeting to maximize the category recognition accuracy on the given dataset[1].

Assume we have a planar patch with the surface normal N of the unit length, while i and j are unit vectors pointing in the directions of the X and Y axes of the camera-based reference frame. Algorithm 4 describes how to compute the index of the vocabulary part $partID$, ranged from 1 to n_b^2, given the surface normal N. Figure 5.4 and 5.5 visualize the main steps of this algorithm. We project N to the XZ and YZ planes, obtaining vectors N_{xz} and N_{yz}. Then the angles $angleX$ and $angleY$ between these vectors and the vector $-Z$ (opposite to the viewing direction) are computed (Figure 5.4 (b) and (c)). Then each of these angles is assigned to one of n_b orientational bins $binX$ and $binY$ (as shown in Figure 5.5), and finally $partID$ is computed using indices $binX$ and $binY$.

Note, that if the absolute values of either $angleX$ or $angleY$ are larger than a predefined threshold $range$ the patch is not assigned to any bin, i.e. there are no inferred realizations of L_1 parts at this point. This is done because in this case surface becomes poorly visible (since it is "almost parallel" to the viewing direction), and therefore it becomes very hard or impossible to reliably estimate the surface properties, for instance to perform the planarity test or to compute a surface normal. Therefore, this parameter is data-dependent, and should be selected taking into account the question whether the surface properties (in particular, the planarity, and the orientation of surface normals) can be reliably estimated. For the experiments presented in this chapter we used the

[1]Also the settings with 5×5, 7×7, and 11×11 orientational bins were evaluated

Algorithm 4: Compute the part id of a given planar surface patch

Data: Planar surface patch with normal of the unit length N
Threshold value $range$
Number of orientational bins n_b

Result: The index $partId$ of the bin the surface patch belongs to

$angleX = 90° - \arccos(N^T i)$;
$angleY = 90° - \arccos(N^T j)$;
if $\mid angleX \mid < range$ ***and*** $\mid angleY \mid < range$ **then**
  $binX = round\left(\frac{n_b\ angleX}{2\ range}\right) + ceil(\frac{n_b}{2})$;
  $binY = round\left(\frac{n_b\ angleY}{2\ range}\right) + ceil(\frac{n_b}{2})$;
  $partId = n_b\ binX + binY$;
else
  $partId = 0$;
end

value $range = 70°$.

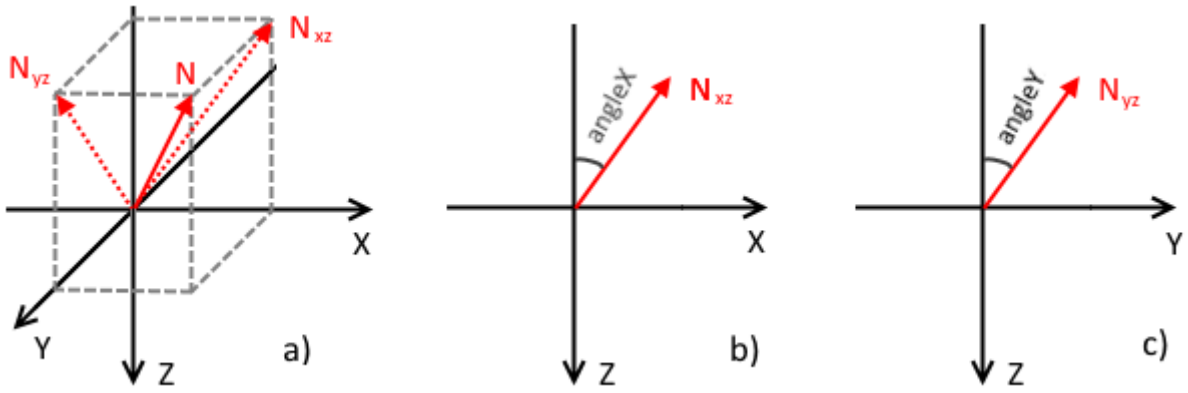


Figure 5.4: Illustration for the Algorithm 4. a) Surface normal translated to the origin of the camera-based reference frame, b) $angleX$, c) $angleY$

5.3.3 Inference of the First Layer

In this subsection the inference of the first layer parts from the input point cloud is described. In Subsection 5.3.1 we decided to define the first layer parts such that their orthographic projections on the XY plane are equally sized squares. This choice makes the inference procedure straightforward, since all L_1 parts have the same support regions on the XY plane in this case.

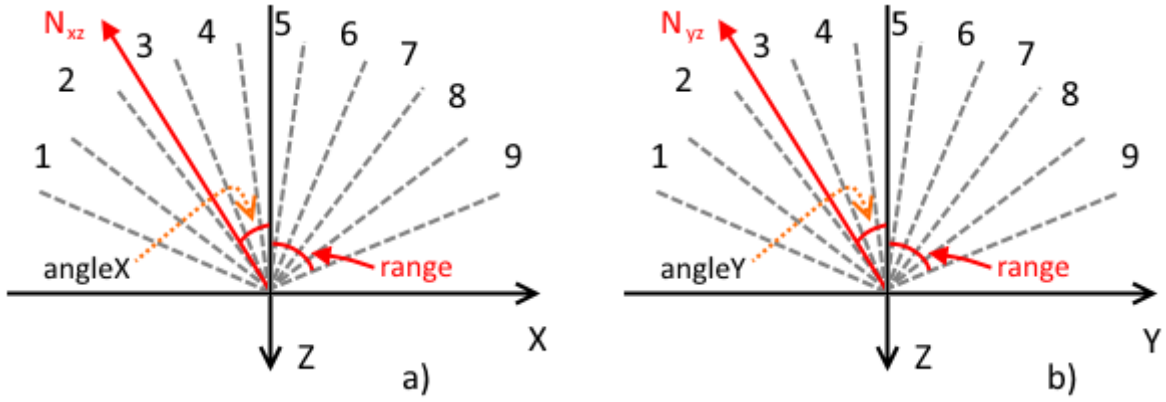


Figure 5.5: Illustration for the Algorithm 4. a) $angleX$ is assigned to one of 9 orientational bins (bin 3) b) $angleY$ is assigned to the bin 3 as well

Assume the support region (on the XY plane) of each first layer part is a square with side $2r$. Then to perform inference of a first layer part from a point cloud, the following three main steps should be performed: (i) extract points of the point cloud that belong to a support region, (ii) perform the planarity test of these points, (iii) if planarity is confirmed, compute the surface normal and assign it to one of the 81 bins according to its orientation, as shown in Algorithm 4.

The full algorithm for inference of the first layer parts is presented in Algorithm 5. The input data for this algorithm comprises a list of m point clouds $\mathcal{T}_p$ representing range scans of the objects, while the output $\mathcal{T}_{\mathcal{S}(L_1)}$ is the list (of length m) of sets, each of which contains all part realizations of the layer L_1 detected in the corresponding point cloud.

We use one of the following two ways of performing planarity tests given a set of points Λ. If the input data is not very noisy, we perform **least squares fitting** which works well in most of the cases, otherwise we apply the classical RANSAC algorithm [211]. We do not provide formulas for both of these methods in this thesis, since the list squares fitting is described in many textbooks, while application of RANSAC to plane fitting in depth images and point clouds is a commonly used technique, also described in many sources [212, 213].

Algorithm 5: Inference of the L_1 parts

Data: List of m point clouds $\mathcal{T}_p$
Vocabulary of first layer $\mathcal{S}(L_1)$
Result: Input data represented in terms of the vocabulary of the first layer $\mathcal{T}_{\mathcal{S}(L_1)}$

for $j = 1$ **to** m **do** `// for each point cloud`
  $\mathcal{P}_j = \mathcal{T}_p(j)$;
  $\mathcal{Q} = \emptyset$;
  foreach $(p_x, p_y, p_z) \in \mathcal{P}_j$ **do** `// for each point`
    find a set $\Lambda \subset \mathcal{P}_j$ with coordinates $x \in [p_x - r, p_x + r]$ and $y \in [p_y - r, p_y + r]$;
    perform **planarity test** for the set Λ;
    if *planarity is confirmed* **then**
      compute surface normal N for this plane;
      compute $partId$ (Algorithm 4);
      create a realization R:
        $R.layer = 1$; $R.id = partId$; $R.coord = (p_x, p_y, p_z)$;
      include R to the set $\mathcal{Q}$;
    end
  end
  $\mathcal{T}_{\mathcal{S}(L_1)}(j) = \mathcal{Q}$
end

5.4 Composition Rules

A set of composition rules is one of the most important ingredients of each compositional hierarchical system, since the composition rules dictate legitimate bindings of subparts, and therefore define which configurations of subparts may or may not appear on different layers. Thus the set of the first layer features and the set of composition rules eventually define a range of learnable and recognizable shapes. In this section the full set of composition rules (denoted $\mathcal{B}$) for the view-based compositional hierarchy is formulated.

Note, that composition rules only define whether or not subparts in a certain spatial configuration are eligible for compositing, while this eligibility does not guarantee that these subparts will be composed in the learning and/or inference phases. Therefore, the eligibility of certain spatial configurations of subparts for compositing should be considered as a necessary but not sufficient condition for making compositions.

Let us start from considering the example shown in Figure 5.6(a). It shows a point

cloud $\mathcal{P}$ (blue points) representing a planar surface patch of the rectangular shape. Assume the support regions of the first layer parts on the XY plane are squares of size $2r$, as shown in 5.6(b). Note, that the average distance between nearby points of the point cloud is significantly smaller than r, therefore the inference algorithm of the first layer parts (Algorithm 5) will produce realizations of L_1 parts with strongly overlapped support regions[1]. For the given example, it will produce realizations of the first layer parts in each point of the point cloud. In this situation, it becomes important to define the rules specifying which configurations of L_1 part realizations are eligible for compositing on the next layer and these rules should mainly be purposed to avoid the combinatorial explosion.

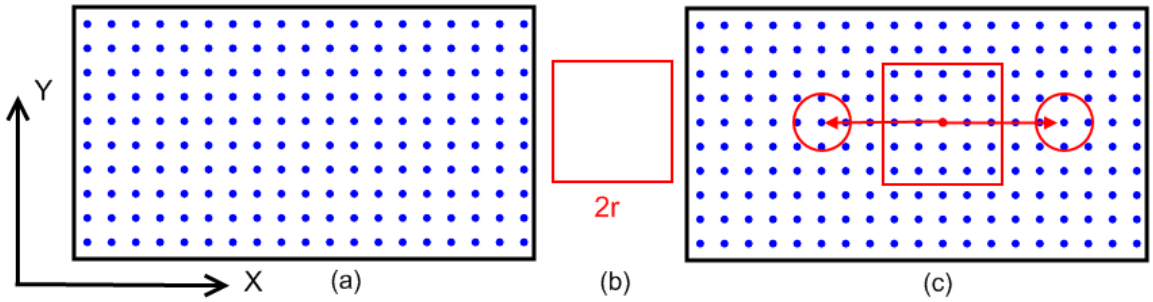


Figure 5.6: Composition rule for the layer L_2. a) A point cloud representing the planar surface patch, b) size support region of the L_1 parts, c) part realizations located within the red circles are eligible for composing with the central part realization (shown by the red dot).

The proposed composition rules can be described as follows:

1. Each part of the layer $L_n, (n \geqslant 2)$ should be composed of three subparts, i.e. the central subpart and two other subparts, located on different sides from the central subpart,

2. Subparts should be composed using the following rule: On the even layers of the hierarchy (e.g. L_2, L_4) subparts adjacent in the X direction should be composed, while on the odd layers subparts adjacent in the Y direction are composed,

[1]Remember, that the existence of part realizations with the overlapped support regions is an important property of the proposed compositional hierarchical system, as was explained in Section 4.1.3

3. Offsets and overlaps between the support regions of subparts in XY plane should be minimized,

4. The existence of **empty subparts** (empty cells) indicating the absence of surface in the area adjacent to the central subpart is permitted,

5. Each compositional part can contain only one empty subpart. That means each part of the layer $L_n, (n \geqslant 2)$ must contain a non-empty central subpart, and either two other non-empty subparts or one non-empty and one empty subpart.

Since these rules require some explanation, let us consider the example shown in Figure 5.6(c). It shows how the composition rules are applied when composing the parts of the layer L_1 to parts of the layer L_2. Since L_2 is the even layer, we search subparts on the left and the right side from the central subpart, i.e. on different sides from the central subpart in the direction of the X axes. Note, that for the odd layers the subparts on the top and on the bottom from the central subpart, i.e. in the direction of Y axes, should be considered. Subparts should be located approximately at the distance $\delta = 2r$ from the central subpart[1] since this distance minimizes overlaps between the support regions and prevents the existence of large gaps between them. Thus δ defines the preferable offset between the subparts. If the central subpart has coordinates (x, y), then the preferable positions of other subparts will be $(x - \delta, y)$ and $(x + \delta, y)$. However, since it may happen that there are no part realizations exactly at the preferable positions, relatively small circular neighbourhoods around preferable positions should be considered. Radii of these neighbourhoods are computed as 0.25τ, where τ is the smallest dimension of the support region of the central subpart ($0.5r$ in the given example)[2]. These neighbourhoods are shown as red circles in Figure 5.6(c), and the realizations located in these neighbourhoods are eligible for composing with the central subpart on the next layer.

[1] distances are measured in XY plane

[2] The choice of this parameters depends on the density of an input point cloud. The parameter should be approximately chosen such that at least several data points (ideally around 3 - 5 points) lie within the circle. This is purposed to provide the inference algorithm with several alternative options.

Figure 5.7 shows some examples of L_2 parts that are learned under the set of composition rules presented above, while Figure 5.8 depicts examples of L_3 parts of composed under the same set of rules.

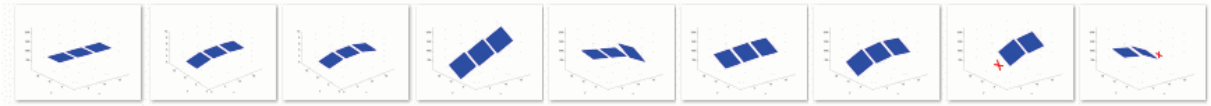


Figure 5.7: Parts of the layer L_2 learned under composition rules $\mathcal{B}$. Empty cells are shown with red crosses

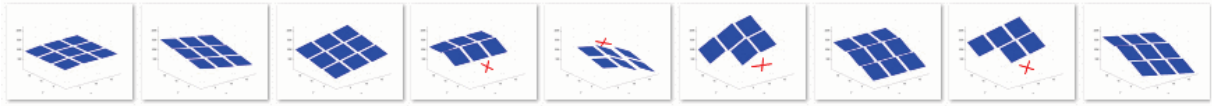


Figure 5.8: Parts of the layer L_3 learned under composition rules $\mathcal{B}$. Empty cells are shown with red dots

5.4.1 Empty subparts

An *empty cell* is an abstraction used in the view-based compositional hierarchy to indicate the *absence of surface* in a certain location. Suppose there is a point cloud representing a planar surface of a circular form, as shown in Figure 5.9(a). As in the example described in the previous chapter, the inference algorithm will produce L_1 part realizations with strongly overlapped support regions in the XY plane. Assume, the size of these support regions is $2r$, as shown as a red square in Figure 5.9(b). Also assume there is a part realization R_i located in the point A with coordinates (x, y), and the goal is to define the part realizations eligible for composing with R_i on the next layer. To this end the preferable offset $\delta = 2r$, and the preferable positions of the subparts $(x-\delta, y)$ and $(x+\delta, y)$ are defined, in the same way as it was shown in the previous example. However, as we can see from Figure 5.9(a) there is no surface in the neighbourhood of the point $(x-\delta, y)$. In such situations, the subpart located in the point A (central subpart) and the subparts located in the neighbourhood of the point $(x+\delta, y)$ can still be composed to a L_2 part, but

description of this part must contain a certain indication, showing the absence of surface around the point $(x - \delta, y)$. Empty subparts serve as indicators of that type.

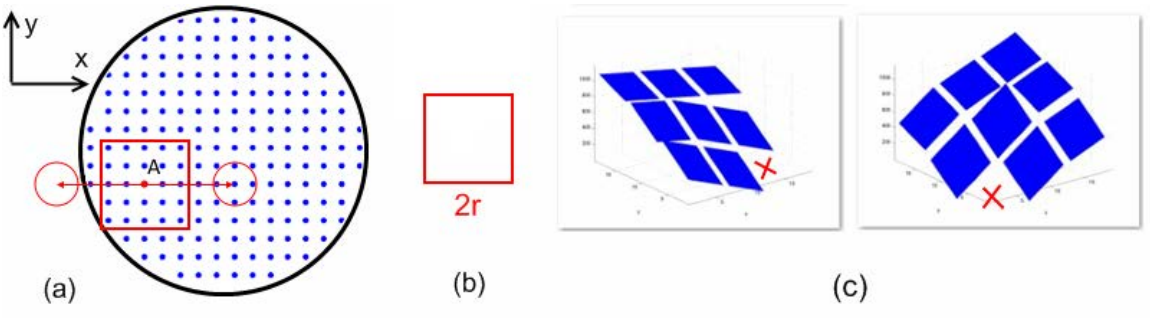


Figure 5.9: (a) Point cloud of the circular planar surface, (b) Size of the support region of L_1 parts in the XY plane, (c) Examples of L_3 parts comprising L_2 parts with empty cells

It is important to emphasize that empty subparts show the **absence of surface, but not the absence of part realizations** in a certain area. For instance, if a surface exists in a certain in a certain area, i.e. there are some points of the input point clouds, but there are no part realizations in this area, then the empty subparts are not used.

When processing the point clouds of single objects, the absence of surface can be verified by checking the existence of points in a neighbourhood. However, when processing point clouds of scenes, by the absence of surface we actually mean "the absence of surface within a certain range of depths". That mean another surface may exist in the background, but if its relative depth w.r.t. the central subpart is larger than a pre-defined threshold, this is interpreted as the absence of surface, and empty subparts can be used in such cases.

5.5 Vocabulary Learning

This section details the vocabulary learning algorithm described for a single layer L_n, $(n \geqslant 2)$. A high-level description of this algorithm was provided in Section 4.2.2, while the purpose of this section is to provide a detailed description of each step.

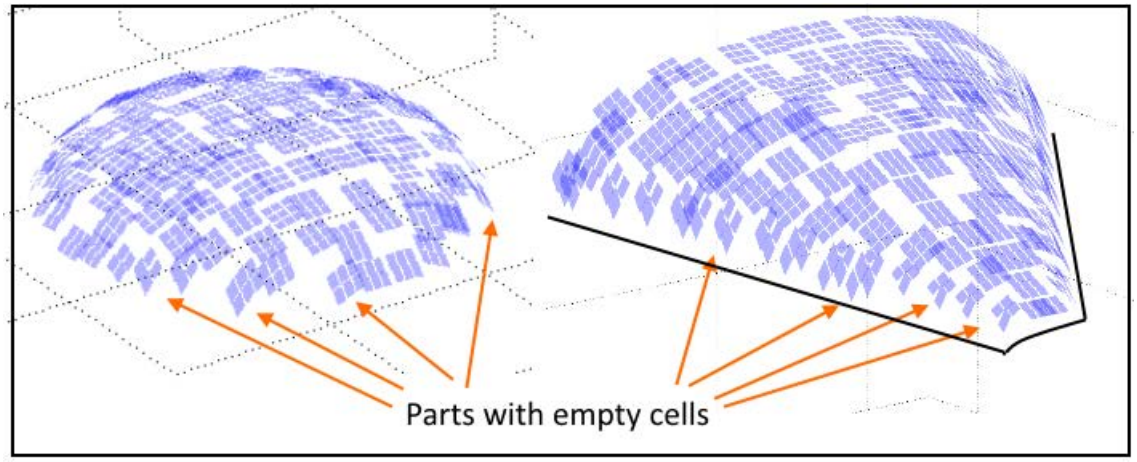


Figure 5.10: Parts with empty subparts typically appear in the locations where the surface ends of occludes itself

5.5.1 Collecting Co-Occurrence Statistics

The first step of learning the shape vocabulary $\mathcal{S}(L_n)$ for a single layer $L_n, (n \geqslant 2)$ is collecting the statistics of co-occurrences of part realizations of the previous layer L_{n-1} and representing this statistics in the form of statistical maps.

Statistical map $\mathcal{M}^n_{i,j} : \mathbb{Z}^3 \to \mathbb{N}$ is a function showing how many times realizations of the part P^n_j are observed in the training data at the certain relative position w.r.t. realizations of the part P^n_i. For instance, the equation $\mathcal{M}^n_{i,j}(x, y, z) = 15$ means that realizations of the part P^n_j appeared 15 times at the relative position $s_t(x, y, z)$ relative to realizations of the part P^n_i. Since the statistical maps are discrete functions, the parameter s_t should be used to define the discretization. Typically, we make statistical maps of size $25 \times 25 \times 25$, in which case s_t should be computed such that all relative coordinates are mapped to the integers from the range [-12, 12]. Note, that each statistical map only depicts co-occurrence statistics of spatial configurations that are eligible for composing under set of composition rules $\mathcal{B}$.

When learning the vocabulary of the layer L_n the statistical maps for each pair of parts from the previous layer L_{n-1} should be built, thus if $\mid \mathcal{S}(L_{n-1}) \mid = m$ then m^2 statistical maps should be built. Note, however, that most of the statistical maps remain empty during learning since many shapes never co-occur in real-world objects. Algorithm

6 describes the process of building statistical maps.

Algorithm 6: Collecting co-occurrence statistics and building statistical maps

Data: Training set $\mathcal{T}_{\mathcal{S}(L_{n-1})}$ of the length k represented in terms of the vocabulary $\mathcal{S}(L_{n-1})$
Set of composition rules $\mathcal{B}$
Size of the vocabulary of the layer L_{n-1}: $\mid \mathcal{S}(L_{n-1}) \mid = m$
Parameter for discretization of statistical maps s_t

Result: A set of statistical maps $\{\mathcal{M}_{i,j}^{n-1}\}_{1 \leq i,j \leq m}$

```
Make all statistical maps {M_{i,j}^{n-1}}_{1≤i,j≤m} empty;
for j = 1 to k do                                   // for each training model
    Q = T_{S(L_{n-1})}(j) ;                         // set of L_{n-1} realizations
    foreach R ∈ Q do                                // for each part realization
        find a set Λ ⊂ Q of part realizations eligible for composing with R under a
         set of composition rules B ;
        foreach K ∈ Λ do                            // for each realization
            (x, y, z) = K.coord − R.coord ;         // relative position
            x = round(x / s_t);
            y = round(y / s_t);
            z = round(z / s_t);
            M^{n-1}_{R.Id,K.Id}(x, y, z) = M^{n-1}_{R.Id,K.Id}(x, y, z) + 1;
        end
    end
end
```

5.5.2 Clustering of Statistical Maps and Parameterization of Clusters

After co-occurrence statistics is collected and statistical maps are built, we cluster each statistical map and fit the data in each cluster with the multivariate Gaussian distribution. Since the co-occurrence statistics are collected only for spatial configurations eligible for composing under the set of composition rules $\mathcal{B}$, statistical maps contain well separated clusters in the statistical maps, therefore the choice of a clustering algorithm does not make difference, and we use a very simple clustering algorithms, namely the weighted version of the *k-means* algorithms [214].

When clusters in statistical maps are found, we parameterize each cluster using maximum likelihood estimation (MLE), i.e. we estimate three-dimensional vector μ and 3×3

covariance matrix Σ for each cluster. Figure 5.11(a) shows the parameterized statistical map $\mathcal{M}^1_{41,41}$.

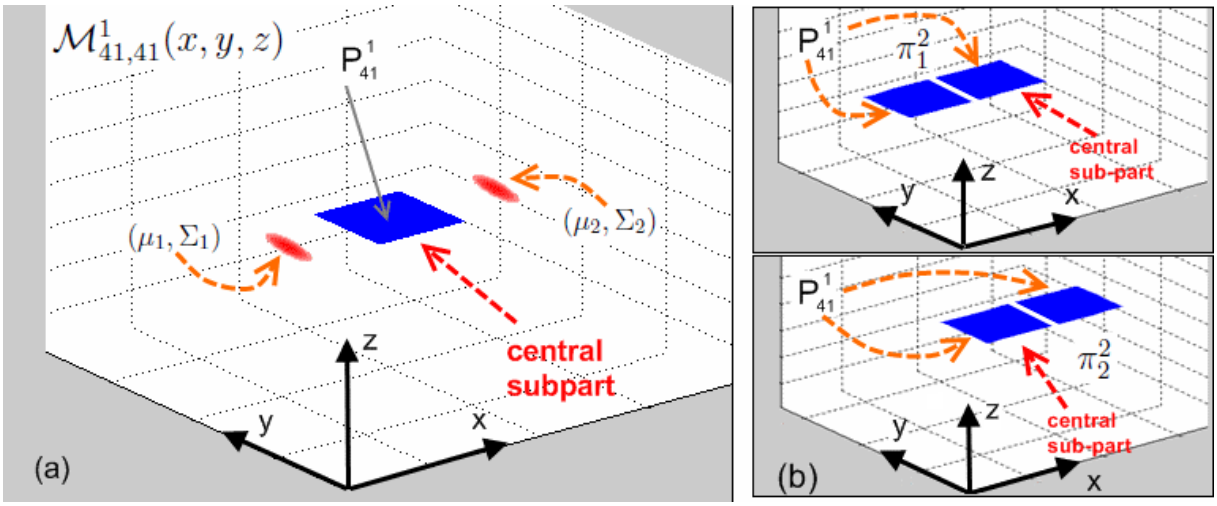


Figure 5.11: (a) Parameterized statistical map $\mathcal{M}^1_{41,41}(x,y,z)$ depicting co-occurrences of realizations of the part P^1_{41} (b) Doublets formed from this statistical map

After parameterizing statistical maps we form a tuple of doublets $\mathcal{D}(L_n)$ of the layer L_n. Each parameterized cluster in each statistical map results in a doublet. For instance, as the parameterized statistical map $\mathcal{M}^1_{41,41}$ shown in Figure 5.11(a) has two clusters, it will result in two doublets, shown in Figure 5.11(b) namely $\pi^2_1 = (P^1_{41}, (P^1_{41}, \Delta^{sp}_1))$, and $\pi^2_2 = (P^1_{41}, (P^1_{41}, \Delta^{sp}_2))$, where Δ^{sp} is a description of spatial relation of subparts, i.e. $\Delta^{sp}_1 = (\mu_1, \Sigma_1)$, and $\Delta^{sp}_2 = (\mu_2, \Sigma_2)$. If there are t parameterized clusters in all statistical maps, then the same number of doublets of the layer L_n will be formed, and therefore the length of the tuple $\mathcal{D}(L_n)$ will be equal to t.

The tuple of doublets $\mathcal{D}(L_n)$ learned so far does not yet represent the vocabulary of the layer L_n. To complete learning of the layer L_n it is necessary to collect the statistics showing how different doublets co-occur in the training data and build more complex configurations of subparts on basis of this statistics. Let us consider the example shown in Figure 5.11(b) once again. After clustering the statistical map and parameterizing the clusters we obtained two doublets π^2_1 and π^2_2 both having the central subpart P^1_{41} and another subpart P^1_{41} located on the left and on the right from the central part correspondingly.

Note, however, that it is not always the case, that these two doublets are activated simultaneously at the same data point. Activation of one of these doublets (say π_1^2) in a data point ρ tells us about surface properties around this point (central subpart) and on the left side from it, but it does not provide any information about the surface on the right side from ρ. Surface on the right side from ρ may have different properties, therefore different doublets may in principle be activated simultaneously with π_1^2 in ρ.

This suggests that the next stage of the learning process should involve collecting of the statistics of co-activations of doublets in the training data and compose doublets into more complex parts based on these statistics. This can be done in three steps: (i) perform inference of doublets from the training data (ii) measure frequencies of co-activations of different doublets in the training data, (iii) form the set of candidate parts $\mathcal{C}(L_n)$ of the layer L_n using frequently co-activated pairs of doublets. In the following subsubsections we provide more details about these steps.

5.5.3 Inference of Doublets from the Training Data

Assume the set of doublets $\mathcal{D}(L_n)$ contains a doublet $\pi_i^n = (P_c^{n-1}, (P_j^{n-1}, \Delta_i^{sp}))$, which includes the central subpart P_c^{n-1}, another subpart P_j^{n-1} and the description $\Delta_i^{sp} = (\mu_i, \Sigma_i)$ of the distribution of possible relative positions of P_j^{n-1} w.r.t. P_c^{n-1}. Also assume a realization of the part P_c^{n-1} is located in the point ρ_c, while a realization of the part P_j^{n-1} is located at the relative position $\delta = (x,\, y,\, z)^T$ w.r.t. ρ_c. We say, that the doublet π_i^n is activated in the point ρ_c **if and only if** δ belongs to the distribution described by Δ_i^{sp}. To test whether or not δ belongs to the given Gaussian distribution we use Mahalanobis distance [215]:

$$d_M(\delta, \mu_i) = \sqrt{(\delta - \mu_i)^T \Sigma_i^{-1} (\delta - \mu_i)} \tag{5.4}$$

If $d_M(\delta, \mu_i)$ is less than the threshold value (usually taken as 3.0, since "nearly all" values of the Gaussian distribution lie within three standard deviations from the mean), then we say that π_i^n is activated in the point ρ_c. The Equation 5.4 describes the **activation**

test which is the key part of the inference procedure. The algorithm for inference of doublets from the training data is described in Algorithm 7.

Algorithm 7: Inference of doublets of the layer $L_n, (n \geqslant 2)$

Data: Training data $\mathcal{T}_{\mathcal{S}(L_{n-1})}$ represented in terms of the vocabulary $\mathcal{S}(L_{n-1})$
Set of composition rules: $\mathcal{B}$
Doublets of the layer L_n : $\mathcal{D}(L_n)$

Result: Training data $\mathcal{T}_{\mathcal{D}(L_n)}$ represented in terms of the doublets $\mathcal{D}(L_n)$

$\mathcal{T}_{\mathcal{D}(L_n)}$ is empty;
for $j = 1$ **to** k **do** `// for each model`
  $\mathcal{Q} = \mathcal{T}_{\mathcal{S}(L_{n-1})}(j)$; `// set of` L_{n-1} `realizations`
  $\mathcal{DD}$ is empty ; `// set of` L_n `doublets activations`
  foreach $R \in \mathcal{Q}$ **do** `// for each part realization`
    find a set $\Lambda \subseteq \mathcal{Q}$ of part realizations eligible for composing with R under a set of composition rules $\mathcal{B}$;
    find all doublets $\mathcal{D}_s(L_n) \subseteq \mathcal{D}(L_n)$ having the central subpart $P^{n-1}_{R.id}$;
    foreach $K \in \Lambda$ **do**
      find all doublets $\mathcal{D}_{ss}(L_n) \subseteq \mathcal{D}_s(L_n)$ having the non-central subpart $P^{n-1}_{K.id}$;
      if $\mathcal{D}_{ss}(L_n) \neq \emptyset$ **then**
        $\delta = K.coord - R.coord$; `// relative coordinates`
        foreach $\pi^n_{cur} = (P^{n-1}_{R.id}, (P^{n-1}_{K.id}, \Delta^{sp}_{cur})) \in \mathcal{D}_{ss}(L_n)$ **do**
          Perform **activation test** to check if δ fits the distribution Δ^{sp}_{cur};
          if *activation test is successful* **then**
            $RR.id = cur$; `// create a realization of doublet` π^n_{cur}
            $RR.coord = R.coord$;
            include RR to the set $\mathcal{DD}$;
          end
        end
      end
    end
  end
  $\mathcal{T}_{\mathcal{D}(L_n)}(j) = \mathcal{DD}$
end

5.5.4 Co-activations of Doublets

After inference of doublets of the layer L_n from the training data is complete, we obtain activations of doublets in different data points. Obviously, some of the data points do not contain any activations, because the local surface structure in the neighborhood of such

points does not fit any templates represented by the set of doublets $\mathcal{D}(L_n)$. On the other hand, in some of the data points two or more doublets may be activated, i.e. there are co-activations of doublets in some points. In general, our goal is to learn the statistics of co-activations and form a set of candidate parts $\mathcal{C}(L_n)$ using these statistics.

There exist several possible cases depending on how many doublets are simultaneously activated in the same data point and what are the properties of these doublets. Assume there is a data point ρ and a part realization of the part P_c^{n-1} (of the previous layer L_{n-1}) activated in this point. The following items describe all possible cases:

1. If there are no doublets of the layer L_n activated in the point ρ then, obviously, there are no activations of candidate parts in this point.

2. If two doublets are activated in the data point ρ, for example, doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$, and these doublets describe surface in the opposite directions from ρ (e.g. on the left and on the right sides from ρ), then we say that there is an activation of a candidate part $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_k^{n-1}, \Delta_j^{sp}))$ residing in the point ρ (see the illustration in Figure 5.12 (a)).

3. If there are more than two doublets activated in the point ρ, for example, $m \geq 1$ doublets describing surface on one side from ρ, and $k \geq 1$ doublets describing the surface of the other side from ρ, where $max(k, m) > 1$, then we say there are activations of $m \times k$ candidate parts in the point ρ. Let us consider the example when $m = 2$ (doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$) and $k = 1$ (doublet $(P_c^{n-1}, (P_b^{n-1}, \Delta_a^{sp}))$). Then we have $m \times k = 2 \times 1 = 2$ activations of two candidate parts in the point ρ, i.e. $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$ (see the illustration in Figure 5.12 (b)).

4. If there is only one doublet $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ activated in the point ρ, then we need to perform an extra check to figure out whether or not surface in the direction opposite to the one described by Δ_i^{sp} exists. For instance, let us assume, that a

doublet describes a surface on the left side from ρ. Then we should check whether or not surface on the right side from the central subpart exists[1].

- If the surface exists, then there are no candidate parts of the layer L_n residing in the point ρ (see the illustration in Figure 5.12 (c)).
- If the surface does not exist, then there is a candidate part $(P_c^{n-1}, (P_l^{n-1}\Delta_i^{sp}))$ in the point ρ[2] (see the illustration in Figure 5.12 (d)).

5. If there are m doublets activated in the point ρ, where $m > 1$, and all these doublets describe surface in the same directions from ρ then we have to check the existence of surface in the opposite direction. If surface exists there, then there are no candidate parts of the layer L_n in the point ρ. If there is no surface there, then there are m candidate parts with empty cells, formed as shown in the previous item.

Forming a set of candidate parts

So far we have identified which candidate parts reside in different points of the input data. Some data points contain no activations of candidate parts, while other data points may have one or more activations. Obviously, many candidate parts, representing regular surfaces, are activated in multiple data points, while other candidate parts, representing rare surfaces, are activated only in one or few data points.

To complete forming the set $\mathcal{C}(L_n)$ of candidate parts of the layer L_n we have to measure frequencies ν_i for each candidate part C_i^n, showing the number of occurrences of activations of this candidate part in the training data. Then we define a small threshold value t_r (usually $t_r = 1 + log(m)$, where m is a number of training models) and include all candidate parts with the frequency larger than this value to the set of candidate parts $\mathcal{C}(L_n)$. This thresholding is done to avoid including very rarely observed parts to the set

[1]The test for the existence or the absence of surface has been described in the end Subsection 5.4.1)

[2]Remember, that if a vocabulary contains only one subpart, this implicitly indicates the existence of an empty subpart in the opposite direction. If, yes, than this part has a realization in a point ρ, otherwise, there are no realizations of the layer L_n residing in the point ρ

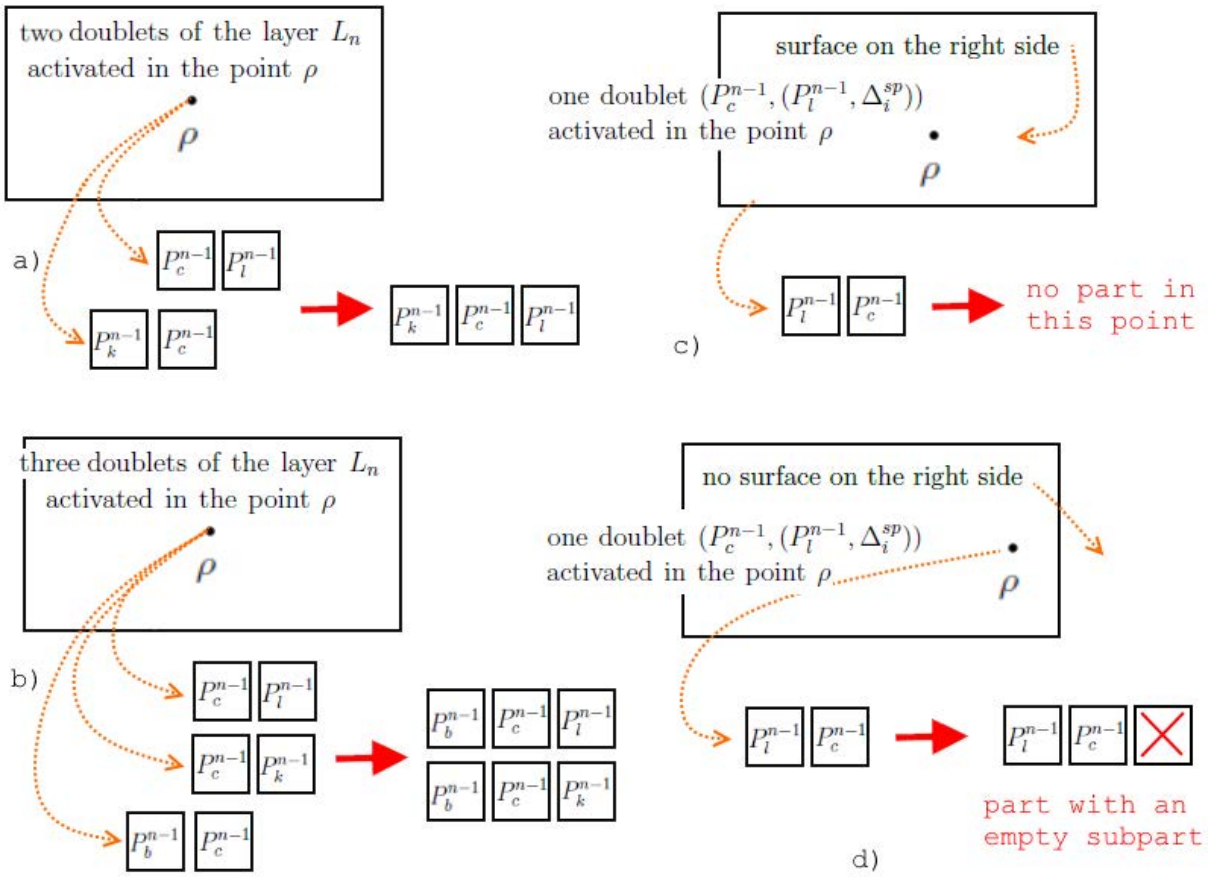


Figure 5.12: Illustration for different co-activation cases

of candidate parts. We assume, that if a candidate part appears only once or few times in the training data, then it is not statistically relevant anyways, and it should not undergo the part selection process. After the set of candidate parts is formed, we start the part selection procedure, that selects a vocabulary of the layer L_n from the set of candidate parts, i.e. if searches for $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$.

5.6 Part Selection Problem

Part selection is the key stage of the learning procedure. At this stage, we select a subset $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$ from the set of candidate parts and form the vocabulary of the layer L_n.

There are several main reasons for doing part selection:

- A set of candidate parts is typically very large. For example, when performing

vocabulary learning from the large Washington dataset [48], a number of candidate parts of the layers L_2 becomes close to $\mid \mathcal{C}(L_2) \mid \approx 2.5 * 10^4$, and this number grows rapidly for each subsequent layer, this is why a certain compression is required.

- There are many candidate parts representing surfaces geometrically similar to each other. On this basis certain grouping (e.g. clustering) techniques can be applied, such that only one representative part from each group is included in $\mathcal{S}(L_n)$. Such a grouping is purposed to facilitate generalization where a set of similar shapes is represented by a single part.

- Importance of candidate parts varies for different tasks, such as surface reconstruction, localization or discrimination of object categories. For instance, many frequently occurring parts represent planar surface patches or surface patches of low curvature. They are important for surface reconstruction, but usually not very discriminative, as they are activated in objects of many categories. On the other hand, some candidate parts may represent category-specific surfaces, i.e. those surfaces which appear in objects of a certain category only. These parts have a very large discriminative power, and therefore should be included in the vocabulary according to this criterion. Another example is that some surface structures, e.g. corners are usually well-localized in images, therefore these are significant cues for object localization, though these parts may be less important for other tasks.

These reasons suggest that part selection should have multiple objectives, i.e. (i) *compression* to reduce the number of parts included in the vocabulary, (ii) generalization, in which parts representing surfaces with similar geometric properties are grouped into a single node (e.g. cluster), and (iii) specific importance measures which may be different for different tasks (e.g. reconstruction, localization, and object category recognition) should be taken into account.

There are several ways how the part selection problem can be solved. One of the ways is to formulate and solve a multi-objective optimization problem. In this approach, the

cost function, comprising two or more terms is formulated, where each term represents a certain condition. For instance, the cost function may contain one or more *data terms* encoding similarity between parts and/or importance measures for candidate parts and the *regularization term* restricting the size of the vocabulary. The main advantage of such approaches is that they usually assume a simple and explicit formulation of the problem without any hidden constraints and assumptions. On the other hand, there are several drawbacks of this approach. First, the cost function should have one or more parameters assigning weights to each term, showing how influential this term is, and is not a trivial task to find an optimal set of weights. Second, the resulting solution of the multi-objective optimization problem represents a certain trade-off between several terms, therefore it may not be optimal from the point of view of each particular task (e.g., surface reconstruction or object category recognition).

The alternative to the multi-objective optimization problem is the "divide and conquer" strategy, in which the whole part selection problem is split into several simpler sub-problems, each of which takes into account only one or two criteria. Then each subproblem is solved separately, i.e. a certain set of parts representing the solutions for this subproblem is found, and after that, the solutions of all subproblems are merged into a single vocabulary $\mathcal{S}(L_n)$. In this thesis we investigate both ways of performing part selection.

Since this is a large section, it is worth describing how the remainder of this section is organized. Subsection 5.6.1 introduces a measure of geometric similarity of candidate parts, while Subsection 5.6.2 introduces different measures of importance for candidate parts. In Subsection 5.6.3 we formulate the part selection as multi-objective optimization problem, and finally in Subsection 5.6.4 we use "divide and conquer" strategy and perform part selection separately according to each criterion.

5.6.1 Distance Between Parts

Before formally formulating and solving the part selection problem, it is important to define a *measure of geometric similarity* of candidate parts. The similarity measure of two candidate parts C_i^n and C_j^n is denoted $d_v(C_i^n, C_j^n)$. It approximates the **volume between surfaces**, representing mean reconstructions of these two parts, given that these parts are centered in the same point. *Mean reconstruction* of a part is the configuration of its subparts in which they are located exactly at the relative positions defined by the parameters μ of the corresponding Gaussian distributions. Remember, that the distribution of possible relative positions of each subpart of a part is defined by the Gaussian distribution $\Delta_k^{sp} = (\mu_k, \Sigma_k)$, so the mean reconstruction of a part means that each subpart of this part is located exactly at the relative positions μ_k w.r.t. the central subpart.

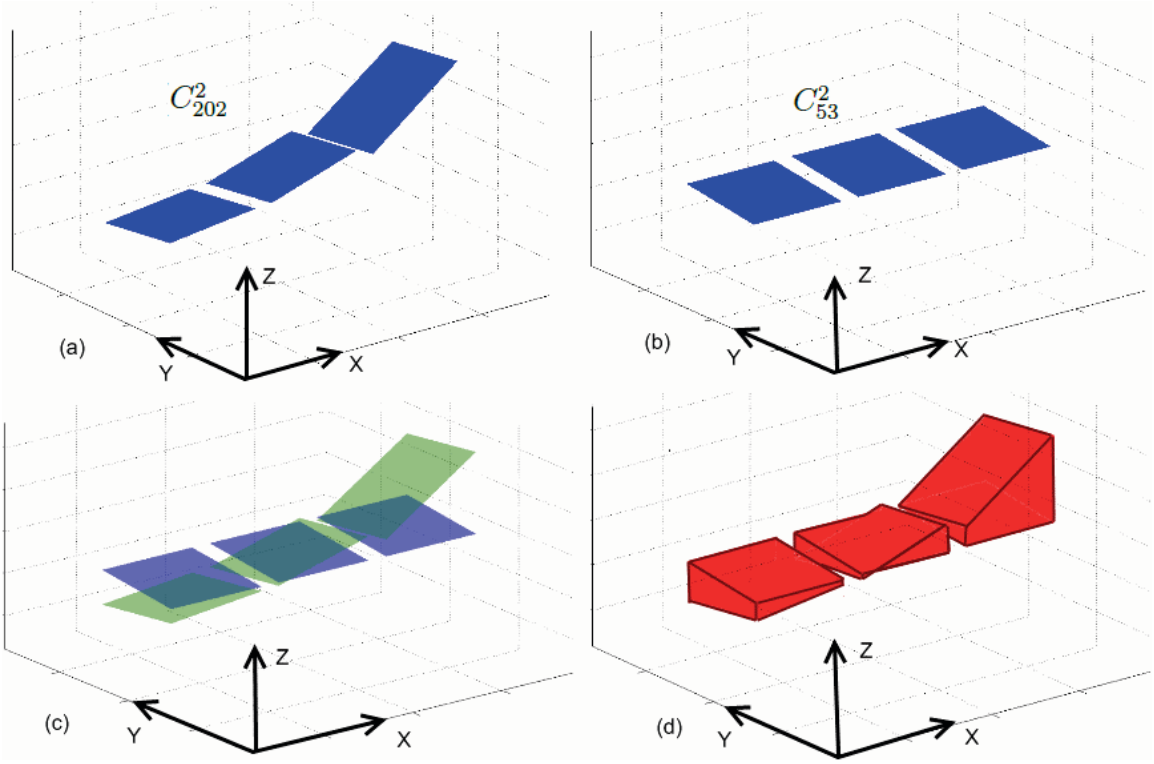


Figure 5.13: Distance between two candidate parts: (a) The mean reconstruction of a part, (b) The mean reconstruction of another part, (c) Both parts with aligned centres, (d) Volume between two surfaces

This similarity measure is illustrated using the example shown in Figure 5.13. Assume we measure the distance between two candidate parts C^2_{202} and C^2_{53} (learned from the

Washington dataset), whose mean reconstructions are shown in Figure 5.13(a) and (b). Once the surfaces representing the mean reconstructions of these parts are recovered, they should be located such that their central subparts are placed at the same point (Figure 5.13(c)). Then the volume between these two surfaces, shown in Figure 5.13(d) is used as a similarity measure $d_v(C^2_{202}, C^2_{53})$ for these two candidate parts.

If we measure $d_v(C^n_i, C^n_i)$, then obviously, the volume is equal to zero. If parts represent different surfaces, or the same surfaces of different orientations, then the volume becomes larger than zero. If at least one part contain an empty cell, then the following three cases should be taken into account:

Case 1. If one part contains the empty cell, while another part does not, these parts are always considered to be non-similar, and d_v is set to the infinity.

Case 2. If both parts contain empty cells at the same side from the central subpart, then the distance is computed in a normal way, as a volume between two surfaces.

Case 3. If two parts contain empty cells on different sides from the central subparts, then these parts are considered to be non-similar, and d_v is set to the infinity.

Note, that the proposed similarity measure may be applied both to measure the similarity between candidate parts and vocabulary parts. Also note, that for computational reasons, (e.g. when minimizing the energy functions described in Subsection 5.6.3) we replace infinite distances between candidate parts by a very large constant.

5.6.2 Importance Measures of Candidate Parts

In the previous subsection the measure of geometric similarity between candidate parts was introduced. In addition to that, the part selection procedures requires defining different importance measures for candidate parts, and this subsection describes several measures of importance for candidate parts.

Discriminativeness of Candidate Parts

Since one of the primary goals of this thesis is object category recognition, *discriminativeness* is the most substantial importance measures for candidate parts. This importance measure is based on the distribution of realizations of a candidate part across object categories. For instance, if a candidate part is activated in objects of one category only, this part is considered to be very discriminative (category-specific), while if a candidate part appears in objects of many categories, it is much less discriminative, and therefore less helpful for object categorization.

Discriminativeness of each candidate part is measured in two steps. First, the histogram showing the occurrence counts of part realizations in objects of different categories are built. After normalization, these histograms approximate probabilities of categories given a candidate part. Figure 5.14 depicts normalized histograms for three candidate parts learned from the Washington dataset, containing 51 object categories. Second, the *histogram entropy* H_e, which serves as a measure of part discriminativeness, is computed from the histogram of each candidate part using the Equation 5.5:

$$H_e = -\sum_{i=1}^{n} b_i \log(b_i), \tag{5.5}$$

where b_i is the i-th bin of the histogram approximating probability of object category i given the part, and n is a number of object categories in the dataset. In the case of $b_i = 0$ for some i, the value of the corresponding summand $0 \log(0)$ is taken to be 0, which is consistent with the limit:

$$\lim_{p \to 0+} p \log(p) = 0.$$

Parts with lower entropy are more discriminative, than parts with higher entropy. Figure 5.14 (a) shows the histogram for the candidate part, which is activated in objects of many categories. Such a distribution of part realizations across object categories leads to high values of histogram entropy H_e. On the other hand, Figure 5.14 (c) depicts a histogram for the candidate which has high probabilities of some categories given a part. Such a

distribution leads to lows value of histogram entropy H_e.

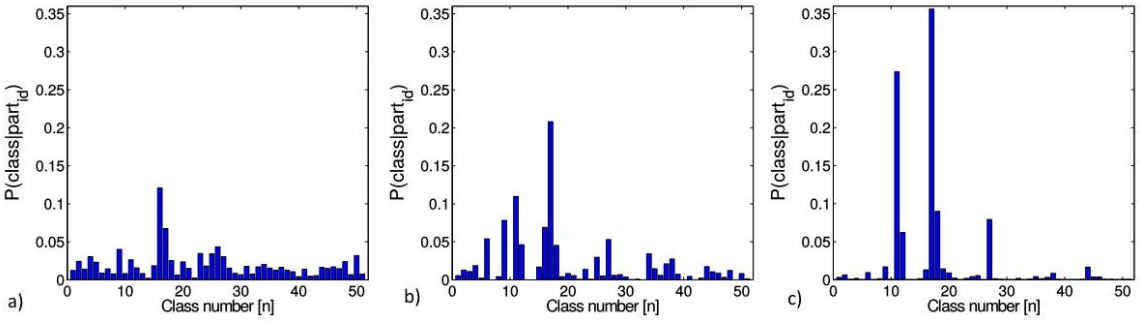


Figure 5.14: Histogram for candidate parts with: a) high entropy, b) moderate entropy, c) low entropy.

Coverage of Candidate Parts

Each part represents a surface of a certain area, which can be assumed to be proportional to the average number of data points, i.e. pixels in range images or points in point clouds, spanned by realizations of this part. The following importance measures for a candidate part approximate the overall surface area represented by all realizations of this part, by counting the number of data points spanned by these realizations.

Coverage (denoted $Cover(C_i^n)$) of the candidate part C_i^n is the number of data points in the whole training set, spanned by all realizations of this part. *Conditional coverage* $Cover(C_i^n \mid \mathcal{Y})$ of the candidate part C_i^n given a set of parts $\mathcal{Y}$ is the number of data points in the training set, spanned by all realizations of the part C_i^n, excluding those data points that are spanned by realizations of all parts from the set $\mathcal{Y}$.

Figure 5.15 illustrates these definitions. Assume we have two candidate parts C_1^2 and C_2^2. Figure 5.15 (b) shows projections of the surfaces modeled by these candidate parts on the XY plane. Also assume the training set contains only one surface represented by the point cloud (with blue points) shown in Figure 5.15 (a) and (c). Figure 5.15 (a) shows that the candidate part C_1^2 (red) is activated in two locations on this surface. Since all realizations of this candidate part span the area in which 27 points of the point cloud are located, coverage of this candidate part is equal to 27, i.e. $Cover(C_1^2) = 27$. The second

candidate part $Cover(C_2^2)$ (green) has only one part realization on this surface, as shown in Figure 5.15 (c). This part realization spans the area in which 14 points of the point cloud reside, therefore $Cover(C_2^2) = 14$.

Now assume $\mathcal{Y} = \{C_2^2\}$, and compute the conditional coverage $Cover(C_1^2 \mid \mathcal{Y})$. Overall, realizations of the part C_1^2 span 27 points, however, 8 of these points are also spanned by the realization of the part C_2^2 from $\mathcal{Y}$, therefore $Cover(C_1^2 \mid \mathcal{Y}) = 19$. Similarly if $\mathcal{Y} = \{C_1^2\}$, then $Cover(C_2^2 \mid \mathcal{Y}) = 6$.

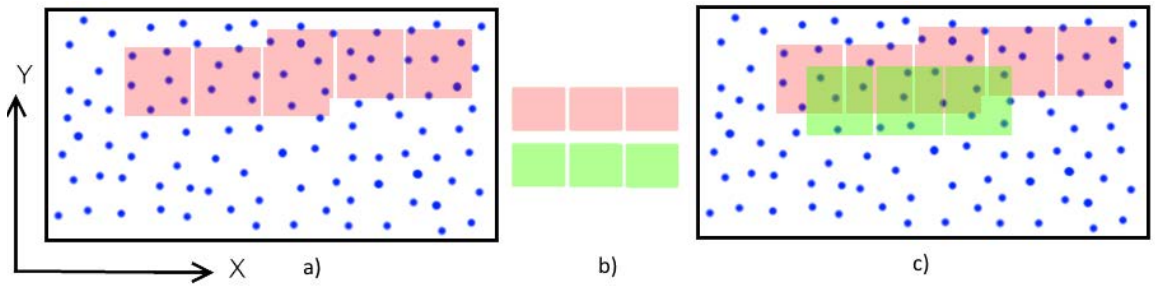


Figure 5.15: Explanation for coverage and conditional coverage. a) area spanned by realizations of the candidate part C_1^2, b) projections of the candidate parts C_1^2 and C_2^2 on the X-Y plane, c) area spanned by realizations the candidate parts C_1^2 and C_2^2

5.6.3 Part Selection as a Multi-Objective Optimization Problem

One of the approaches to the part selection problem is to formulate and solve a multi-objective optimization problem. We define a *cost function* (also termed an *energy function*) E and formulate the part selection problem as an energy minimization problem. At the first stage we defined a relatively simple energy function comprising two terms. The first term, called *similarity term* measures average distance between selected parts and the candidate part which has not been selected, thus measuring the reconstruction error, i.e. how well the set of candidate parts can be is represented by the vocabulary. The second term penalizes vocabularies with large cardinality, i.e. it restricts the number of parts to be included in the vocabulary. Thus the cost E function takes the following form:

$$E(\mathcal{S}(L_n)) = \sum_{C_i^n \in \mathcal{C}(L_n)} d_v(C_i^n,\, C'(C_i^n))\ \nu_i\ +\ \alpha\, |\mathcal{S}(L_n)| \tag{5.6}$$

where ν_i is the number of occurrences of activations of the candidate part C_i^n in the training data, $d_v(\cdot,\cdot)$ is a distance function that quantifies the geometric similarity between two parts (described in Section 5.6.1), and $C'(C_i^n)$ is the part in $\mathcal{S}(L_n)$ that is closest to C_i^n, according to this similarity measure. Also $\alpha \in \mathbb{R}^+$ is a meta-parameter that regulates the trade-off between the precision of the representation and the number of selected parts.

The objective is to find the vocabulary $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$ that minimizes the cost function described by Equation 5.6, i.e. the resulting vocabulary should be found by solving the following equation:

$$\mathcal{S}(L_n) = \underset{\mathcal{S}(L_n)\subseteq\mathcal{C}(L_n)}{\arg\min}\ E(\mathcal{S}(L_n)) \tag{5.7}$$

We solve this problem using a greedy algorithm, which includes candidate parts in the vocabulary one after another, on each iteration searching for the candidate part $C_i^n \in \mathcal{C}(L_n)$ leading to the smallest value of $E(\mathcal{S}(L_n) \cup C_i^n)$ and including this part to the vocabulary $\mathcal{S}(L_n)$. The algorithm stops when there are no more candidate parts in $\mathcal{C}(L_n)$ which, being included in the vocabulary, would lead to the further decrease in the cost function E. Note that this algorithm is not guaranteed to find a globally optimal solution, however, it is a useful strategy for finding reasonably good feasible solutions to the given problem[1].

The vocabulary $\mathcal{S}(L_n)$ obtained by minimizing this cost function has the following properties:

1. Parts with large number of activations in the training data stand higher chances to

[1] In Section 5.10 we will show that the results, achieved using the vocabularies selected by the presented greedy algorithm, do not substantially differ from the results achieved by other part selection methods (described in the next Subsections) and from the results achieved by other methods tested on the same dataset.

be included in the vocabulary, than parts with small number of activations,

2. Discriminativeness of candidate parts is not taken into account,

3. As parts representing corners, edges or other surfaces of high curvature are usually much less frequently observed in the training data, these surface types become poorly represented in the vocabulary.

To fix the problems described in the items (2) and (3) we incorporate the measure of the part discriminativeness (histogram entropy H_e) into the cost function by introducing the additional data term purposed to stimulate selection of discriminative parts to the vocabulary. The cost function comprising three terms looks as follows:

$$E(\mathcal{S}(L_n)) = \sum_{C_i^n \in \mathcal{C}(L_n)} d_v(C_i^n, C'(C_i^n))\ \nu_i \ + \ \alpha\,|\mathcal{S}(L_n)| \ + \ \beta \sum_{C_j^n \in \mathcal{S}(L_n)} H_e(C_j^n) \qquad (5.8)$$

where $H_e(C_j^n)$ is histogram entropy (introduced in Equation 5.5) of the candidate part C_j^n and $\beta \in \mathbb{R}^+$ is a meta-parameter showing importance of the third term relatively to the first and the second one. Results for both part selection strategies are presented in Section 5.10.

5.6.4 Divide and Conquer Strategy for Part Selection

The idea of formulation of the part selection problem as multi-objective optimization has some substantial drawbacks, the most important of which is that the resulting solution represents a certain trade-off between several terms, therefore it usually becomes not optimal for particular tasks (e.g. object categorization). Especially, this problem becomes acute when two data terms of the cost function contradict to each other, for instance, if a candidate part has a low number of occurrences ν_i, but high discriminativeness (low entropy H_e). In such cases it becomes quite unlikely that this part will be included in the vocabulary, despite its usefulness for object categorization.

As an alternative to the multi-objective optimization, it is possible to apply a "divide and conquer" strategy, i.e. to separately find several subsets of parts, representing solutions for different sub-problems, and then merge these subsets to a single vocabulary. For instance, we can apply MDL-based part selection to find the minimal set of parts $\Lambda_r(L_n) \subseteq \mathcal{C}(L_n)$ required for surface reconstruction, and the simple optimization function for selecting a set of the most discriminative parts $\Lambda_d(L_n) \subseteq \mathcal{C}(L_n)$, and finally obtain the vocabulary by merging these two sets $\mathcal{S}(L_n) = \Lambda_r(L_n) \cup \Lambda_d(L_n)$.

MDL-Based Part Selection

Rissanen's Minimal Description Length (MDL) principle [110] has been quite commonly employed in the context of compositional hierarchies [28, 32, 36]. In the approach presented in this subsection we solve the part selection problem based on similar ideas. We search for the minimal subset of candidate parts which is required to explain the training data in terms of the shape vocabulary. Coverage (presented in Subsection 5.6.2) is used as a measure of the explanatory power of each part. In our approach the approximate solution for this problem is found using a greedy algorithm, similar to the algorithm used by Fidler *et al.* [31, 107, 42].

The algorithm for part selection is presented in Algorithm 8. In the beginning, the part with the largest coverage is selected from the set of candidate parts $\mathcal{C}(L_n)$ (assume this is a part C_i^n). Then this part is excluded from the set $\mathcal{C}(L_n)$ and included in the vocabulary $\Lambda_r(L_n)$. After that, all parts that are similar to C_i^n according to the similarity measure d_v are excluded from the set of candidate parts $\mathcal{C}(L_n)$. This step is done to prevent including many parts representing similar surfaces in the vocabulary. After that, the same procedure is performed on each iteration: we search for the part with the largest conditional coverage given the vocabulary $\Lambda_r(L_n)$, include this part in the vocabulary $\Lambda_r(L_n)$ and exclude it (together with the similar parts) from the set of candidate parts $\mathcal{C}(L_n)$. In other words, on each iteration we search for a part with the largest explanatory power and include this part in the vocabulary. The explanatory power is measured as a conditional coverage, i.e.

the amount of data points that can be explained by each candidate part, and can not be explained by the parts previously included in the vocabulary. Note, that the conditional coverage of the non-selected candidate parts may decrease when new parts are included in the vocabulary, that is why conditional coverage of all parts from the set $\mathcal{C}(L_n)$ has to be re-computed after each iteration.

Algorithm 8: MDL-based part selection algorithm

Data: Set of candidate parts $\mathcal{C}(L_n)$,
Pre-defined similarity threshold T_s,
Pre-defined coverage threshold T_c.
Result: The vocabulary $\Lambda_r(L_n)$.

$\Lambda_r(L_n) = \emptyset$;
Find the part $C_i^n \in \mathcal{C}(L_n)$ with the largest $Cover(C_i^n)$;
Exclude the part C_i^n from the set $\mathcal{C}(L_n)$ and include it in $\Lambda_r(L_n)$;
Find the set of parts $\Upsilon = \{C_j^n\}_j$ such that $d_v(C_i^n, C_j^n) < T_s$;
Exclude parts that belong to Υ from the set $\mathcal{C}(L_n)$;
repeat
    Find the part $C_k^n \in \mathcal{C}(L_n)$ with the largest $Coverage(C_k^n|\Lambda_r(L_n))$;
    Exclude C_k^n from $\mathcal{C}(L_n)$, and include it in the $\Lambda_r(L_n)$;
    Find the set of parts $\Upsilon = \{C_l^n\}_l$ such that $d_v(C_k^n, C_l^n) < T_s$;
    Exclude parts that belong to Υ from the set $\mathcal{C}(L_n)$;
until *largest* $Coverage(C_m^n|\mathcal{S}(L_n)) > T_c$;

Note that, strictly speaking, surface area is not proportional to the number of points of the point cloud, due to varying density of point clouds. For instance, surfaces that are perpendicular to the viewing direction, are represented by a much more dense point cloud than those surfaces that are "almost parallel" to the viewing direction. Therefore, coverage and conditional coverage (introduced in Subsection 5.6.2 and used in Algorithm 8) of parts representing surfaces that are "almost perpendicular" to the viewing direction is typically much larger than coverage and conditional coverage of parts representing surfaces that are "almost parallel" to the viewing direction.

Now we have to highlight that this is not a problem, and the presented algorithm still works properly and includes in the vocabulary parts representing differently orientated surfaces. The algorithm starts with parts having the largest coverage, i.e. typically parts representing surfaces that are "almost perpendicular" to the viewing direction, and

includes them in the vocabulary. Once parts with large coverage are included, the algorithm proceeds to parts with smaller coverage and includes them in the vocabulary as well. That means the varying density of point clouds affects only the order in which the algorithm includes parts in the vocabulary, but ultimately does not prevent including parts representing all orientations in the vocabulary.

Entropy-based Part Selection

The objective of the entropy-based part selection is to select a set of the most discriminative parts. In this case, the simplest possible way of part selection would be to measure the histogram entropy H_e (introduced in Subsection 5.6.2) for each candidate parts, then define a certain threshold T_H and finally include all parts with entropy $H_e \leq T_H$ in the vocabulary. However, this approach would result in the selection of many parts representing similar surfaces in the vocabulary. That means the part selection procedure should take into account similarities between candidate parts to prevent selection of parts representing similar surfaces.

Our strategy for selecting the set of discriminative parts can be described as follows. At the first stage we find the subset $\Lambda_{dd}(L_n) \subseteq \mathcal{C}(L_n)$, such that all parts from the set $\Lambda_{dd}(L_n)$ have histogram entropy lower than the pre-defined threshold T_H. After that $\Lambda_{dd}(L_n)$ contains only discriminative parts, many of which, however, represent surfaces which are similar to each other. To reduce a size of the vocabulary and to facilitate better generalization we should find a subset $\Lambda_d(L_n) \subseteq \Lambda_{dd}(L_n)$ such that (i) $\Lambda_d(L_n)$ sufficiently well represents the set $\Lambda_{dd}(L_n)$, and (ii) the cardinality of $\Lambda_d(L_n)$ is rather small. To satisfy both constraints we define the following cost function:

$$E(\Lambda_d(L_n)) = \sum_{C_i^n \in \Lambda_{dd}(L_n)} d_v(C_i^n, C'(C_i^n)) \;+\; \gamma\,|\Lambda_d| \tag{5.9}$$

where $C'(C_i^n)$ is the part in $\Lambda_d(L_n)$ that is closest to C_i^n, according to the similarity measure d_v. Also $\gamma \in \mathbb{R}^+$ is a meta-parameter that regulates the trade-off between the

precision of the representation and the number of selected parts. Note, that the important difference of Equation 5.9 from Equation 5.6 is that number of parts activations (ν_i) in the training set is not taken into account in Equation 5.9. The reason for this is that we find that these two conditions (frequency and discriminativeness) usually contradict to each other, i.e. very discriminative candidate parts are usually non-frequently observed in the training data. That's why occurrence counts of part realizations in the training data are not taken into account in this energy function.

We find the set $\Lambda_d(L_n) \subseteq \Lambda_{dd}(L_n)$ minimizing the cost function $E(\Lambda_d(L_n))$ using the same greedy algorithm as the one described in Subsection 5.6.3. After sets $\Lambda_d(L_n)$ and $\Lambda_r(L_n)$ are found they are merged to a single shape vocabulary $\mathcal{S}(L_n) = \Lambda_r(L_n) \cup \Lambda_d(L_n)$. Results for both part selection strategy are presented in Section 5.10.

Part selection is the last step of the learning procedure and the resulting vocabulary $\mathcal{S}(L_n)$ of the layer L_n represents the output of the learning algorithm.

5.7 Inference of a Single Layer

In this section the inference procedure for a single layer $L_n, (n \geqslant 2)$ is described. The goal of inference is to match the input data against the vocabulary in order to find realizations of vocabulary parts in the data.

This algorithm proceeds in two steps. First, inference of doublets from the input data is done, exactly as was described in Algorithm 7. Second, when activations of doublets of the layer L_n in the training data are found, the co-activations of doublets each data point are analyzed and the inference of parts is performed based on this analysis.

There are five possible scenarios, which are described below. Notice, that these scenarios are very similar to the cases described in Section 5.5.4, that is why the same Figure (Figure 5.12) may be used for illustration.

1. If there are no doublets of the layer L_n activated in a point ρ then, there are no part realizations of the layer L_n in this point.

2. If two doublets are activated in the point ρ, for example, doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$, and these doublets describe surface in the opposite directions from ρ (e.g. on the left and on the right sides), then we check if the vocabulary contains a part $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}), (P_l^{n-1}, \Delta_i^{sp}))$. If yes, then there is a realization of this part in the point ρ, otherwise there are no part realizations of the layer L_n in this point (see the illustration in Figure 5.12 (a)).

3. If there are more than two doublets activated in the point ρ, for example, $m \geq 1$ doublets describing surface on one side from ρ, and $k \geq 1$ doublets describing the surface on the other side from ρ, where $max(k, m) > 1$, then we form $m \times k$ parts as shown in Item 2. Let us consider the example when $m = 2$ (doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$) and $k = 1$ (doublet $(P_c^{n-1}, (P_b^{n-1}, \Delta_a^{sp}))$). Then we have to form $m \times k = 2 \times 1$ parts from these doublets, i.e. $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$. Then for each of these parts we check if it belongs to the vocabulary, and, if yes, then it has a realization in the point ρ (see the illustration in Figure 5.12 (b)).

4. If there is only one doublet $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ activated in a point ρ, then we need to perform an extra check to figure out whether or not surface in the direction opposite to the one described by Δ_i^{sp} exists. For instance, let us assume that a doublet describes a surface on the left side from ρ. Then we should check whether or not surface on the right side from ρ exists.

   - If the surface exists, then there are no part realizations of the layer L_n residing in the point ρ (see the illustration in Figure 5.12 (c)).
   - If the surface does not exist, then we check if a part $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ exists in the vocabulary[1] If so, then this part has a realization in a point ρ, otherwise, there are no realizations of the layer L_n residing in the point ρ (see the illustration in Figure 5.12 (d)).

[1]Remember, that if a vocabulary contains only one subpart, this implicitly indicates the existence of an empty subpart in the opposite direction.

5. If there are m doublets activated in the point ρ, where $m > 1$, and all these doublets describe surface in the same directions from ρ then we have to check the existence of surface in the opposite direction. If the surface exists there, then there are no part realizations of the layer L_n in the point ρ. If there is no surface there, then we form m parts with empty cells, check if these parts exist in the vocabulary and infer realizations of these part accordingly.

5.8 Pooling

Pooling is the form of non-linear downsampling, purposed to progressively reduce the spatial resolution of the representation by reducing the number of part realizations in order to speed-up computations in higher layers. Notice that pooling is an optional procedure, meaning that the presented system works without it, however, in this case the computations on higher layers become several times slower. The difference between inference time with and without pooling becomes especially large on higher layers. For instance, as will be shown in Subsection 5.10.1, inference of the layer L_5 with pooling takes approximately 5-10 minutes, while without pooling this time exceeds two hours.

Note, that all experiments presented in Section 5.10 were performed **with pooling**, otherwise the computational time would be too large (up to several days for each experiment)[1]. That is why we did not evaluate the performance of the system without pooling on the Washington dataset. We assume, that computations **without pooling** could lead to better results, since pooling removes many part realizations, some of which may carry a certain discriminative information.

Our pooling algorithm can be described as follows. After inference of every other layer (i.e. L_3, L_5) we tile the support region of the object in the XY plane by non-overlapping equally-sized squares. For the layer L_3 the size of these squares should be chosen such

[1]Notice, that in order to perform an experiment on the Washington dataset, one has to split the dataset into the training and testing set 10 times, and therefore 10 times perform learning of the vocabulary, inference and categorization

that each region contains 3×3 pixels of the original depth image, while for the layer L_5 the squares become 3 times larger. Then we extract all part realizations located within each square region. After that, we measure the number of occurrences of realizations of each part in the region. Assume there is a square region, in which four realizations of the part P_i^n, two realizations of the part P_j^n and one realization of the part P_k^n are located. Then the pooling algorithm will delete all realizations of parts P_j^n and P_k^n, and 3 out of 4 realizations of the part P_i^n. Only one part realization of P_i^n should be left, namely, the realization which is the closest to the centre of the square region[1]. If two or more parts have equal number of realizations in a support region, e.g. if both parts P_i^n and P_j^n have 4 part realizations in the region, while the part P_k^n has only three realizations, then we should leave one realization of P_i^n and one realization of P_j^n in the region, and delete all remaining realizations.

5.9 Object Categorization from Range Images

In this section the approach for object categorization from range images is described. Subsection 5.9.1 presents the Histograms of Compositional Parts (HoCP) which serve as a category descriptor, while Subsection 5.9.2 describes classification of HoCPs by the Support Vector Machines (SVMs) with four different types of kernels.

5.9.1 Histogram of Compositional Parts

The descriptors used for category recognition in this thesis are Histograms of Compositional Parts (HoCP), which are block-wise histograms showing occurrence counts of part realizations in point clouds. Similar category descriptors for 2D images, showing occurrence counts of part realizations in images, were used by Tabernik *et al.* [216]. In this subsection the process of building histograms of compositional parts is presented.

Assume there is a depth image I representing a range scan of an object. During

[1]If there are two or more realizations at the same distance from the centre, the choice is made randomly

pre-processing (Section 5.2) this image is converted to the point cloud $\mathcal{P}$, and then the inference procedure (described in Section 5.7) is performed subsequently for all layers $L_n\,(n \in [1, \mathcal{N}])$ starting from the layer L_1. After that, the histograms of compositional parts are built as shown in Figure 5.16. The rectangular region Ω of the XY plane containing the point cloud $\mathcal{P}$ is partitioned into 4 (2×2) and 9 (3×3) sub-regions of equal size, and the histograms are built for each sub-region. We denote the histogram for the i-th sub-region as $\mathcal{H}_i$, where $i \in [1, 14]$, i.e. the histogram $\mathcal{H}_1$ shows occurrence counts of part realizations in the whole region Ω, the histograms from $\mathcal{H}_2$ to $\mathcal{H}_5$ show occurrence counts of part realizations in different 2×2 sub-regions of Ω, and the remaining histograms from $\mathcal{H}_6$ to $\mathcal{H}_{14}$ correspond to the partitioning of Ω into 3×3 sub-regions. As each histogram $\mathcal{H}_i$ depicts the number of occurrences of part realizations of all layers from L_1 to $L_{\mathcal{N}}$, its length can be computed using the following equation:

$$| \mathcal{H}_i | = \sum_{n=1}^{\mathcal{N}} | \mathcal{S}(L_n) | \tag{5.10}$$

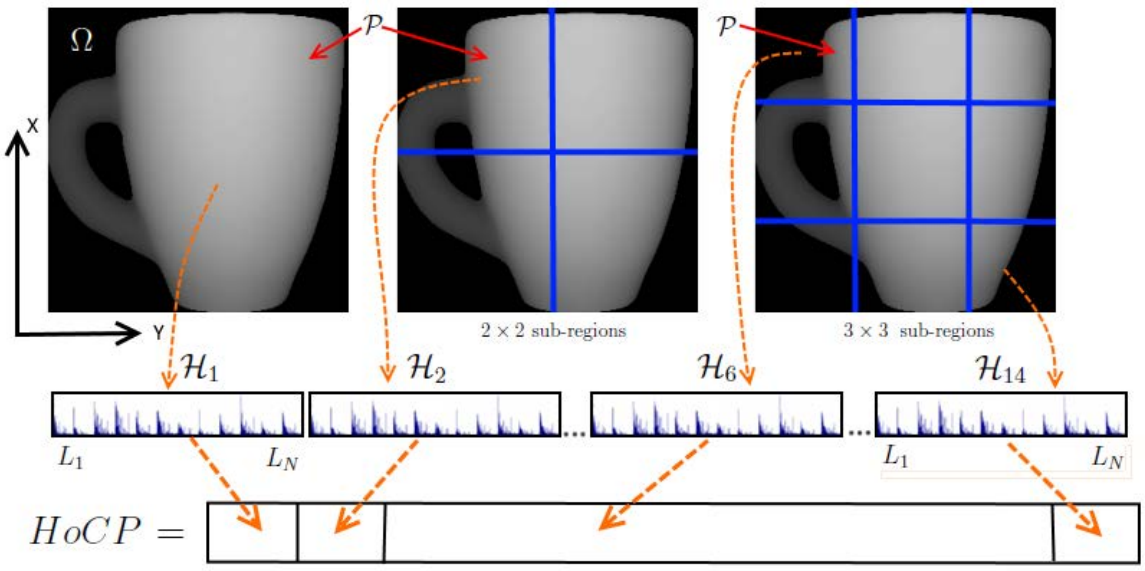


Figure 5.16: Histogram of Compositional Parts

After all histograms $\mathcal{H}_i$ are built, each of these is normalized in the following way. As it shows occurrence counts for part realizations of different layers (from 1 to $\mathcal{N}$), the normalization is done for each layer separately. That means, for each n from 1 to $\mathcal{N}$

we extract bins $\Psi_n \subset \mathcal{H}_i$, showing occurrence counts of part realizations of the layer L_n, sum up all values in these bins (assume the sum is equal to S), and, if $S > 0$, divide the values in bins Ψ_n by S, thus performing layer-wise normalization. Such a normalization is important for the EMD and Quadratic-Chi affinity measures, where cross-bin dependencies are taken into account. See Subsection 5.9.2 for more details.

When all 14 histograms are built and normalized, they are concatenated into a single Histogram of Compositional Parts, which therefore has length computed using the equation:

$$N = \mid HoCP \mid = 14 \mid \mathcal{H}_i \mid \tag{5.11}$$

These descriptors are classified using Support Vector Machines with different types of kernels which are considered in the next subsection.

5.9.2 Classification of HoCP by SVM

After histograms of compositional parts are built, they are used as descriptors for category recognition. For classification we use the Support Vector Machine (SVM) classifier, since it achieves outstanding performance in numerous applications, and therefore it is one of the most popularly employed classifiers in our days. It was originally invented by Vapnik and Chervonenkis in 1963, but modified to its current form (with soft margin) by Cortes and Vapnik in 1995 [217]. SVM requires defining *kernels*, i.e. similarity functions over pairs of data points. We use four types of kernels for classifying HoCPs: Gaussian kernels, χ^2 kernels, Quadratic-chi [218] kernels, and Earth Mover's Distance (EMD) [219][220] kernels.

Gaussian and χ^2 kernels represent bin-by-bin similarity measures which are very commonly employed in SVMs, moreover, these kernels are implemented in standard packages for SVM, such as LibSVM [221], therefore the detailed description of these kernels is not provided in this thesis. However, usage of two other similarity measures, namely Quadratic-chi and EMD, within the SVM framework is much less straightforward, be-

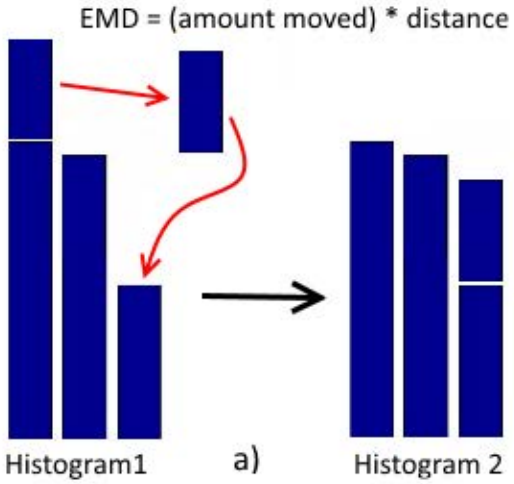
cause (i) these similarity measures require defining cross-bin distance matrices, (ii) these measures are not guaranteed to produce positive semi-definite (PSD) kernel matrices. In the following subsections we shortly describe the Quadratic-chi and EMD affinity measures, show how to define a cross-bin distance matrix A, describe the *kernelization trick* enabling both affinity measures to produce PSD kernel matrices, and present our approach for parameter selection.

Earth Mover's Distance

The Earth Mover's Distance (EMD), invented by Rubner *et al.* [219][220], is the affinity measure showing the distance between two distributions (e.g. histograms) over a certain domain. Informally, these distributions can be understood as two different ways of piling up a certain amount of dirt over this domain and the EMD is the minimal amount of work that must be performed to transform one pile into the other. The amount of work is calculated as the amount of dirt moved times the distance by which it is moved. For computing the EMD distance between two bounded histograms, e.g. B and Q, comprising N bins ($B, Q \in [0,1]^N$, where N is computed by Equation 5.11) it is necessary to define a cross-bin similarity matrix A, i.e. a non-negative bounded symmetric matrix ($A \in [0,U]^N \times [0,U]^N$ and $\forall_{i,j}\ A_{ii} \leq A_{ij}$) whose elements A_{ij} denote distance between bins i and j of these histograms, i.e. the distances by which dirt is moved when transferred from one bin to another. U here is a very large constant ($0 \ll U < \infty$), used for bins maximally distant from each other.

In Subsection 5.9.1, where HoCPs are described, it is shown that these histograms contain bins representing occurrence counts of part realizations of different layers (from 1 to $\mathcal{N}$) of the hierarchy located in different sub-regions (14 sub-regions) of the input image. Therefore, the matrix A can be defined in the following way:

- $A_{ii} = 0 \ \ \forall\, i$;
- If bins i and j represent parts from different sub-regions of the image, then $A_{ij} = U$,



A	1	2	3
1	0	1.91	2.23
2	1.91	0	2.15
3	2.23	2.15	0

b)

Figure 5.17: Illustration of the Earth Mover's Distance measure. a) EMD is the amount of dirt moved times the distance by which it is moved, b) Distances between bins are stored in the matrix A

i.e. we consider bins from different sub-regions to be non-similar;

- If bins i and j represent parts of different layers from the same sub-region of the image, then $A_{ij} = U$, i.e. we don't take into account similarities of parts of different layers and such parts are assumed to be always non-similar to each other;

- If bins i and j represent parts (e.g. parts P_k^n and P_k^n) of the same layer from the same sub-region of the image, then $A_{ij} = d_v(P_k^n, P_m^n)$, where d_v is the parts similarity measure introduced in Section 5.6.1. In case $d_v(P_k^n, P_m^n) = \infty$ (which may happen for the parts with empty cells), this value is replaced by the constant U.

This means that cross-region bins, and bins representing parts of different layers are considered to be non-similar, while in other cases similarity of bins depends on geometric similarity of the corresponding vocabulary parts.

Quadratic-Chi distance measure

Despite the fact that the EMD affinity measure takes into account cross-bin relations, it has one substantial drawback, which may become crucial when classifying histograms of compositional parts. This drawback is that the EMD is proportional to the amount of

mass moved from one bin to another, that is why the bins with large amount of mass (i.e. with higher values of B_i and Q_i) become more influential than the bins with a smaller amount of mass. This means that the EMD of two histograms is mainly affected by differences between large bins of these histograms rather than by differences of smaller bins. However, in HoCPs differences between small bins are sometimes very important, as many discriminative parts are non-frequent, therefore the number of occurrences of these parts in objects is rather small, and consequently the corresponding bins of the HoCPs always take small values. Actually, these small bins are quite often even more important for category recognition than large bins representing occurrence counts of very frequent but non-discriminative parts.

That explains the necessity of an affinity measure in which differences between small bins become as important as differences between large bins. An affinity measure of this type is a Quadratic-Chi [218] distance measure. This measure combines both properties that are important for classification of HoCP i.e. it reduces the effect of differences caused by bins with large values, and takes cross-bin relationships into account. The Quadratic-Chi histogram distance is computed by the following equation:

$$QC_m^A(B,Q) = \sqrt{\sum_{ij}\left(\frac{(B_i - Q_i)}{(\sum_c (B_c + Q_c)A_{ci})^m}\right)\left(\frac{(B_j - Q_j)}{(\sum_c (B_c + Q_c)A_{cj})^m}\right)A_{ij}}, \qquad (5.12)$$

where $0 \leq m < 1$ is a normalization parameter. The Quadratic-Chi affinity measure also requires defining a matrix A_{ij} for cross-bin similarities and we define this matrix in the same way as for EMD. Our experiments (Section 5.10) show that the Quadratic-chi distance measure achieves the best results in classifying histograms of compositional parts.

Kernelization of EMD

One of the constraints of the SVMs is that each affinity measure that is used in this classifier must produce a positive semi-definite (PSD) kernel matrix. A symmetric $n \times n$ real matrix M is positive semi-definite if the scalar $s = z^T M z$ is non-negative for each

non-zero column vector z of length n. A large number of efficient affinity measures (including EMD) are not guaranteed to produce a PSD kernel matrix. Zamolotskikh *et al.* [222] discuss several methods how to produce a PSD kernel from the EMD. We use one of the methods proposed in this paper, i.e. apply the transformation $K(Q,B) = exp(-\frac{EMD(Q,B)}{h})$, where Q and B are two HoCPs, and the h is the parameter (parameter selection is discussed in the next subsection).

Parameter selection

Support Vector machine requires defining several parameters (usually one or two). For instance, for the linear SVM one parameter C has to be defined, the SVM with Gaussian Kernels requires defining two parameters C and γ, while the SVM with Quadratic-Chi kernels requires defining parameters C and m (normalization parameter). We use 5-fold cross-validation to perform the grid search of the best set of parameters. Notice, that the parameter selection is a very time-consuming procedure, since the model has to be re-trained for each set of parameters. That is why a 5-fold cross-validation is a reasonable choice, although 10-fold can lead to more precise parameter selection.

Since doing a complete grid search with small step is time-consuming, we use a coarse grid search (with large step) at the first stage. After identifying a better region on the grid, a finer grid search inside that region is performed.

5.10 Evaluation

Table 5.1 presents the results in object categorization achieved by different methods on the Washington dataset. We use the standard evaluation protocol proposed by the authors of the dataset [48], according to which the dataset is split into the training and testing sets 10 times, and the average object categorization accuracy is computed. Similar to many other authors, we used only 20% of the dataset for training and testing, i.e. we were taking every fifth image of each object.

At the first stages we evaluated our method without empty subparts using HoCPs as category descriptors, and SVM with χ^2 kernels for classification. First we trained the layers L_2-L_5 of the hierarchy using the part selection method described by Equation 5.6. We performed a grid search to find the optimal parameter value α. The best accuracy we could achieve was 77.8 $\pm$ 2.4% using parts of the layers L_1-L_5. Since this method includes only frequently occurring candidate parts in the vocabulary (i.e. parts with the large value of ν) the discriminative power of this method is not very large. That means, a large number of occurrences of a candidate part does not imply the large discriminative power of this part.

Table 5.1: Results of different methods on Washington RGB-D dataset (depth channel) (*) - Methods pre-trained on ImageNet.

Method	Result, %
Random Forest [48]	66.8 $\pm$ 2.5
CNN-RNN [24] (2012)	78.9 $\pm$ 3.8
Hierarchical Matching Pursuit [205]	81.2 $\pm$ 2.3
CNN [26] raw depth data	80.1 $\pm$ 2.6
CNN [26] Colorization using surface normals (*)	84.7 $\pm$ 2.3
CNN [26] HHA colorization (*)	83.0 $\pm$ 2.7
CNN [26] Colorization using jet colormap (*)	83.8 $\pm$ 2.7
MM-LRF-ELM[1] [223]	82.8 $\pm$ 2.1
Semi-Supervised CNN [224]	82.6 $\pm$ 2.3
Convolutional Fisher Kernel [52]	85.8 $\pm$ 2.3
Our approach (χ^2-distance, layers 1-5, Eq. 5.6 for part selection	77.8 $\pm$ 2.4
Our approach (χ^2-distance, layers 1-5, Eq. 5.8 for part selection	78.1 $\pm$ 2.6
Our approach (χ^2-distance, layers 1-5, MDL)	78.4 $\pm$ 2.0
Our approach (QC-distance, layers 1-5, MDL)	79.3 $\pm$ 2.3
Our approach (QC-distance, layers 1-5, MDL+Entropy)	80.1 $\pm$ 2.2

[1]Multi-Modal Local Receptive Field Extreme Learning Machine

Continuation of Table 5.1	
Method	Result
Our approach (QC-distance, layers 1-5, MDL+Entropy, Empty subparts)	**80.4 ± 2.4**

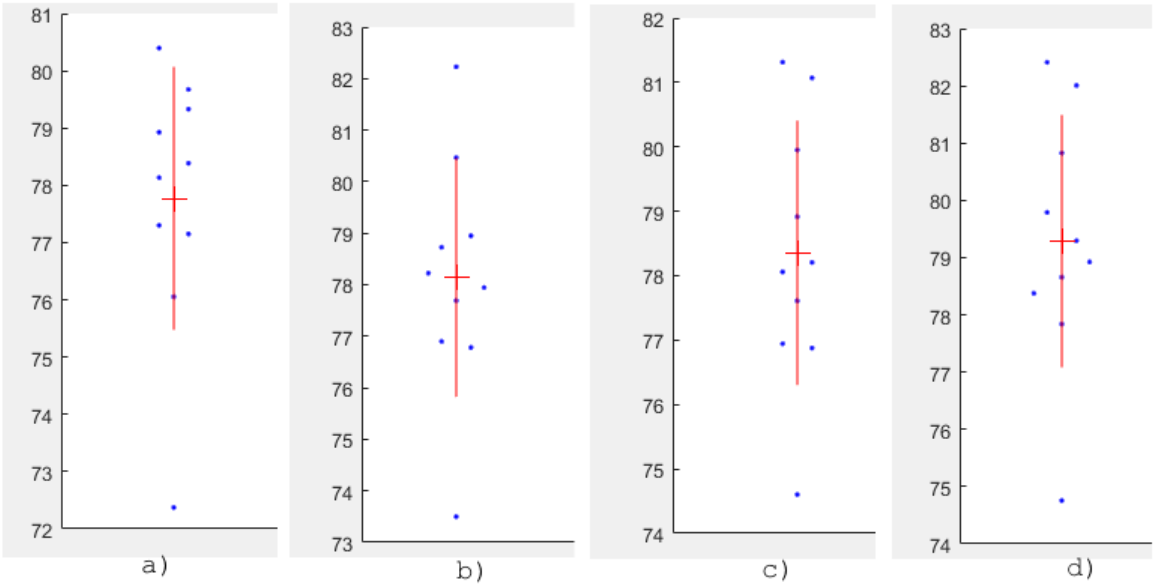


Figure 5.18: Beeswarm plots illustrating performance of different settings. Horizontal red line shows a mean value, while vertical lines show the standard deviation. a) χ^2-distance, layers 1-5, Eq. 5.6 for part selection, b) χ^2-distance, layers 1-5, Eq. 5.8 for part selection, c) χ^2-distance, layers 1-5, MDL-based part selection, d) QC-distance, layers 1-5, MDL-based part selection

On the next stage we evaluated the part selection method described by Equation 5.8. The grid search is done for two parameters α and β providing trade-off between different terms of the cost function. Although this part selection method led to slightly better results, namely 78.1 ± 2.6%, the improvement over the previous part selection method was very small. Apparently, the reason for this is that two conditions of the cost function, namely frequency of candidate parts, and their discriminativeness, often contradict each other, i.e. many frequently occurring parts are not discriminative, and vice versa. The attempt to find a trade-off between these two conditions did not lead to substantial improvement of the overall classification result.

Then we proceed to the MDL-based part selection method described in Section 5.6.4. This part selection improved the categorization result to 78.4 ± 2.0.% In contrast to

two previous part selection strategies, this method can select non-frequent parts, and presumably, this is the reason for the improvement of the results.

After that, we evaluate different kernels for SVM. It turns out, that EMD kernels do not provide any improvement over the χ^2 kernels, apparently because they mainly count differences between large bins, as was explained in Section 5.9. On the other hand, Quadratic-Chi kernels, for which the difference between small bins is more important, improved the results compared to χ^2 kernels ($79.3 \pm 2.3\%$).

On the next step we separately selected two sub-vocabularies, namely the MDL-based one, and the entropy-based one, computed using Equation 5.9, and then merged them to a single vocabulary. This merged vocabulary substantially improved the results ($80.1 \pm 2.2\%$), with Quadratic-chi kernels). This shows that the best strategy for part selection is to separately select sub-vocabularies according to each condition and merge them after that, instead of trying to find a trade-off between different terms of a cost function. For these experimental settings we also evaluated the results achieved by different kernels, and by using parts of different layers, as presented in Figure 5.19. Note that this Figure shows the results achieved using parts of L_1, then parts of L_1 and L_2, parts of L_1, L_2, L_3, etc.

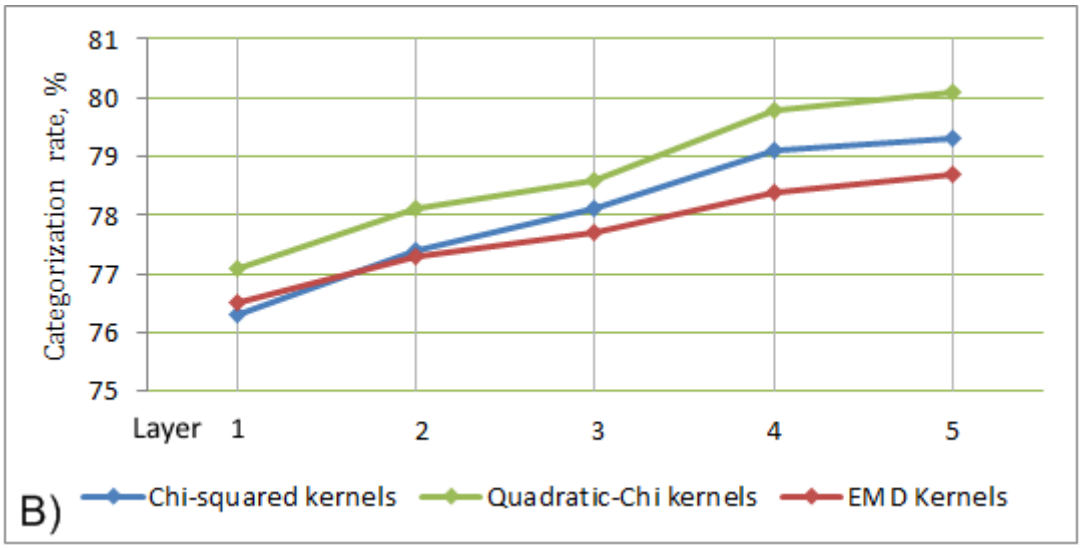


Figure 5.19: Object categorization results using parts of different layers from L_1 to L_5 selected using MDL-based and Entropy-based algorithms. SVM with χ^2, Quadratic-Chi and EMD kernels is used

Also, we present the similar plot showing difference between MDL-based part selection and the MDL+Entropy, where χ^2 kernels are used for classification (Figure 5.20). We see that the difference between both methods becomes larger on higher layers.

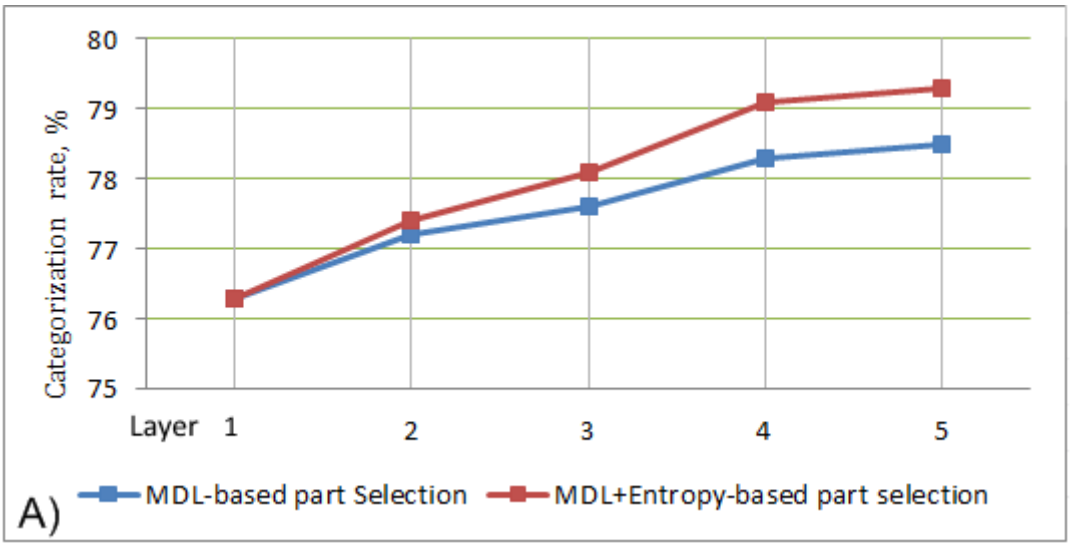


Figure 5.20: Object categorization results using parts of different layers from L_1 to L_5. MDL and MDL+Entropy methods are compared. SVM with χ^2 kernels are used for classification of HoCPs

Since we introduced empty subparts in the last phase of this work, we evaluated them in a separate experiment. We used the same settings i.e. parts from the layers L_1-$L5$, the MDL-based and the entropy-based part selection, and Quadratic-chi kernels for SVM. However, empty subpart led to only a very small improvement of the results ($80.4 \pm 2.4\%$).

5.10.1 Computational Resources and Timing

For the experiments presented in this Section we used the computer with the AMD FX(tm)-8300 eight-core processor and 16GB of the main memory. All algorithms are implemented in Matlab, and most of them are parallelized, i.e. all eight cores are used for most of the computations. In such settings learning of a single layer takes from approximately 25-30 minutes to 2.5 hours, depending on the employed part selection procedure. The slowest way of part selection is the MDL-based one, since the conditional coverage

has to be re-computed after each iteration, which requires going through the whole training set. On the other hand, other part selection methods are substantially faster, since they do not require such re-computations on each iteration. The inference of a single layer takes 5-10 minutes for each layer. However if the pooling is switched off, the inference time goes up very rapidly for higher layers, exceeding two hours for the layer 5. The cross-validation for the parameter selection takes 1-1.5 hours. SVM classification typically takes 2-10 minutes, depending on the used kernels and parameters.

5.10.2 Absence of OR-Nodes

In this subsection we explain why the concept of the presented view-based compositional hierarchical framework does not contain OR-nodes. The main reason for this is that parts are included in the vocabulary based on multiple criteria. For instance, some parts may be included in the vocabulary by the MDL-based algorithm, while other parts may be selected by the entropy-based algorithm. Given that parts are selected according to different criteria, there is no point in grouping them into OR-nodes, according to a single criterion, e.g. based on their geometric similarity. For instance, if two parts represent geometrically similar surfaces, but only one of them have a large discriminative power, there is no reason for grouping them into a single node (or cluster).

Notice, that in the surface-based hierarchy, presented in the next chapter, parts are selected in the vocabulary only according to a single criterion, namely the frequency of their occurring in the training data. In this framework geometrically similar parts are grouped into OR-nodes, and the most frequently occurring part within each OR-node becomes its *representative part*. More details will be explained in the next chapter.

5.11 Conclusion

In this chapter we described the novel compositional hierarchical framework for learning of surface shape features. We described the design principles behind this framework and

all ingredients of the system, i.e. the reference frame, the vocabulary representation, the composition rules, empty subparts, etc. We also provided the detailed description of each step of the learning and inference algorithms, including pre-processing of the input data, inference of the first layer, collecting of the co-occurrence statistics, clustering of statistical maps and forming doublets, inference of doublets from the training data, part selection, and pooling.

The most difficult learning stage is the part selection. We proposed several importance measures and evaluated different approaches to part selection, i.e. minimization of a multi-objective cost function and the divide-and-conquer approach which assumes that different sub-algorithms should perform part selection based on different criteria, and then the results of these sub-algorithms should be merged. Our experiments show that the divide-and-conquer approach leads to substantially better results in object categorization.

Additionally, we explained how to build HoPS, serving as a category descriptor, and how to apply SVM with different kernels for classifying these descriptors. The framework demonstrated sufficiently good results on the Washington RGB-D dataset, and these results were close to state-of-the-art level on this dataset at the time the framework was finished.

This work presented in this section solves the first research goal of this thesis, i.e. it successfully transfers the ideas of hierarchical compositionality to the domain of surface shape features. Additionally, this work confirms the first hypothesis of this thesis, i.e. it shows that the principles of hierarchical compositionality can be used as a basis for the framework capable of learning surface shape features facilitate efficient category recognition from a large dataset.

CHAPTER 6

A SURFACE-BASED COMPOSITIONAL HIERARCHY

In this chapter, a surface-based compositional hierarchical representation is described. As was shown in Chapter 5, the view-based compositional hierarchy and the category recognition system based on it achieve good results for object categorization in the scenario when objects in the training set are approximately in the same positions and orientations w.r.t. to the camera as objects of the testing set. However, this framework does not facilitate the property of geometric invariance, and it is in general case not able (by design) to recognize known shapes at unseen poses. The reason for this is that in the view-based hierarchy each vocabulary part represents a surface shape model of a certain orientation, and there is no mechanism for extrapolating these models to perform recognition from novel views and in-plane rotations[2]. One of the ways of solving this problem is augmenting the training data by presenting each training model under multiple orientations, however, this is computationally inefficient.

There are several advantages of having orientation-independent parts. The first advantage is that these parts can be recognized from the unseen poses, and this property should facilitate robustness of this system to rigid body transformations. The second

[2]This is mainly affected by quantization of surface orientations at the first layer, described in Subsection 5.3.2. For instance, if a "known" surface is rotated, its surface normals change their orientation in the camera-based reference frame, and they are likely to be assigned to another orientational bin, that's why L_1 part realizations change their part IDs. Once this happens, inference of higher layer part realizations becomes impossible (unless the "orientated" surface was presented in the training set as well)

advantage is a much greater compactness of the vocabulary. When vocabulary parts are orientation-independent, it is sufficient to represent each surface shape by a single vocabulary part, while in the view-based hierarchy each surface shape has to be encoded by multiple vocabulary parts representing this surface under different orientations.

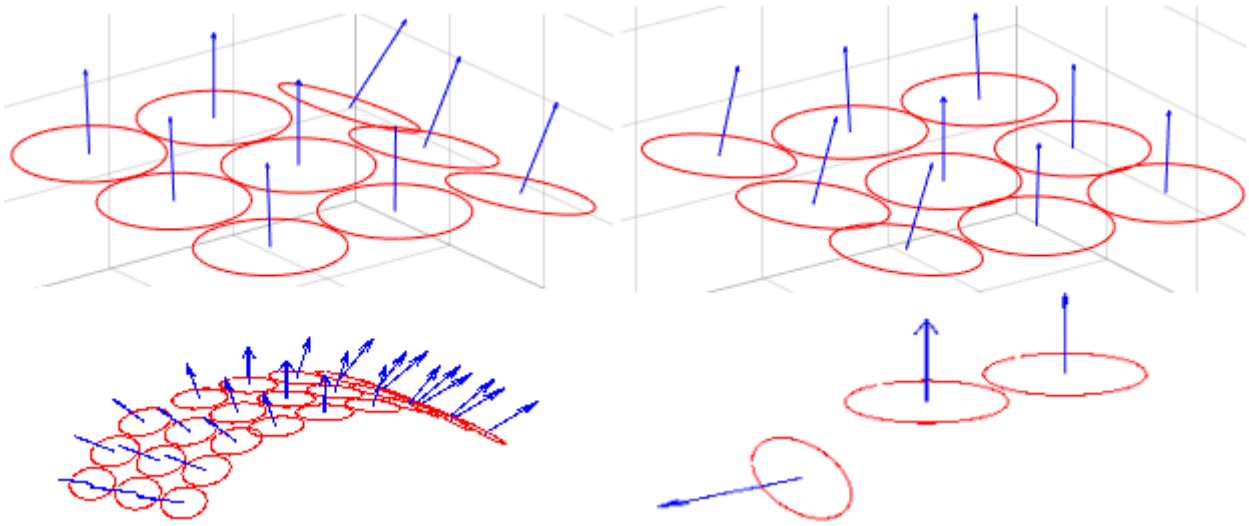


Figure 6.1: Examples of parts of different layers of the surface-based hierarchy

The rest of this chapter is organized as follows. Section 6.1 describes the main design principles on which the surface-based hierarchy is built. Section 6.2 describes pre-processing of input data, while Section 6.3 describes the first layer and shows inference of the first layer from the pre-processed input data. Section 6.4 introduces the composition rules that are used on different layers of the compositional hierarchy. Sections 6.5, 6.6, and 6.7 describe all steps of the vocabulary learning procedure for a single layer $L_n, (n \geqslant 2)$. The most difficult step of the vocabulary learning procedure is clustering of statistical maps, this is why it is presented in a separate section (Section 6.6). Section 6.8 describes inference of a single layer $L_n, (n \geqslant 2)$, while Section 6.9 outlines the pooling procedure. Section 6.10 presents the results of experimental evaluation of the system. Section 6.11 concludes the chapter.

6.1 Design Principles

In this section the main design principles of the surface-based compositional hierarchy are described.

6.1.1 Part-Based Reference Frames

In contrast to the view-based framework (described in Chapter 5), where each vocabulary part represents a surface shape model of a certain orientation, in the surface-based framework vocabulary parts represent orientation-independent shape models. Technically, this means that in the view-based hierarchy the information about surface orientation is encoded in vocabulary parts, while each part realization R carries only the part ID ($R.id$) and the information about the position of this part realization in the data ($R.coord = (x, y, z)$). In contrast, in the surface-based compositional hierarchy the information about the orientation is kept in part realizations, i.e. each part realization R contains not only $R.id$ and $R.coord$ but also the field $R.orient$. Note that, in contrast to the view-based system, where the orientations are quantized, in the surface-based system the orientations of part realizations are represented in the continuous space.

The L_1 vocabulary of the surface-based hierarchy contains only one part representing a small disk-shaped planar surface patch. Since it is not possible to uniquely define a full reference frame (three axes) for a disk-shaped planar surface patch, orientations of L_1 part realizations can only be described by surface normals. As for the part realizations of the subsequent layers $L_n, (n \geqslant 2)$, each of them has its own reference frame, in which the positions and the orientations of the subparts are described. Reference frames of part realizations represent right-handed coordinate systems with three axes.

Quaternions

Orientations of the L_1 part realizations are represented by surface normals, while orientations of realizations of higher layers are represented by reference frames comprising three

axes. Since both these types of parts realizations undergo the same learning and inference algorithms, it would be very convenient to have a unified representation of orientations for both cases, i.e. regardless of how many axes are specified.

Fortunately, there exists a convenient mathematical tool called *quaternions* which provide such functionality. Quaternions are the 4-dimensional vector used for representing orientations[1] and rotations of objects in 3D space. There exist other mathematical tools for representing orientations as well, for example, Euler angles and 3×3 rotation matrices. However, in general, quaternions have certain advantages over other tools, since they are compact, numerically stable, computationally efficient (e.g. very simple to compose), and therefore, they avoid different problems which appear in other representations, for example the gimbal lock property of Euler angles, and the non-compactness (leading to computational inefficiency) of rotation matrices. Due to these reasons we use quaternions as a unified tool for representing both orientations and rotations of parts and part realizations.

6.1.2 Learning and Inference from Triangulated Mesh Models

In contrast to the view-based hierarchy, which performs learning and inference from depth images, representing partial views of objects, the surface-based hierarchy does not have this restriction, and it can learn and infer the vocabulary both from partial views and from full object models. It is very convenient to unify the format of the input data on the pre-processing step, i.e. before the input data undergoes learning and inference. That is why during pre-processing the input data is converted to triangulated mesh models which can be conveniently used for representing both full shape models and partial views.

Since triangulated mesh models are used as input data, the learning and inference algorithms presented in this chapter are to a certain extent based on low-level functionality for mesh processing, for instance, for computing geodesic distances on meshes. In general,

[1] they can be used both for representing orientation of a single vector and orientation of a full reference frame

the functionality for mesh processing is quite complicated and mathematically involved [225]; in addition, most of mesh processing algorithms are typically based on multiple low-level sub-algorithms. Moreover, mesh processing algorithms[1] are typically sensitive to the quality of meshes (e.g. size of triangles) and to different mesh degradations, therefore, the correct handling of multiple specific cases is often required. Generally, the overall complexity of the low-level mesh processing functionality makes it impossible to provide a detailed step-by-step description of all mesh processing algorithms employed in this chapter, which is why we often provide only high-level descriptions.

Note, the low-level mesh processing functionality of the surface-based compositional hierarchical framework is to a large extent based on Gabriel Peyre's mesh processing toolboxes [226], for example the General Toolbox, the Graph Toolbox, and the Fast Marching Toolbox. However, a certain number of the mesh processing functions required for the framework have either been implemented from scratch, or obtained by modifying the functionality provided in the standard toolboxes.

The following functionality have been implements from scratch:

- A function for checking how many times a given ray r intersects a given mesh,
- A function for computing a geodesic circle, centered in a given point on a mesh,
- A function for regularly sampling points on a surface of a given triangle,
- A function computing pariwise geodesic distances between points on a mesh,
- A function for computing areas of all triangles of a mesh,
- A function for triangulating a depth image (more details will be presented in Section 6.2.)

The function *meshFlipFaces* from the toolbox has been modified such that it flips the surface normals pointing inside the object (more details will be presented in Section 6.2.)

[1]For example, the fast marching algorithm which approximately computes geodesic distances on meshes

Also, several utility functions (e.g. for computing adjacency matrices for vertices and faces) have been modified to work with sparse matrices, since it is very costly (in terms of memory) to use full matrices for meshes with large (more than five or six thousand) number of vertices.

6.2 Pre-Processing of the Input Data

Pre-processing of the input data involves several steps. If the input dataset $\mathcal{T}$ is represented by depth images, then these images have to be pre-smoothed at the first stage. We pre-smooth each input image $I = \mathcal{T}(i)$, $(i \in \mathbb{N})$ using the Gaussian filter of the guided image filter [210] as shown in Section 5.2. After that each pre-smoothed depth image I_σ has to be converted to the triangulated mesh model.

The triangulated mesh model is a pair (V, F) where V is a list of vertices, and F is a list of faces. The list of vertices V can be obtained from the pre-smoothed depth image I_σ by taking each pixel $\rho = (x, y) \in \Omega$ from the image domain $\Omega \subset \mathbb{Z}^2$ and computing the corresponding 3D point using the Equations 5.1, 5.2, 5.3 [1]. The list of faces is computed such that the points obtained from neighbouring pixels of a depth image are connected to each other using the following rule. Each four points $\rho_1, \rho_2, \rho_3, \rho_4$ obtained from the neighboring pixels with coordinates (x, y), $(x+1, y)$, $(x+1, y+1)$, $(x, y+1)$ should form two faces, namely the faces (ρ_1, ρ_2, ρ_3) and (ρ_2, ρ_3, ρ_4).

After triangulated mesh models are computed[2], these models undergo several more pre-processing steps, in which regularly sampled points on each face are defined and surface normals for each of these points are computed. These computations are performed in several steps: (i) surface normals for each face $f_i \in F$ are computed (these normals are termed face normals) (ii) orientations of face normals are checked and, if necessary, reversed, (iii) vertex normals for each vertex $v_j \in V$ are computed as the average of

[1] Note, that in this case the coordinates of vertices will be described in the camera-cantered reference frame defined in Section 5.1.1.

[2] Or if the input data is initially represented by a dataset of triangulated mesh models

the face normals of the faces adjacent to the vertex v_j, (iv) regularly sampled points on each face are defined, and (v) normals for these points are computed from the vertex normals using the barycentric coordinates. Next, more details about each of these steps are provided.

At the first step, surface normals for each face of the triangular mesh are computed using the cross product rule. Assume, there is a face f_i connecting points $A \in V$, $B \in V$ and $C \in V$. We compute the vectors $\vec{u_1} = B - A$ and $\vec{u_2} = C - A$, normalize both these vectors, and then compute the surface normal N_i as a cross product of $\vec{u_1}$ and $\vec{u_2}$.

The next step is to check the orientations of face normals to make sure they point outside the object[1]. If the input data was initially represented by range images (representing partial views of objects), face normals should be reversed only if their Z component is positive. However, if the input data was initially represented by the triangulated mesh models of full 3D objects, then the more complex normal orientation test should be performed. This test comprises tracing of a ray originating from the point $\rho_c = \frac{A+B+C}{2}$ (middle point of each face) and pointing to the direction of a face normal N_i. The goal is of this tracing is to check how many times this ray intersects the surface, represented by the mesh (V, F). If there is an even number of intersections (e.g. 0, 2, 4 intersections), then the orientation of N_i is computed correctly, otherwise, the normal N_i should be reversed.

After face normals are computed and their orientations are checked, we compute normals for each vertex $v_i \in V$ by averaging all face normals of the faces adjacent to v_i (as shown in Figure 6.2(a)), i.e. compute $N_x = \frac{1}{n} \sum_{j=1}^{n} N_j$, where N_x is the vertex normal of the vertex v_i, n is the number of faces adjacent to this vertex, while N_j are face normals of these faces. When vertex normals are computed, we define a regular point grid on each face (as illustrated in Figure 6.2(b)). A set of points on a face f_k is denoted $\mathcal{P}(f_k)$. Finally, for each of these points, we compute the point normal using barycentric coordinates. Assume there is a point D on a face connecting the vertices $A \in V$, $B \in V$ and

[1]Some surface normals computed using the cross-product rule point inside the volume.

$C \in V$, vertex normals at these points are N_A, N_B and N_C, and our goal is to compute the normal N_D at the point D (Figure 6.2(c)). This is done using barycentric coordinates (Equation 6.1):

$$N_D = w_1\, N_A + w_2\, N_B + w_3\, N_C, \tag{6.1}$$

where weights w_i are computed using the ratio of areas (denoted S) of the corresponding triangles, i.e. $w_1 = \frac{S(\triangle BCD)}{S(\triangle ABC)}$, $w_2 = \frac{S(\triangle ACD)}{S(\triangle ABC)}$ and $w_3 = \frac{S(\triangle ABD)}{S(\triangle ABC)}$. Note, that since these weights sum up to 1, w_3 can also be computed as $w_3 = 1 - w_1 - w_2$.

In general, for computation of the area of a triangle with sides of length a, b and c, we use Heron's formula (Equation 6.2):

$$S(\triangle) = \sqrt{p(p-a)(p-b)(p-c)}, \tag{6.2}$$

where p is a semi-perimeter, i.e. $p = \frac{a+b+c}{2}$.

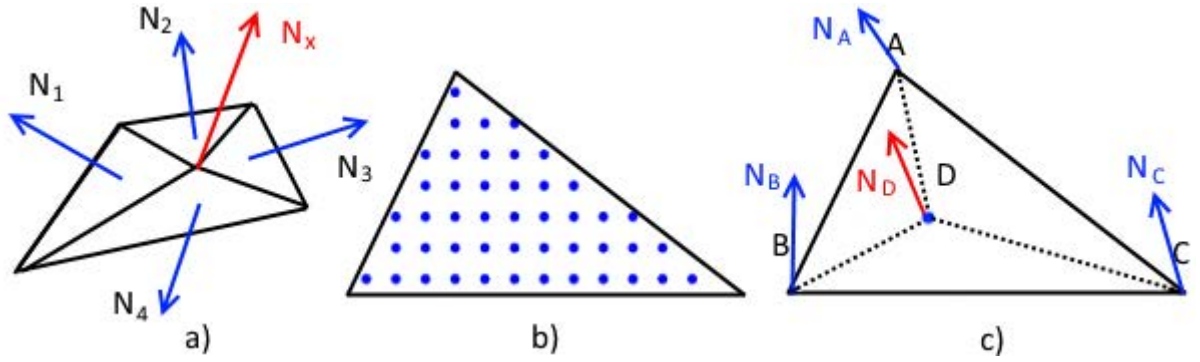


Figure 6.2: Illustration for different pre-processing steps: a) Vertex normal N_x is computed as an average of the face normals of the adjacent faces (N_1, N_2, N_3, N_4), b) Points sampled on a face, c) Illustration for barycentric coordinates.

6.3 First Layer

In Chapter 4 the decision to use small planar surface patches as L_1 parts for both the view-based and the surface-based compositional hierarchies was explained. Since in the surface-based compositional hierarchy each part represents an orientation-independent

shape model, the first layer L_1 of this hierarchy contains only one part representing a small disk-shaped planar surface patch.

6.3.1 Inference of the First Layer

This subsection describes inference of the first layer from a pre-processed dataset of triangulated mesh models $\mathcal{T}_p$. Since there is only one planar part in the L_1 vocabulary, the main step during inference is the planarity test. We define a geodesic circle[1] around each data point ρ on a surface, extract a set of surface points that lies within this circle and perform the planarity test for this set of points. If planarity is confirmed then we have a realization R of a part P_1^1 at the point ρ. The realization R should have the following characteristics: part ID $R.id = 1$, position $R.coord = \rho.coord$ and the orientation $R.orient = \rho.N$ defined by the surface normal[2] at the point ρ. The inference procedure is described in Algorithm 9. The radius of all first layer parts is assumed to be equal to r. The algorithm for defining a geodesic circle centered at the given point ρ is described in Subsection 6.3.2.

6.3.2 Defining a Geodesic Circle

The procedure of defining a geodesic circle centered in a given surface point is called during learning and inference on multiple hierarchical layers, therefore the algorithm described in this subsection will be also used in the next sections.

Assume there is a point $\rho_c \in \mathcal{P}(f_i)$ that lie on a face $f_i \in F$ of the triangulated mesh model (V, F), and our task is to define the geodesic circle of radius r around this point. There are multiple possible scenarios, which are reflected in Figure 6.3. Figure 6.3(a) shows the simplest scenario, where the geodesic circle covers only the area within one triangular face. In this case, the geodesic circle will comprise only the points $\{\rho_j\}_j \subseteq \mathcal{P}(f_i)$

[1]Geodesic circle is a set of points on a two-dimensional manifold whose geodesic distances from a fixed point (circle centre) is a constant

[2]Remember that surface normals for each data point were computed on the pre-processing step described in Section 6.2.

Algorithm 9: Inference of the first layer

Data: List of m triangulated mesh models $\mathcal{T}_p$
Vocabulary of first layer $\mathcal{S}(L_1)$ (planar surface patch of radius r)
Result: Input data represented in terms of the vocabulary of the first layer $\mathcal{T}_{\mathcal{S}(L_1)}$

for $j = 1$ **to** m **do** `// for each mesh`
  $(F_j, V_j) = \mathcal{T}_p(j)$;
  $\mathcal{Q} = \emptyset$;
  foreach $f_i \in F_j$ **do** `// for each face`
    foreach $\rho \in \mathcal{P}(f_i)$ **do** `// for each point on a face`
      find a set of surface points Λ that belong to a geodesic circle of radius r centered at the point ρ;
      perform the **planarity test** for the set Λ;
      if *planarity is confirmed* **then**
        create a realization R: $R.orient = \rho.N$;
        $R.id = 1$;
        $R.coord = \rho.coord$;
        include R to the set $\mathcal{Q}$;
      end
    end
  end
  $\mathcal{T}_{\mathcal{S}(L_1)}(j) = \mathcal{Q}$
end

such that $d_E(\rho_c, \rho_j) \leq r$, where d_E is the Euclidean distance between two points[1]. Figure 6.3(b) shows a more difficult case where the geodesic circle spans several faces adjacent to f_i. In this case, we use a similar algorithm, i.e. we first extract all points from the face f_i and from all faces adjacent to f_i and use the Euclidean distances to approximately define which of these points lie within a geodesic circle.

The most difficult situation is shown in Figure 6.3(c) where the geodesic circle spans a large number of faces, i.e. when the area of a geodesic circle is much larger than the area of the face f_i. In this case, for defining geodesic circles we use the algorithm based on the Fast Marching Algorithm (FMA). The FMA [175] is a numerical algorithm for computing geodesic distances from the set of starting vertices on a mesh (or from a single vertex) to all other vertices. Note, that the FMA is closely related to the Dijkstra's algorithm [227] that finds shortest paths on graphs by explicitly traversing vertices of this graph. In

[1]That means in this case geodesic distances are approximated by the Euclidean distances

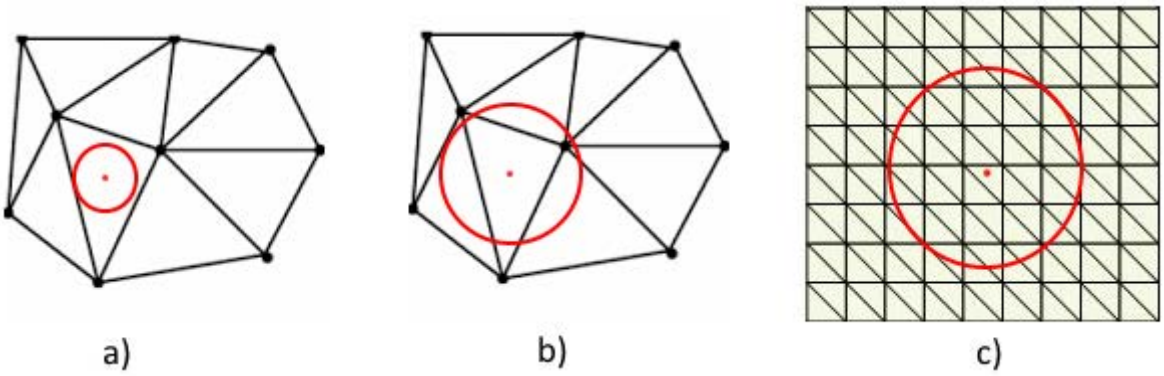


Figure 6.3: Illustration for geodesic circles: a) The geodesic circle spans the area within a single triangle, b) The geodesic circle spans the area covering several neighboring triangles c) The geodesic circle covers a large number of triangles.

contrast, the FMA works substantially faster since it searches for the approximate solution using a gradient descent of the distance function, and typically converges in a relatively small number of iterations. Once geodesic distances from f_i to other mesh vertices are computed using the FMA, a geodesic circle is defined by thresholding these distances, i.e. selecting only the vertices with geodesic distances smaller than r.

6.4 Composition Rules

In general, the composition rules of the surface-based compositional hierarchy are very similar to the rules used in the view-based hierarchy described in Chapter 5. There are, however, three differences between them. First, is that in the surface based-hierarchy the preferable positions of the subparts are defined using the axes of a local reference frame (LRF), not the global one as in the view-based hierarchy. Second, the offsets between subparts are measured by geodesic distances between them, not the Euclidean distance in the XY plane. Third, there are no empty subparts in the surface-based compositional hierarchy.

Figure 6.4 illustrates the main difference between the composition rules of the view-based and the surface-based hierarchies. In the view-based compositional hierarchy (Figure 6.4(a)) part realizations adjacent in the directions of the X (on the layers L_2, L_4,

etc.) and Y (L_3, L_5, etc.) axes are composed. That means the X and Y axes of the global reference frame are used to specify the preferable positions (shown by red circles) of subparts. In principle, in the surface-based compositional hierarchy (Figure 6.4(b)) a very similar strategy is used, however, we use the LRF to compute preferable positions of the subparts.

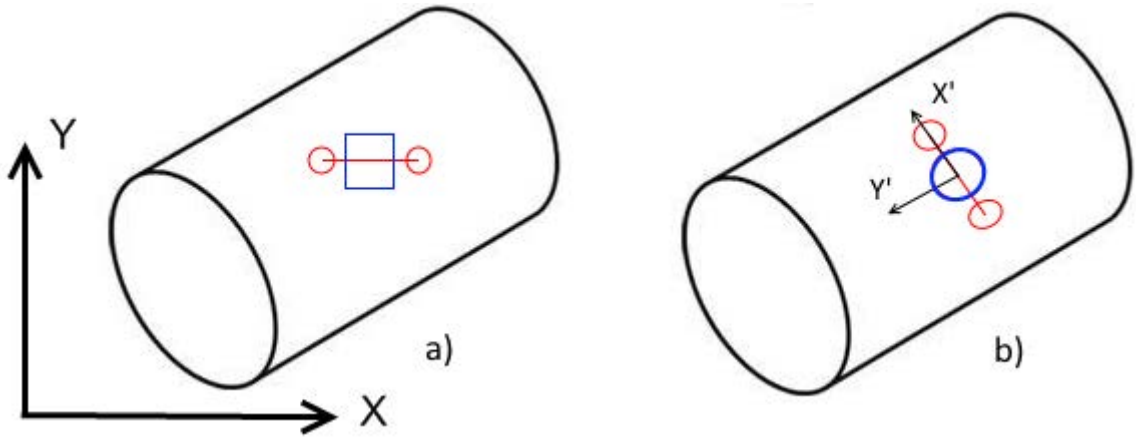


Figure 6.4: Comparison of the composition rules of the view-based and the surface-based compositional hierarchies a) In the view-based compositional hierarchy subparts adjacent in the directions of the X and Y axes are composed b) In the surface-based hierarchy subparts adjacent in the direction of the principal curvatures are composed.

Now we consider where the LRF should be taken from. As was mentioned in Subsection 6.1.1, there are two types of part realizations: realizations having only a surface normal, and realizations having a full reference frame. If a part realization R has a full reference frame, than this reference frame should be used to compute positions of part realizations eligible for composing with R. However, if the realization R has only a surface normal, then the reference frame has to be estimated. To this end, we define the receptive field $\mathcal{F}$, i.e. a geodesic disk centered in R of the radius $3\,r$ (where r is a radius of the support region of R). Then the reference frame for this receptive field is estimated using the principal curvatures (as shown in Subsection 6.4.1), and after that, this estimated reference frame is used for defining the positions of part realizations eligible for composing with R. Note, that the direction of the largest principal curvature always corresponds to the X axes of the LRF.

There are two special cases, in which it becomes impossible to define the directions

of principal curvatures, i.e. when the geodesic circle $\mathcal{F}$ represents the planar surface patch or umbilic surface[1]. In these cases, the axes of the LRF are defined randomly, more specifically, we first randomly generate a unit vector in the tangent plane and then compute another vector in the tangent plane orthogonal to it.

Except for the reference frames, there is another important difference between composition rules of the view-based and the surface-based systems. In the view-based system the preferable offsets between subparts are defined by Euclidean distances in the XY plane. In contrast in the surface based hierarchy the offsets are measured using geodesic distances. However, since computation of geodesic distances of a triangulated mesh is a computationally expensive operation, we use the following approximation. If the receptive field represents the surface of low curvature (i.e. a surface close to the planar surface), then geodesic distances are approximated by the Euclidean distances, computed in 3D space. However, if the receptive field represent the surface of high curvature, we have to use the FMA for computing geodesic distances.

6.4.1 Local Reference Frame

This subsection describes the computation of the local reference frame for a given receptive field. Assume there is a receptive field $\mathcal{F}$ representing a geodesic circle centered at the point ρ_c on a surface. The surface normal at the point ρ_c is N_c. Also assume there is a set of surface points $\Lambda = \mathcal{P}(\mathcal{F})$ located within the receptive field $\mathcal{F}$.

For computation of the Darboux frame we use our modification of the algorithm proposed by Gabriel Taubin [228]. Taubin proposed to use eigenvectors of the 3-dimensional *curvature tensor* M_ρ at the point ρ as an estimate of the Darboux frame. The curvature tensor can be computed using the Equation 6.3:

$$M_\rho = \sum_{\rho_i \in \Lambda} w_i \, k_{ci} \, T_{ci} \, T_{ci}^T, \tag{6.3}$$

[1]For example, a surface patch on a sphere

where T_{ci} is a tangent vector, computed as a normalized projection of the vector $(\rho_i - \rho_c)$ onto the tangent plane of the surface $\mathcal{F}$ at the point ρ_c, k_{ci} is the approximation of the directional curvature $k(T_{ci})$, while w_i is a weight of each point ρ_i, which can be made inversely proportional to that point's density estimate. The normalized tangent vectors T_{ci} can be computed using the Equation 6.4:

$$T_{ci} = \frac{(I - N_c\, N_c^T)(\rho_c - \rho_i)}{\|(I - N_c\, N_c^T)(\rho_c - \rho_i)\|}, \tag{6.4}$$

where I is a 3×3 identity matrix, while the approximation k_{ci} of the directional curvature $k(T_{ci})$ is computed using the Equation 6.5:

$$k_{ci} = \frac{2\, N_c^T\, (\rho_i - \rho_c)}{\|\rho_c - \rho_i\|^2}. \tag{6.5}$$

The original method of Taubin has a substantial drawback when it is applied to triangulated mesh models. This drawback is that the method uses the positions of surface points $\rho_i \in \Lambda$, and does not consider the orientations of surface normals in these points. Consequently, this algorithm fails to correctly compute the principal curvatures in the case if the receptive field $\mathcal{F}$ is located only within a single face of the mesh. Let us consider the example shown in Figure 6.5 (a), in which the receptive field $\mathcal{F}$ (shown as a red circle) spans only surface points within a single face. In this case all points $\rho_i \in \Lambda$ will be located within the same plane, which makes the computation of the principal curvatures impossible.

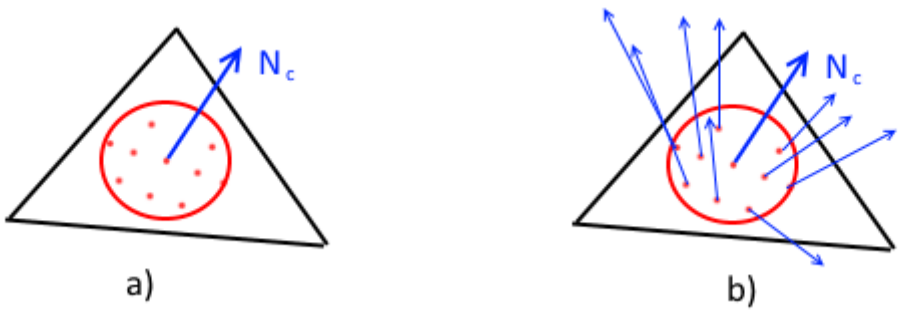


Figure 6.5: Computation of the curvature tensor using surface points and surface normals: a) Only surface points are used b) Surface normals at these points are employed

A better idea would be to use the surface normals in each point $\rho_i \in \Lambda$ to compute directional curvatures, since these normals are interpolated within each face on the pre-processing step, and therefore, they will provide a more robust estimation of principal curvatures. We propose Equation 6.6, which is theoretically equivalent to Equation 6.5 however, in practice it leads to much more robust results since surface normals (in contrast to surface points) are interpolated on the pre-processing step. The proposed equation looks as follows:

$$k_{ci} = \frac{\sqrt{2\left(1 - N_c^T N_i\right)}}{\|\rho_c - \rho_i\|}, \tag{6.6}$$

where N_i is a surface normal at the point ρ_i. Note, that vectors N_i and N_c must be normalized before computing k_{ci}.

After the curvature tensor M_ρ at the point ρ is computed, the Darboux frame, comprising the surface normal at the point ρ and two principal directions T_1 and T_2, can be estimated using eigenvectors of this tensor, while the principal curvatures k_1 and k_2 can be computed from the eigenvalues of M_ρ using very simple linear equations shown in [228].

6.5 Vocabulary Learning

This section describes different steps of the algorithm for vocabulary learning of a single layer L_n, $(n > 1)$. A high-level description of this algorithm was presented in Section 4.2.2, while in this section we provide more details on each step.

6.5.1 Collecting of Co-occurrence Statistics

The first step of learning the shape vocabulary $\mathcal{S}(L_n)$ of a single layer $L_n, (n \geqslant 2)$ is collecting the statistics of co-occurrences of part realizations of the previous layer L_{n-1} and representing these statistics in the form of statistical maps.

Statistical map $\mathcal{M}^n_{i,j} : \mathbb{Z}^7 \rightarrow \mathbb{N}$ is a function showing how many times realizations of the part P^n_j are observed in the training data at the certain relative position and relative orientation w.r.t. realizations of the part P^n_i. For instance, the equation $\mathcal{M}^n_{i,j}(x, y, z, q_1, q_2, q_3, q_4) = 25$ means that realizations of the part P^n_j appeared 25 times at the relative position defined by the arguments x,y, and z and the relative orientation defined by the arguments q_1, q_2, q_3, and q_4 with respect to realizations of the part P^n_i.

Note, that the statistical maps are discrete functions, while relative positions and relative orientations (described by quaternions) are continuous variables. That is why, before including the relative positions and orientations in a statistical map, they should be re-scaled (using the parameters s_t and s_q) and then rounded to the closest integer.

Remember that when learning the vocabulary of the layer L_n the statistical maps for each pair of parts from the previous layer L_{n-1} should be built, thus if $\mid \mathcal{S}(L_{n-1}) \mid = m$ then m^2 statistical maps should be built. However, most of the statistical maps remain empty during learning since many shapes never co-occur in real-world objects. Algorithm 10 describes the process of building the statistical maps.

Note that in this algorithm (as well as in some of the following algorithms) we have a step described as "Define the LRF $\mathcal{R}$ centered in $R.coords$". What is actually meant by that was explained in Section 6.4, however we have to make it absolutely clear. If the part realization R has a reference frame, then $\mathcal{R} = R.orient$, i.e. we take a reference frame of this realization. However, if the part realization R does not have a reference frame (in case if it has only a surface normal), then the reference frame $\mathcal{R}$ is estimated using the curvature tensor of the receptive field, as was shown in Subsection 6.4.1.

6.5.2 Transformation from Global to Local Reference Frame

In Algorithm 10 we converted the position of each part realization $K \in \Lambda$ from the global coordinate system to the LRF and computed the quaternion $q_k = (q_1, q_2, q_3, q_4)$ describing the orientation of K in this LRF. In this subsection we present the equations for these computations. There are two main scenarios: (i) when orientations of part realizations

Algorithm 10: Collecting co-occurrence statistics and building statistical maps

Data: Training set $\mathcal{T}_{\mathcal{S}(L_{n-1})}$ of the length m represented in terms of the vocabulary $\mathcal{S}(L_{n-1})$
Set of composition rules $\mathcal{B}$
Size of the vocabulary of the layer L_{n-1}: $\mid \mathcal{S}(L_{n-1}) \mid = m$
Discretization parameters s_t and s_q

Result: A set of statistical maps $\{\mathcal{M}_{i,j}^{n-1}\}_{1 \leq i,j \leq m}$

Make all statistical maps $\{\mathcal{M}_{i,j}^{n-1}\}_{1 \leq i,j \leq m}$ empty;
for $j = 1$ **to** m **do** `// for each training model`
  $\mathcal{Q} = \mathcal{T}_{\mathcal{S}(L_{n-1})}(j)$; `// set of` L_{n-1} `realizations`
  foreach $R \in \mathcal{Q}$ **do** `// for each part realization`
    Define the LRF $\mathcal{R}$ centered in $R.coords$;
    Find a set $\Lambda \subset \mathcal{Q}$ of part realizations eligible for composing with R under a set of composition rules $\mathcal{B}$;
    foreach $K \in \Lambda$ **do** `// for each realization`
      compute position (x, y, z) of K in the LRF $\mathcal{R}$;
      compute orientation $q = (q_1, q_2, q_3, q_4)$ of K in the LRF $\mathcal{R}$;
      $x = round(\frac{x}{s_t})$;
      $y = round(\frac{y}{s_t})$;
      $z = round(\frac{z}{s_t})$;
      for $k = 1$ **to** 4 **do** `// for each quaternion value`
        $q_k = round(\frac{q_k}{s_q})$;
      end
      $\mathcal{M}_{R.id,K.id}^{n-1}(x, y, z, q_1, q_2, q_3, q_4) = \mathcal{M}_{R.id,K.id}^{n-1}(x, y, z, q_1, q_2, q_3, q_4) + 1$;
    end
  end
end

are represented only by surface normals, and (ii) when orientations of part realizations are represented by full local reference frames.

For the first case let us consider the example shown in Figure 6.6(a). Assume there are two part realizations R and K with positions $p_r = (x_r,\, y_r,\, z_r)^T$ and $p_k = (x_k,\, y_k,\, z_k)^T$ in the global reference frame. The orientations of these part realizations in the global reference frame are described by the vectors N_r and N_k. Assume we defined the receptive field $\mathcal{F}$, centered at the point p_r. Then we have to compute the (LRF) $\mathcal{R}$ for this receptive field. This LRF (shown in Figure 6.6(a) by blue axes) comprises the axes X', Y' and Z', where the unit vector X' has the coordinates $(X'_x,\, X'_y,\, X'_z)$ in the global reference frame, the unit vector Y' has the coordinates $(Y'_x,\, Y'_y,\, Y'_z)$, while the unit vector $Z' = N_r$ has the

coordinates (Z'_x, Z'_y, Z'_z). Our task is to compute the position of the point p_k in this LRF, and the quaternion describing the orientation of N_k in this LRF. Assume all normals and axes have a unit length.

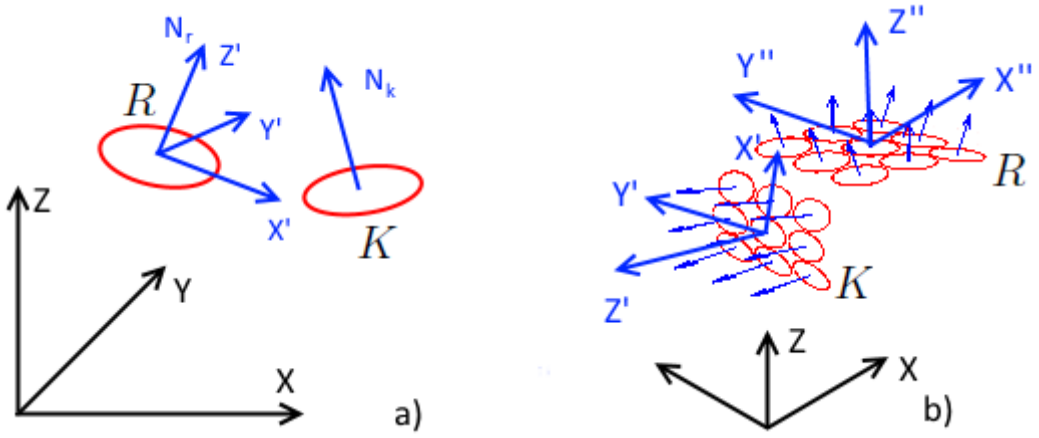


Figure 6.6: Transformation from the global to local reference frame. a) Transformation of part realizations having only one axis, b) Transformation of part realizations having full reference frame.

Since the transformation from one reference frame to another one is a standard procedure described in many textbooks, we do not provide detailed explanations (for example, for the *homogeneous coordinates*); instead, we only show the main equations. We define a 4×4 transformation matrix M_{RT}, which can be computed from the corresponding translation and the rotation matrices M_T and M_R, i.e.

$$M_{RT} = M_R^{-1} \, M_T, \tag{6.7}$$

where

$$M_T = \begin{bmatrix} 1 & 0 & 0 & -x_r \\ 0 & 1 & 0 & -y_r \\ 0 & 0 & 1 & -z_r \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad M_R = \begin{bmatrix} X'_x & Y'_x & Z'_x & 0 \\ X'_y & Y'_y & Z'_y & 0 \\ X'_z & Y'_z & Z'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.8}$$

After the transformation matrix M_{TR} is computed using Equation 6.7, the point $p_k = (x_k, y_k, z_k)$ should be converted to the *homogeneous coordinates* by adding the 4-th

dimension, i.e. the point becomes represented by a vector $H_k = (x_k, y_k, z_k, 1)^T$. After that, this 4-dimensional vector is multiplied with the matrix M: $H'_k = M_{TR}\, H_k$. Finally, the 4-th dimension is removed from the point H'_k, and the resulting 3-dimensional point p'_k represents the coordinates of the point p_k in the LRF.

The next task is to represent the orientation of the normal N_k in the LRF. To this end we have to project the unit vector N_k to the unit vectors X', Y' and Z', i.e., in fact, to compute the dot product of N_k with the vectors X', Y' and Z'. After that, we get a vector N'_k representing N_k in the LRF.

$$N'_k = (N_k \cdot X', N_k \cdot Y', N_k \cdot Z') \tag{6.9}$$

Finally, we have to compute the quaternion representing the orientation of N'_k in the LRF. Note, that N_r is represented by the vector $N'_r = (0, 0, 1)$ in this LRF, therefore the quaternion q can be computed in the following steps:

- compute the cross product: $(q_1, q_2, q_3) = N'_k \times N'_r$,
- compute the dot product: $q_4 = N'_k \cdot N'_r$,
- The resulting quaternion is $q = (q_1, q_2, q_3, q_4)$. [1]

Let us now consider the more difficult case, where orientations of both part realizations R and K are represented by the full reference frames, as shown in Figure 6.6(b). Assume, there are two part realizations R and K located at the points $p_r = (x_r, y_r, z_r)^T$ and $p_k = (x_k, y_k, z_k)^T$ (coordinates are given in the global reference frame). Each of these part realizations has its own reference frame, i.e. $\mathcal{R}_R$ (X", Y", Z") and $\mathcal{R}_K$ (X', Y', Z') correspondingly. For the algorithm 10 we have to find the position of K in the reference frame $\mathcal{R}_R$, and the quaternion q describing the orientation of $\mathcal{R}_K$ relative to $\mathcal{R}_R$.

[1] Notice, that this algorithm does not work correctly only when $N'_k \approx -N'_r$. However, since this situation is impossible for the experimental settings, presented in this thesis, the described way of computing q_k works in all situations. The way of handling the situation when $N'_k \approx -N'_r$ is to choose another rotational axis (other than $N'_k \times N'_r$).

As for the position of p_k in the LRF $\mathcal{R}_R$, it is computed exactly in the same way, as it was done in the previous case, i.e. by defining the translation and rotation matrices, composing them to a single transformation matrix M_{RT}, and using this matrix for computing coordinates of a point in the LRF. The quaternion q showing the orientation of $\mathcal{R}_K$ relative to $\mathcal{R}_R$ is computed in the following way.

First, we define a 3×3 rotational matrix called the directional cosine matrix (DCM). This matrix is computed as shown in the Equation 6.10:

$$M_{DCM} = \begin{bmatrix} X" \cdot Z' & Y" \cdot Z' & Z" \cdot Z' \\ X" \cdot X' & Y" \cdot X' & Z" \cdot X' \\ X" \cdot Y' & Y" \cdot Y' & Z" \cdot Y' \end{bmatrix}. \tag{6.10}$$

Once the DCM is computed, the quaternion q can be computed from this matrix. There are several equivalent ways of doing that. One of the ways is described by the following equations:

$$q_4 = \sqrt{1 + M_{DCM}(1,1) + M_{DCM}(2,2) + M_{DCM}(3,3)}; \tag{6.11}$$

$$q_1 = \frac{M_{DCM}(2,3) - M_{DCM}(3,2)}{2\,q_4}; \tag{6.12}$$

$$q_2 = \frac{M_{DCM}(3,1) - M_{DCM}(1,3))}{2\,q_4}; \tag{6.13}$$

$$q_3 = \frac{M_{DCM}(1,2) - M_{DCM}(2,1))}{2\,q_4}; \tag{6.14}$$

Note, that in order to avoid numerical inaccuracy (e.g. when the denominator is close to zero) other ways can be used. We exploit Jay St. Pierre's Quaternion Toolbox[1] for this operation, where he first computes the trace of the DCM and then chooses one of three equivalent ways of computing the quaternion based on the trace value. This approach helps to avoid numerical inaccuracies. Note, that the quaternion q should be normalized

[1]The Quaternion Toolbox is available for downloading at the Mathworks website (https://uk.mathworks.com/matlabcentral/fileexchange/1176-quaternion-toolbox), although, according to the best of my knowledge, there is no scientific paper, describing this Toolbox

after being computed.

6.6 Clustering of the Statistical Maps

In contrast to the view-based compositional hierarchical system where clusters in statistical maps are always well-separated, clustering of statistical maps in the surface-based system represents a much more difficult problem. The main reason for this is that the information in statistical maps of the surface-based system is **much richer**, and therefore, poor separability of clusters happens very often. This can be understood considering the example of L_2 vocabulary learning. In the view-based hierarchy the first layer vocabulary comprises 81 parts, representing differently orientated planar surface patches, therefore when learning L_2, we form 81×81 statistical maps reflecting the statistics of co-occurrences of these patches. On the other hand, the surface-based hierarchy contains only one part of the first layer, therefore, when learning the layer L_2 we will have only one statistical map. Broadly speaking, this single statistical map contains approximately the same information as 81×81 statistical maps of the view-based hierarchy.

We use the weighted agglomerative hierarchical clustering with average linkage function for clustering of the statistical maps in the surface-based system. There are two main reasons for the choice of the agglomerative hierarchical clustering in preference of other methods of clustering. First, agglomerative hierarchical clustering is very flexible since various distance metrics and linkage functions can be used within this method (more details will be discussed in Subsection 6.6.1). Second, agglomerative hierarchical clustering results in a *dendrogram*, i.e. a tree-like diagram illustrating the arrangement of the clusters. This structure can be very conveniently used for cluster separability analysis, and for the selection of the optimal number of clusters.

In the following subsections we describe the agglomerative hierarchical clustering and the linkage functions (Subsection 6.6.1), introduce the distance function for entries of statistical maps (Subsection 6.6.3), and present our approach for computing the opti-

mal number of clusters (Subsection 6.6.4), and describe parameterization of clusters in Subsection 6.6.5.

6.6.1 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering is represented by a number of algorithms which start by placing each input element in its own cluster, and proceed by repeatedly merging the closest (according to the chosen distance measure) pair of clusters until all clusters are merged into a single cluster.

The main difference between multiple agglomerative hierarchical clustering algorithms lies in a way of computing distances between clusters, i.e. employed *linkage functions* that map pairs of clusters to non-negative real numbers. Formally, a linkage function can be defined as $\mathcal{L} : \{(\mathcal{X}_1, \mathcal{X}_2, d, w) \mid d, w \, over \, \mathcal{X}_1 \cup \mathcal{X}_2\} \rightarrow \mathbf{R}+$, where $\mathcal{X}_1$ and $\mathcal{X}_2$ are clusters, and w are weights of each input elements and d are distances between input elements [214].

There are several popularly employed linkage functions, for instance, *complete linkage* $\mathcal{L}_c = \max\{\, d(x,y) \, : \, x \in \mathcal{X}_1, \, y \in \mathcal{X}_2 \,\}$, where distance between clusters is assumed to be equal the maximal distance between points from both clusters, and *single linkage* $\mathcal{L}_s = \min\{\, d(x,y) : x \in \mathcal{X}_1, \, y \in \mathcal{X}_2 \,\}$, where distance between two clusters is assumed to be represented by the smallest distance between points from both clusters. However, results of both of these linkage functions are unaffected by changes of element weights, which makes them not suitable for weighted clustering.

Average linkage is one of the most popular linkage functions. It computes distance between clusters as the average distance between all elements in these clusters, as shown in Equation 6.15:

$$\mathcal{L}_a = \frac{1}{|\mathcal{X}_1| \cdot |\mathcal{X}_2|} \sum_{x \in \mathcal{X}_1} \sum_{y \in \mathcal{X}_2} d(x,y). \tag{6.15}$$

In contrast to complete linkage and single linkage, the average linkage function can be made weight-sensitive [214], in which case it takes the form presented in the Equation

6.16:

$$L_{aw} = \frac{\sum_{x \in \mathcal{X}_1} \sum_{y \in \mathcal{X}_2} d(x,y)\, w(x)\, w(y)}{w(\mathcal{X}_1)\, w(\mathcal{X}_2)} \tag{6.16}$$

where $w(\mathcal{X}_i)$ is the weight of the cluster $\mathcal{X}_i$, which is computed as a sum of weights of all points in this cluster the using the Equation 6.17:

$$w(\mathcal{X}) = \sum_{x \in \mathcal{X}} w(x) \tag{6.17}$$

6.6.2 Input Data for Clustering

After describing the average linkage function we show how to apply this function to clustering of the statistical maps. The weighted average linkage function shown in Equation 6.16 requires specifying the list of input entries, weights of these entries, and pairwise distances between them. Note, that while the entries and their weights can be obtained directly from the statistical maps, defining distances between these entries is a more difficult task discussed in Subsection 6.6.3.

Assume, there is a statistical map $\mathcal{M}_{i,j}^{n-1}$ with m non-zero values. Each non-zero value of this map will be considered as an entry B_k, $(k \in [1,m])$ which is characterized by the position $p_k = (x,y,z)^T$, the orientation $q_k = (q_1,q_2,q_3,q_4)^T$, and the weight w_k showing the number of occurrences of realizations of the part P_j^{n-1} at the relative position p_k and relative orientation q_k from realizations of the part P_i^{n-1}. For example, if there is a non-zero value of the statistical map $\mathcal{M}_{i,j}^{n-1}(x_s,y_s,z_s,q_{s1},q_{s2},q_{s3},q_{s4}) = a$, where $(a > 0)$, then we can get the entry B_k characterized by position $p_k = s_t\,(x_s,y_s,z_s)^T$, orientation $q_k = s_q\,(q_{s1},q_{s2},q_{s3},q_{s4})^T$ and the weight $w_k = a$.

After defining the input entries and their weights, we have to specify distances $d(B_k,B_l)$ between each pair of entries B_k and B_l, $(k,l \in [1,m])$.

6.6.3 Pairwise Distance Between Entries

As entries represent translations and orientations, their similarity measure should comprise both of them. We use a distance measure which is closely related to the measure proposed by Kuffner *et al.* [229]. This measure represents a weighted sum of the Euclidean distance between two points and the orientational distance between two quaternions. The distance $d_{ER}(B_h, B_k)$ between two entries B_h and B_k, representing positions p_h and p_k and orientations q_h and q_k, is defined using Equation 6.18:

$$d_{ER}(B_h, B_k) = d_E(p_h, p_k) + \alpha\, d_R(q_h, q_k), \tag{6.18}$$

where $d_E(\cdot, \cdot)$ is the Euclidean distance between positions and $d_R(\cdot, \cdot)$ is a rotational distance between quaternions q_h and q_k. The rotational distance is defined using Equation 6.19:

$$d_R(q_h, q_k) = \sqrt{1 - \mid q_h\, q_k \mid}. \tag{6.19}$$

If the quaternions q_h and q_k are equal, then $d_R(q_h, q_k) = 0$, while if the quaternions q_h and q_k represent opposite orientations, then $d_R(q_h, q_k) = 1$. The parameter α in Equation 6.18 is chosen to make put the rotational and translational distances within each receptive field approximately to the same range. Since q_h always produces values in range $[0, 1]$, while translational distances can be in the range $[0, 2\,r]$, where r is the radius of the receptive field. That is why the weighting parameter α can be made $\alpha = 2\,r$ in order to make both distance measures produce the same range of values.

Note, that all quaternions must be normalized before computing d_R. The norm of a quaternion $\mid q \mid$ is found using Equation 6.20:

$$\mid q \mid = \sqrt{\sum_{i=1}^{4} q_i^2} \tag{6.20}$$

6.6.4 Optimal Number of Clusters

In the previous subsection we introduced everything needed for performing agglomerative hierarchical clustering using the weighted average linkage function (described by Equation 6.16) i.e. the input entries, their weights, and distances between them. This allows computing the dendrogram, showing arrangements of clusters. An example of the dendrogram for the layer L_2 is shown in Figure 6.7(a), while a possible result of clustering using the dendrogram is shown in 6.7(b) [1].

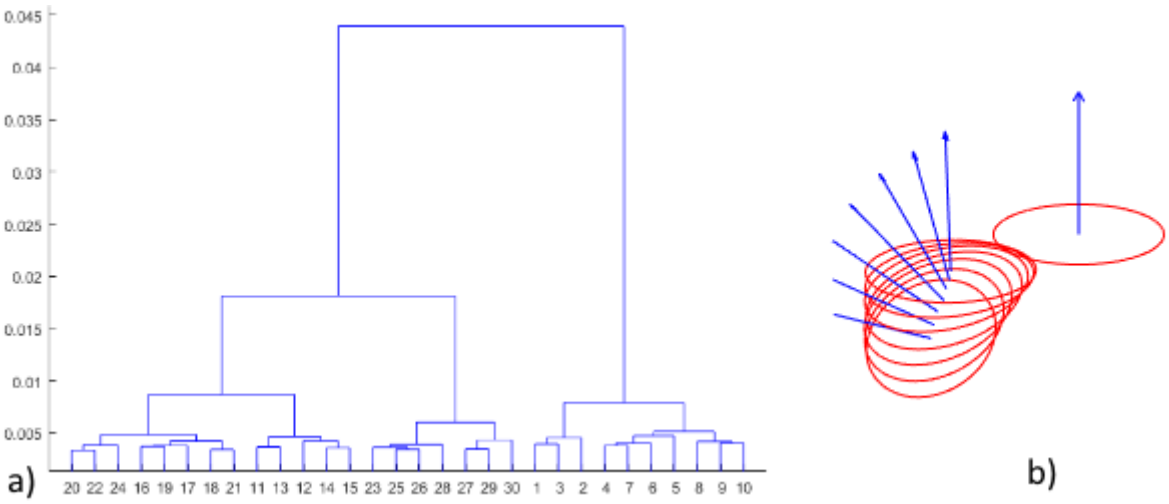


Figure 6.7: Illustration for clustering of statistical a map: a) Dendrogram b) Resulting clusters. The figure shows the central subpart (on the right) and other subparts, in the positions of cluster centres (on the left)

After the dendrogram is built, the decision about the optimal number of clusters should be made. Making different numbers of clusters from a dendrogram can be understood as cutting off a dendrogram at different distance levels (i.e. along the vertical axis). If the dendrogram presented in Figure 6.7(a) is cut at distance 0.05, we will have a single cluster, at the level 0.03 - two clusters, 0.01 - three clusters, etc. Since the optimal number of clusters is unknown, we have to use certain criteria to compute this number.

Our first choice was the Davies – Bouldin index [230] (DBI), which is defined using the Equation 6.21:

[1]Note, that this dendrogram has been computed from the statistical map collected from a very small dataset (6 models), while if the vocabulary is learned from a large dataset, the dendrograms typically link a much larger numbers of input entries (sometimes 3 or 4 thousand entries).

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d_{ER}(c_i, c_j)} \right), \tag{6.21}$$

where σ_i is the average distance of entries of the cluster $\mathcal{X}_i$ to the centroid c_i of this cluster, while $d_{ER}(c_i, c_j)$ is the distance between centroids c_i and c_j of clusters $\mathcal{X}_i$ and $\mathcal{X}_j$, i.e.

$$c_i = \frac{\sum_{x \in \mathcal{X}_i} x\, w(x)}{w(\mathcal{X}_i)} \tag{6.22}$$

where $w(\mathcal{X}_i)$ is defined in Equation 6.17, while

$$\sigma_i = \frac{\sum_{x \in \mathcal{X}_i} d_{er}(x, c_i)\, w(x)}{w(\mathcal{X}_i)} \tag{6.23}$$

Davies and Bouldin [230] proposed to measure this index for clusterings with different numbers of clusters, and define the optimal number of clusters as the one minimizing the DBI. Note, however, that there are two degenerate cases which may appear when using the DBI. First, when each input entry is placed in its own cluster, in which case all σ_i become equal to zero, end the DBI becomes equal to zero as well. Second, when all input entries are placed to a single cluster, then distances $d(c_i, c_j)$ become undefined. To avoid these problems we introduce an additional constraint specifying the maximal size of clusters (typically $0.25r$ where r is a radius of the receptive field).

The examples of distributions within some clusters are shown in Figure 6.8.

6.6.5 Parameterization of Clusters

We parameterize each cluster with two parametric distributions, namely the 3-dimensional Gaussian representing the variance in position, and the 4-dimensional Bingham distribution representing the variance of the orientations. The Bingham distribution [231] is an antipodally symmetric probability distribution on the n-dimensional sphere. In our case $n = 4$, since quaternions are 4-dimensional vectors. The Bingham distribution over unit

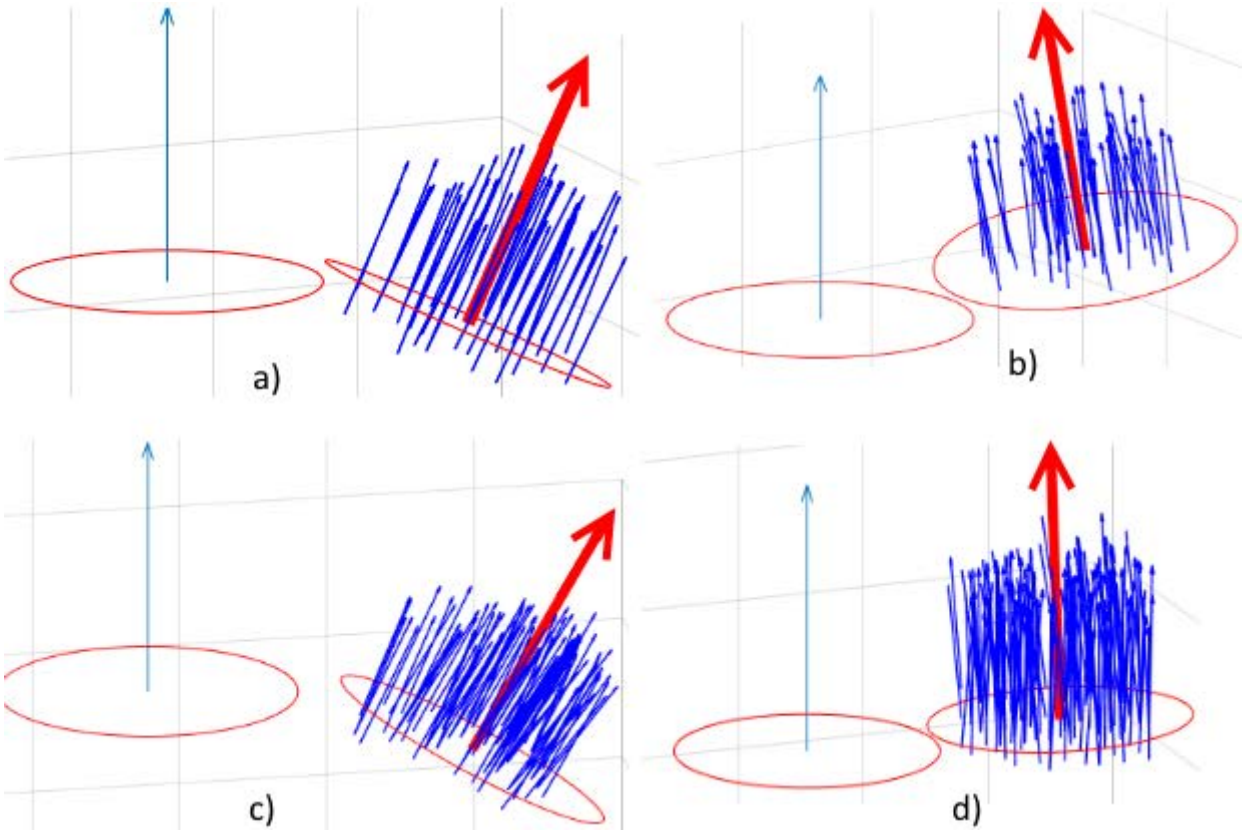


Figure 6.8: Examples of distributions within clusters. Circles on the left side of each picture show the central subparts. Circles on the right from the central subpart show other subparts in the positions of cluster centres. Bold red vectors show the average orientation of this cluster. Blue thin vectors show positions and orientations of non-empty entries of the statistical map, which were assigned to the cluster.

quaternions has been used by multiple authors [232, 233, 234], that is why we do not provide many details about it. The Bingham distribution has four parameters, namely q^a which is a unit vector, equal to the average quaternion (see Equation 6.24), M^q is an orthogonal orientation matrix, Z^q is a diagonal concentration matrix, and the parameter F^q.

Therefore, in contrast to the view-based hierarchy, where each cluster is represented by two parameters μ_i and Σ of the Gaussian distribution (i.e. $\Delta_i^{sp} = (\mu_i, \Sigma_i)$) in the surface-based hierarchy a cluster description looks as follows: $\Delta_i^{sp} = (\mu_i, \Sigma_i, q_i^a, M_i^q, Z_i^q, F_i^q)$. This description comprises 32 float numbers altogether.

Note, that for estimating the parameters of the Bingham distribution we used the

functionality from the Sebastian Riedel's Bingham Toolbox for Matlab [234][1]. Also notice, that averaging of quaternions can not be done using the arithmetic mean. Assume there are n quaternions q_j, where $j \in [1, n]$. Then the average quaternion q^a can be found as the eigenvector, corresponding to the largest eigenvalue of the 4×4 matrix defined by the Equation 6.24 [235]:

$$q^a = \frac{1}{n} \sum_{j=1}^{n} q_j \, q_j^T; \tag{6.24}$$

6.7 Vocabulary Learning (continuation)

6.7.1 Inference of Doublets from the Training Data

After the statistical maps are clustered, we create a set of doublets $\mathcal{D}(L_n)$ from these clusters, such that each cluster from each statistical map is used to create a doublet. For instance, a cluster Δ^{sp} from the statistical map $\mathcal{M}_{ij}^{n-1}$ should result in a doublet $(P_i^{n-1}, (P_j^{n-1}, \Delta^{sp}))$. After a set of doublets is formed, we have to infer doublets from the training data. Inference of doublets from the training data is described in Algorithm 11. Note, that the activation test actually comprises two tests, namely the test showing if the position of a part realization belongs to a given Gaussian distribution and the test showing if the orientation of this realization belongs to the Bingham distribution. The activation test is successful only if both these conditions are fulfilled.

Notice that doublets of the layers $L_n, (n > 2)$ inherit the reference frames from their central subparts. As for the doublets of the L_2, they receive the reference frame, computed from the curvature tensor of the receptive field as shown in Section 6.4.

[1] According to the best of my knowledge, there is no separate paper describing the toolbox itself, that is why I cite the related paper where the author (Dr. Riedel) applies the Bingham mixture model for parameterizing unit quaternions.

Algorithm 11: Inference of doublets of the layer $L_n, (n \geqslant 2)$

Data: Training dataset $\mathcal{T}_{\mathcal{S}(L_{n-1})}$ of the length m represented in terms of the vocabulary $\mathcal{S}(L_{n-1})$
Set of composition rules: $\mathcal{B}$
Doublets of the layer L_n : $\mathcal{D}(L_n)$

Result: Training data $\mathcal{T}_{\mathcal{D}(L_n)}$ represented in terms of the doublets $\mathcal{D}(L_n)$

$\mathcal{T}_{\mathcal{D}(L_n)}$ is empty;
for $j = 1$ **to** k **do** `// for each model`
  $\mathcal{Q} = \mathcal{T}_{\mathcal{S}(L_{n-1})}(j)$; `// set of` L_{n-1} `realizations`
  $\mathcal{DD}$ is empty ; `// set of` L_n `doublets activations`
  foreach $R \in \mathcal{Q}$ **do** `// for each part realization`
    Define the LRF $\mathcal{R}$ centered in $R.coords$;
    find a set $\Lambda \subseteq \mathcal{Q}$ of part realizations eligible for compositing with R under a set of composition rules $\mathcal{B}$;
    find all doublets $\mathcal{D}_s(L_n) \subseteq \mathcal{D}(L_n)$ having the central subpart $P^{n-1}_{R.id}$;
    foreach $K \in \Lambda$ **do**
      find all doublets $\mathcal{D}_{ss}(L_n) \subseteq \mathcal{D}_s(L_n)$ having the non-central subpart $P^{n-1}_{K.id}$;
      if $\mathcal{D}_{ss}(L_n) \neq \emptyset$ **then**
        compute position p_k of K in the LRF $\mathcal{R}$;
        compute orientation q_k of K in the LRF $\mathcal{R}$;
        foreach $\pi^n_{cur} = (P^{n-1}_{R.id}, (P^{n-1}_{K.id}, \Delta^{sp}_{cur})) \in \mathcal{D}_{ss}(L_n)$ **do**
          Perform **activation test** to check if p_k and q_k fits the distribution Δ^{sp}_{cur};
          if *activation test is successful* **then**
            $RR.id = cur$; `//` create a realization of doublet π^n_{cur}
            $RR.coord = R.coord$;
            $RR.orient = \mathcal{R}$;
            include RR to the set $\mathcal{DD}$;
          end
        end
      end
    end
  end
  $\mathcal{T}_{\mathcal{D}(L_n)}(j) = \mathcal{DD}$
end

6.7.2 Co-activations of Doublets

After inference of doublets of the layer L_n from the training data is complete, we obtain activations of doublets in different data points. The next step is to form a set of candidate parts $\mathcal{C}(L_n)$ using the statistics of co-activations of doublets in the training data. There

exist several possible cases depending on how many doublets are simultaneously activated in the same data point. Assume there is a data point ρ and a part realization of the part P_c^{n-1} (of the previous layer L_{n-1}) activated in this point. The following items describe all possible cases:

1. If there are no doublets of the layer L_n activated at the point ρ then there are no activations of candidate parts of the layer L_n in this point.

2. If there are m doublets activated at the point ρ, where $m \geq 1$, and all these doublets describe surface in the same directions from the point ρ then there are no candidate parts of the layer L_n at the point ρ.

3. If two doublets are activated in the data point ρ, for example, doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$, and these doublets describe surface in the opposite directions from the central subpart, then we say that there is an activation of a candidate part $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_k^{n-1}, \Delta_j^{sp}))$ residing at the point ρ.

4. If there are more than two doublets activated at the point ρ, for example, $m \geq 1$ doublets describing surface on one side from the point ρ, and $k \geq 1$ doublets describing the surface of the other side from the point ρ, where $max(k, m) > 1$, then we say there are activations of $m \times k$ candidate parts at the point ρ. Let us consider the example when $m = 2$ (doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$) and $k = 1$ (doublet $(P_c^{n-1}, (P_b^{n-1}, \Delta_a^{sp}))$). Then we have $m \times k = 2 \times 1 = 2$ activations of candidate parts at the point ρ, i.e. $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$.

Forming a Set of Candidate Parts

So far we identified which candidate parts reside in different points of the input data. Some data points contain no activations of candidate parts, while other data points may have one or more activations. To form the set $\mathcal{C}(L_n)$ of candidate parts of the layer L_n

we have to measure frequencies ν_i for each candidate part C_i^n, showing the number of occurrences of activations of this candidate part in the training data. Then we define a small threshold value t_r (usually $t_r = 1 + log(m)$, where m is a number of training models) and include all candidate parts with the frequency larger than this value to the set of candidate parts $\mathcal{C}(L_n)$. This thresholding is done to avoid including very rarely observed parts to the set of candidate parts. After the set of candidate parts is formed, we start the part selection procedure, that selects a vocabulary of the layer L_n from the set of candidate parts, i.e. if searches for $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$.

6.7.3 Part Selection (Grouping by OR-Nodes)

In Section 5.6 it was explained that part selection is the key stage of the learning procedure, since at this stage we form the vocabulary of the layer L_n by selecting some candidate parts $\mathcal{S}(L_n) \subseteq \mathcal{C}(L_n)$. It should be pointed that the part selection problem is substantially easier in the surface-based compositional hierarchy than in the view-based hierarchy, due to a much smaller (typically by at least two orders of magnitude) number of candidate parts[1]. The main reason for this is that parts in the surface-based hierarchy are orientation-independent, therefore, it is sufficient to represent each surface shape by a single part, while in the view-based hierarchy each surface shape should be represented by multiple orientation-dependent parts.

Since the number of candidate parts is much smaller, we can afford including almost all of them in the vocabulary, i.e. there is no need to look for a certain trade-off between importance of candidate parts and the cardinality of the vocabulary. The only procedure we perform on the part selection step is grouping of the candidate parts into clusters (termed OR-nodes), such that each OR-node represents a set of geometrically similar parts. Before making the OR-nodes we first measure the pairwise distances showing geometric similarity of candidate parts. We use the volumetric distance d_v described

[1]For instance, on the second layer of the view-based compositional hierarchy we have approximately $2.5 * 10^4$, while in the surface-based hierarchy this number is around 10^2.

in Section 5.6.1, i.e. $d_v(C_i^n, C_j^n)$ between candidate parts C_i^n and C_j^n approximates the volume between two surfaces, representing mean reconstructions of these parts, given that they are centered in the same point and their reference frames are aligned. After measuring pairwise distances between candidate parts we use Algorithm 12 for grouping parts into OR-nodes.

Algorithm 12: Algorithm for making OR-nodes

Data: Set of candidate parts $\mathcal{C}(L_n)$,
Pre-defined similarity threshold T_s,
Pre-defined frequency threshold T_ν.
Result: The vocabulary $\mathcal{S}(L_n)$.

$\mathcal{S}(L_n) = \emptyset$;
repeat
  Find the part $C_i^n \in \mathcal{C}(L_n)$ with the largest frequency ν;
  Find the set of parts $\Upsilon = \{C_j^n\}_j$ such that $d_v(C_i^n, C_j^n) < T_s$;
  Exclude all parts that belong to Υ from the set $\mathcal{C}(L_n)$;
  Include all parts from Υ in $\mathcal{S}(L_n)$;
  Make links from all parts from Υ to the part C_i^n;
until $max\left(\nu(\mathcal{C}(L_n))\right) \geq T_\nu$;

The presented algorithm not only includes parts in the vocabulary but also establish links between them. These links point from a group of geometrically similar parts to the part with the largest (within a group) frequency ν, which is considered as a *representative* part for this group. These links are used on the inference step.

6.8 Inference of a Single Layer

In this section the inference procedure for a single layer $L_n, (n \geqslant 2)$ is described. The goal of inference is to match the input data against the vocabulary in order to find realizations of vocabulary parts in the data. The inference algorithm proceeds in two steps. First, inference of doublets from the input data is done, exactly as was described in Algorithm 11. Second, when activations of doublets of the layer L_n in the training data are found, the co-activations of doublets each data point are analyzed and the inference of parts is performed based on this analysis. There are four possible scenarios:

1. If there are no doublets of the layer L_n activated at the point ρ then there are no part realizations of the layer L_n in this point.

2. If there are m doublets activated at the point ρ, where $m \geq 1$, and all these doublets describe surface in the same directions from the point ρ then there are no part realizations of the layer L_n at the point ρ.

3. If two doublets are activated in the data point ρ, for example, doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$, and these doublets describe surface in the opposite directions from the central subpart, then we should check if the part $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_k^{n-1}, \Delta_j^{sp}))$ exists in the vocabulary. It yes, then there is a realization of this part residing at the point ρ.

4. If there are more than two doublets activated in the point ρ, for example, $m \geq 1$ doublets describing surface on one side from ρ, and $k \geq 1$ doublets describing the surface on the other side from ρ, where $max(k, m) > 1$, then we form $m \times k$ parts as shown in Item 2. Let us consider the example when $m = 2$ (doublets $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}))$) and $k = 1$ (doublet $(P_c^{n-1}, (P_b^{n-1}, \Delta_a^{sp}))$). Then we have to form $m \times k = 2 \times 1$ parts from these doublets, i.e. $(P_c^{n-1}, (P_l^{n-1}, \Delta_i^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$ and $(P_c^{n-1}, (P_k^{n-1}, \Delta_j^{sp}), (P_b^{n-1}, \Delta_a^{sp}))$. Then for each of these parts we check if it belongs to the vocabulary, and, if yes, then it has a realization in the point ρ.

Notice that part realizations of the layers $L_n, (n > 2)$ inherit the reference frames from their central subparts. As for the part realizations of L_2, they receive the reference frame, computed from the curvature tensor of the receptive field as shown in Section 6.4.

After part realizations are inferred from the training data, they are replaced by the realizations of the representative parts of the corresponding OR-node. In Section we explained that there are links from each vocabulary part to the representative part of its OR-node. Then, after inference of each part realization, we follow these links and replace the realization of this part by the realization (having the same position and orientation) of the representative part of this OR-node. This is done to avoid overgrowing of the

vocabulary of the next layers, by making sure that a range of similar surfaces is represented by a single vocabulary part.

6.9 Pooling

The pooling procedure is defined in a very similar way as in the view-based compositional hierarchy. The only difference is that we perform pooling separately for each face $f_k \in F$ of the triangulated mesh. We first extract the regularly sampled points of this face $\mathcal{P}(f_k)$, then partition the face to non-overlapping regions approximatelly of size 3×3 points for the layer L_3 (9×9 points for L_5, etc.) and then perform pooling for each of this regions, as was described in Section . Notice that pooling is an optional procedure, and the presented system works without it, however, in this case the computations on higher layers become several times slower.

6.10 Evaluation

In this section we present the results of the experimental evaluation of the surface-based compositional hierarchical framework. One of the research goals of this thesis was to develop a system for multi-layer learning of surface shape features having both large discriminative power and the greater robustness to view changes and in-plane rotations than CNN-based methods. To evaluate the discriminative power we should use the settings in which training and testing objects are aligned, while to evaluate the robustness we should perform recognition from novel (unseen) views and in-plane rotations.

The remainder of this section is organized as follows: Subsection 6.10.1 presents the evaluation methodology, Subsection 6.10.2 presents the dataset used for evaluation, Subsection 6.10.3 describes the CNN-based methods against which we compare our results, while Subsection 6.10.4 presents the results of the experiments, which are compared to the results achieved by the CNN-based methods.

6.10.1 Evaluation Methodology

The widely used protocol for evaluating the discriminative power of computer vision methods is based on the Washington RGB-D dataset [48]. However, this protocol is not suitable for evaluating geometric invariance of computer vision systems due to several reasons. First, this protocol assumes that training objects are approximately in the same range of positions and orientations relative to the camera as testing objects. That means no recognition from radically novel viewpoints and/or in-plane rotations is involved. Second, the Washington dataset represents only rigid objects, mainly having relatively simple surface shapes, for example, banana, orange, can, tomato, etc. This dataset does not contain deformable shape models with articulated parts, for example, models of animals, human models, etc.

Because of these limitations, in this thesis we propose a novel protocol for evaluating both the discriminative power and the geometrical invariance of computer vision methods. There are two key principles which are involved in this protocol. First, the dataset used for learning and inference should be mainly formed of complicated shape models, i.e. **deformable shapes** with articulated parts. Second, the number of objects per category should be rather small, such that only some deformations of non-rigid objects are represented in the training set. In this scenario, those methods which are better in recognition of unseen deformations of objects (e.g. deformations of a standing person to a sitting one, etc.) should be in a more favorable position. Third, training and testing objects should not be aligned, i.e. category recognition is should be done from **novel viewpoints and in-plane rotations**. In general, this protocol is much more difficult than the protocol based on the Washington [48], and only the methods demonstrating both the large discriminative power and the robustness to different geometric transformations can achieve high performance in these settings.

In our experiments the shape models of the input dataset are first approximately aligned both within categories and across categories. For example, in the category of frying pans all handles should be pointing in the same direction (alignment within categories),

and the handles of objects of other categories (e.g. mugs) should be pointing in the same direction as well (alignment across categories). In general, object alignment is not an easy task, especially in the case of deformable objects (e.g. category of snakes), and it is very hard (or impossible) to define the rules for aligning them. We align the dataset manually, however, it should be stressed, that the alignment of deformable objects can sometimes be done only very roughly.

After the alignment is done, depth images representing front views[1] of each object are produced. After that, we perform two types of experiments. In the first experiment we perform leave-one-out cross-validation to partition the data into the training and testing sets. In this experiment use the front view images both for learning and recognition, i.e. in this case learning and recognition are done from the same view (though from different objects).

In the second experiment we use 5-fold cross-validation to partition the input data to the training and testing sets. We use the front view images for training, while the recognition is done from novel (unseen) views, which are obtained by changing the camera's azimuth and elevations by 15°, 30°, 45° and 60° relative to the front view, and from novel in-plane rotations obtained by rotating the camera around its optical axis by 15°, 30°, 45° and 60°. Methods demonstrating better results in recognition from novel views and in-plane rotations are considered to be more robust to these geometric transformations.

6.10.2 DataSet

As was mentioned in Chapter 3, according to the best of our knowledge, there are no large datasets of deformable objects. The largest dataset, we are aware of, is SHREC'15 dataset [141] comprising 1200 models split into 60 categories (20 models per category). Other datasets are substantially smaller, for example, SHREC'11[236, 181] comprises 600 models split into 30 categories, while the Watertight dataset [237] 400 objects split

[1]By the front view we mean the view which makes the largest portion of objects' surface visible

into 20 categories[1]. There exist other datasets of non-rigid objects, such as the McGill benchmark [238] and other datasets, however, these datasets are very small, typically comprising few categories and a small number of models per category. Note, that SHREC'11 and SHREC'15 have many repetitions, and actually, SHREC'15 can be considered as an extension of SHREC'11.

We analyzed the existing datasets and composed a single dataset using models from SHREC'15, SHREC'11 and the Watertight dataset. We removed the repeating categories and the categories which we found non-interesting for shape analysis, for instance, a category of paper sheets. We also included 15 categories of rigid objects from the PaCMan dataset of rigid objects (20 objects per category). Totally we obtained the dataset comprising 72 object categories (20 objects per category, 1440 models altogether), including 52 categories of deformable objects (mainly models of animals, insects, etc.), and 20 categories of rigid objects. Figure 6.9 shows 8 examples of deformable object categories. The dataset is available at [2]

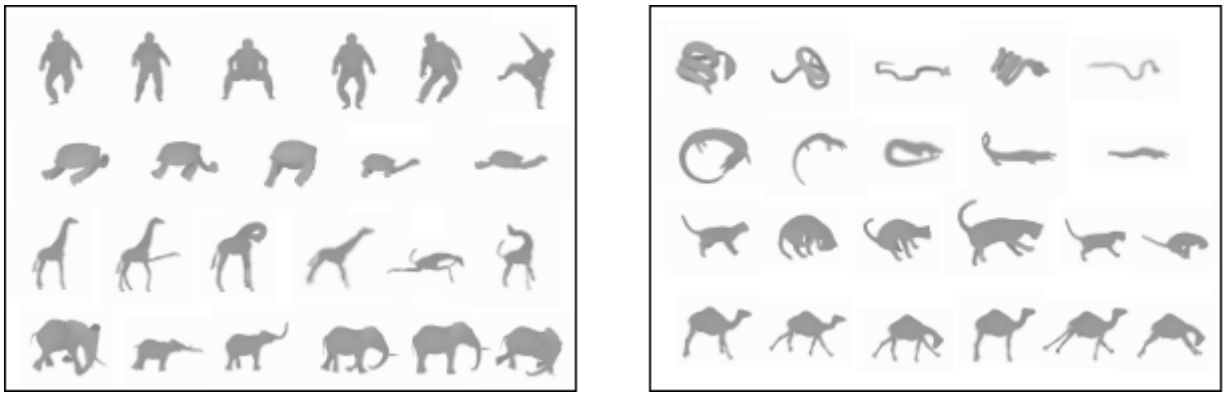


Figure 6.9: Examples of deformable object categories.

6.10.3 Methods for Comparison

We compare our method against CNN-RNN [24] (implementation is downloaded from the author's website) and the CaffeNet [49] open-source version of AlexNet [17]. We use

[1]Though some categories in this dataset represent rigid objects
[2]https://drive.google.com/open?id=0BzGi8qyZ5m6VWUZvNmdOQVlhd00

the CNN-RNN of Socher *et al.* [24] as a baseline method to study the basic properties of CNNs, i.e. how well it learns from a small amount of data (without pre-training on large-scale datasets), and how invariant it is to different geometric transformations. It may be worth mentioning that this architecture has two different pipelines for the RGB and for the depth channel, that is why for our experiments we switch off the RGB pipeline and evaluate only the discriminative power and the invariance of features learned from the depth channel.

As for AlexNet, we chose it in order to compare our method against the state-of the-art architecture. It has been shown that AlexNet (pre-trained on ImageNet [15]) has a very rich set of image features, which are proved to be useful not only for recognition on ImageNet but also for other tasks, including category recognition from depth images. For instance, Schwarz *et al.* [102] extracted the features of this CNN, applied them to categorization of the Washington RGB-D dataset [48] and demonstrated state-of-the-art results.

We decided to follow a similar way, extracting features of different layers from AlexNet and applying them to recognition on our dataset (using linear SVM for classification, similarly to [102]). We evaluated features from the last convolutional layer (conv5), the last pooling layer (pool5), the first fully connected layer (fc6), and the second fully connected layer (fc7). We also tested features from other layers, but they demonstrated substantially worse performance, that is why they are not presented in this thesis.

It has been shown in the literature [13, 102, 26], that colorization of depth images significantly improves the results of CNNs. We used the colorization by surface normals (SN colorization), which was shown to achieve the best result in similar settings [26]. In addition to that, we propose our own way of colorization, in which two principal curvatures and the mean curvature (PC colorization) are used to form a 3-channel RGB image from the depth image[1]. Figure 6.10 shows examples of a depth image colorized using surface

[1]According to the best of our knowledge, nobody used this technique for colorizing depth channels, though Sinha *et al.* [21] used principal curvatures to convert parameterized 3D models to geometry images.

normals and principal curvatures.

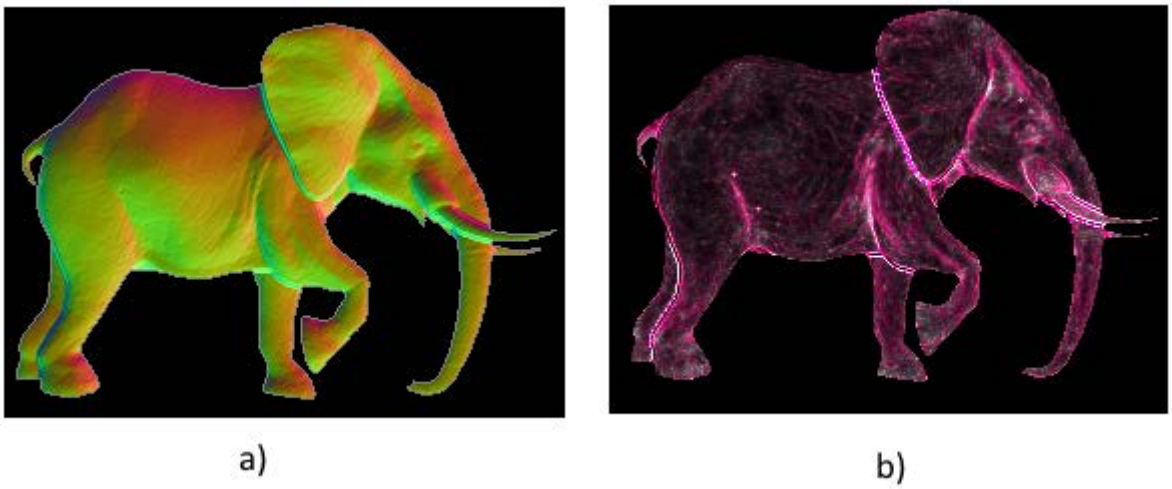


Figure 6.10: Depth image colorized by a) surface normals, b) Principal curvatures

6.10.4 Evaluation Results

In this section the results of experimental evaluations of the surface-based compositional hierarchy are presented.

Category Recognition from the Front View

For this experiment we used leave-one-out cross-validation to split the models into training and testing sets. We used the same category descriptor (i.e. HoCP, described in Chapter 5), and the SVM with χ^2 kernels for classification. Note, however, that in one of the experiments we use another type of HoCP, i.e. a radial HoCP, which partition the domain as shown in Figure 6.11. Although, the radial HoCP is less discriminative, it leads to the full rotational invariance.

As for our approach, we used several different settings. First, we used only L_1 features, which are quantized into 5×5, 7×7, and 9×9 orientational bins (using the Algorithm 4 from Section 5.3.2). Then, we subsequently included features from the next layers (though without quantization of orientations), and demonstrated that the best results are achieved with 9×9 quantization of the first layer features, and features from layers L_2 - L_4. Note,

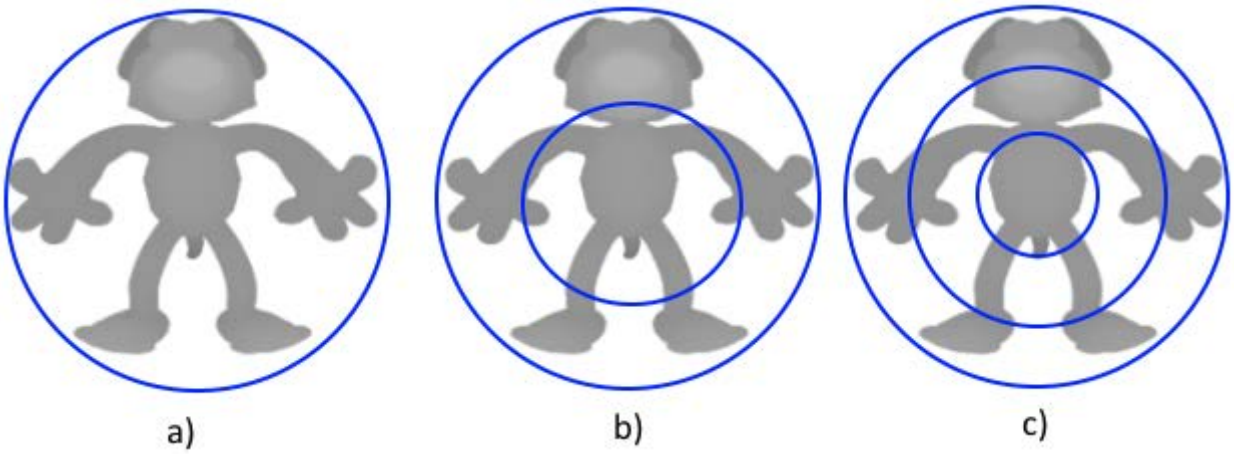


Figure 6.11: Illustration for the radial HoCP. a) The first section of HoCP covers the whole shape, b) Next two sections of HoCP, c) Next three sections of HoPC

that addition of the next layer L_5 slightly decreases the recognition accuracy. Table 6.1 shows the results achieved by different methods on our dataset.

Table 6.1: Results achieved by different methods on the new dataset. Leave-one-out cross validation is used.

Method	Result
CNN-RNN (raw depth data) [24]	75.6 ± 4.8
AlexNet(conv5) raw depth data	82.0 ± 2.6
AlexNet(pool5) raw depth data	**87.1** ± 3.5
AlexNet (fc6) raw depth data	84.2 ± 3.8
AlexNet(pool5+fc6) raw depth data	86.7 ± 3.8
AlexNet(fc7) raw depth data	79.2 ± 5.0
AlexNet(pool5) PC colorization	90.4 ± 3.8
AlexNet(fc6) PC colorization	91.0 ± 3.2
AlexNet(fc6+pool5) PC colorization	**91.1** ± 3.7
AlexNet(fc7) PC colorization	89.0 ± 3.7
AlexNet(fc6) SN colorization [205]	92.05 ± 2.6
AlexNet(pool5) SN colorization	91.9 ± 2.8
AlexNet(fc7) SN colorization	91.9 ± 2.4

Continuation of Table 6.1	
Method	Result
AlexNet(pool5+fc6) SN colorization	**92.4** ± 2.6
View-based system, layers 1-5, Quadratic-chi kernels, 9×9 bins	86.6 ± 4.3
Our approach (L_1 features, 5×5 orientations)	81.4 ± 4.2
Our approach (L_1 features, 7×7 orientations)	83.4 ± 4.1
Our approach (L_1 features, 9×9 orientations)	84.2 ± 3.6
Our approach (L_1 features, 9×9 orientations, + L_2)	89.0 ± 2.6
Our approach (L_1 features, 9×9 orientations, + L_2 + L_3)	90.1 ± 2.5
Our approach (L_1 features, 9×9 orientations, + L_2 + L_3 + L_4)	**91.4 ± 3.3**
Our approach (L_1 features, 7×7 orientations, + L_2 + L_3 + L_4)	91.1 ± 3.7
Our approach (L_1 features, 9×9 orientations, + L_2 + L_3 + L_4 + L_5)	90.4 ± 4.0

Several conclusions can be made from this table. First, we reconfirmed the results of previous works, stating that colorization of the depth channel substantially improved the results of the CNN-based architecture. As expected, the SN colorization led to the best object categorization accuracy. We also demonstrated the usefulness of our method for colorization of depth images using the principal curvatures and the mean curvature, which also led to good results.

As for our method, it has shown good results, which are almost at the level of the advanced CNN-based architecture. Note, that our method significantly outperformed both CNNs working with raw depth images, and was only beaten (by a small 1% margin) by the CNN classifying colorized depth images. Also note, that our system was trained only on a small dataset, while AlexNet was pre-trained on the large-scale dataset, and therefore contains a very rich set of features. That is why we can consider the achieved results as a confirmation of the large discriminative power of our method.

Additionally, we provide the result of evaluation of the view-based system (described in Chapter 5) on the same dataset. The following settings were used: 9×9 orientational

bins at the first layer, layers L_1 - L_5 were used, MDL + entropy-based part selection, empty cells are switched on, pooling is switched on. Figure 6.12 illustrates the beeswarm plot showing how the accuracy fluctuates across 20 iterations.

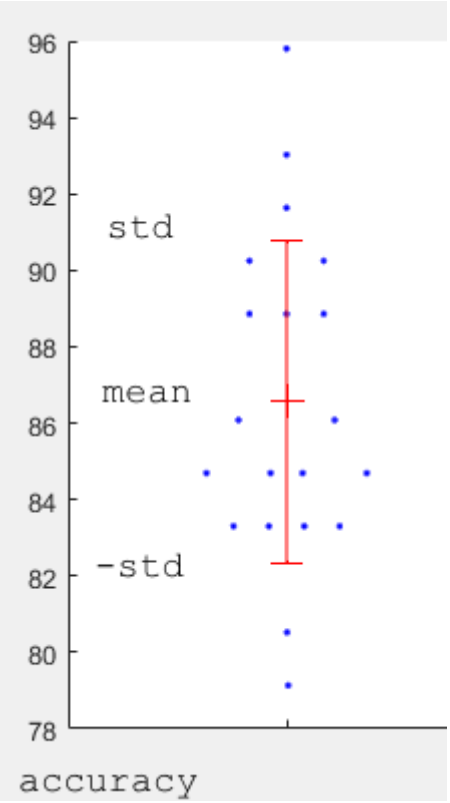


Figure 6.12: Beeswarm plot illustrating the accuracy of the view-based system on the new dataset.

Category Recognition from Novel Views and in-Plane Rotations

In this experiment we evaluate geometric invariance of different methods, using the methodology described in Subsection 6.10.1, in which we use the front views of training objects for learning (both vocabulary learning, and category learning), while testing is done from the novel views and in-plane rotations which are obtained by changing the camera's azimuth and elevation, and by rotating the camera about its optical axis. In contrast to the previous experiments, where leave-one-out cross-validation was used, in this experiment we use 5-fold cross validation, mainly because it is less computationally expensive.

We used four different settings of our method. First, we used 9×9 quantization of orientations of L_1 parts. However, though this quantization proved to be useful for recognition from the front view, it makes the performance of the system worse, when performing recognition from novel views and in-plane rotations. Let us call these settings of the system the **front view mode**. Additionally we used another mode (termed **invariant mode**), where no quantization of orientations is done. Unfortunately, this option decreases the recognition accuracy from the front view (approximately by 3%), but it substantially increases the robustness to different transformations. In the third mode of our system, termed **invariant mode with parameter tuning**, we improved the invariant mode, in particular, we performed a grid search of the parameter regulating the size of the OR-nodes to maximize the recognition accuracy on this dataset. Additionally we switched off pooling after L_3, substantially increasing the number of part realizations on a surface. The fourth mode involves uses **radial HoCP**, making our system fully rotationally invariant.

Results of the comparison of our method (in different modes) with CNN-RNN and AlexNet (in different settings) for out-plane rotations are shown in Figure 6.13. In Figure 6.14 we re-scaled the results by assuming that the performance of each method from the front view is equal to 1. This plot demonstrates how rapidly the performance of each method goes down for different view changes, relative to the performance of this method from the front view. Note that both these plot show average values taken by changing of camera azimuth and elevation.

Similarly structured results for different in-plane rotations are shown in Figure 6.15, while the re-scaled results for the in-plane rotations are shown in Figure 6.16.

First, our methods demonstrated the greater robustness to all transformations than both CNNs working with raw depth images. Second, our method (in any of its modes) achieves much greater robustness to in-plane rotation than both CNN. In one of the modes (with radial HoCPs) our method becomes fully rotationally-invariant. Third, our method (in the invariant mode with parameter tuning) achieves greater robustness (in relative

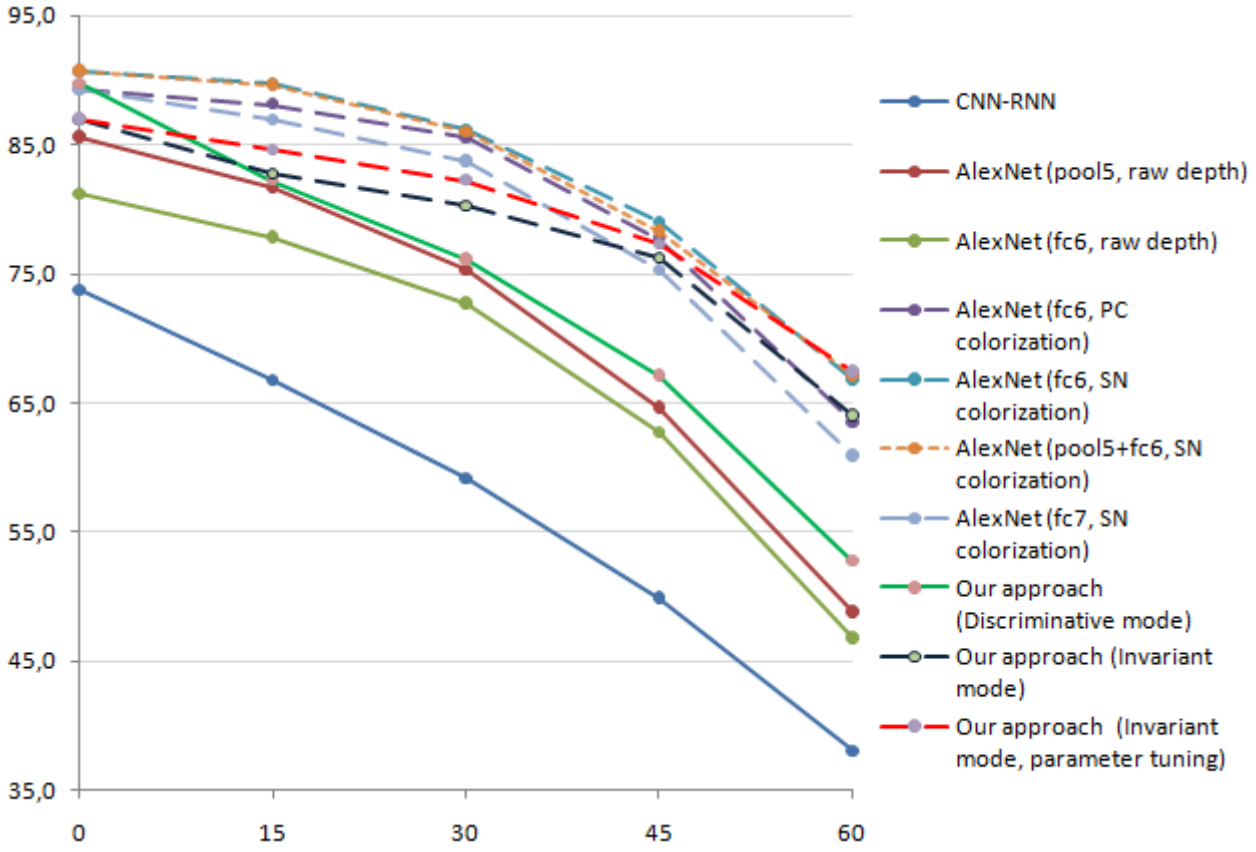


Figure 6.13: Performance of different methods under changes of view

values) to large changes of view (45-60 degrees) than AlexNet, though the margin is not large. However, for small changes of view (15-30 degrees) AlexNet performs better than our methods both in relative and absolute values.

6.11 Conclusion

This chapter presents the surface-based compositional hierarchical framework that is based on the principles of hierarchical compositionality and of the principle of intrinsic part-based reference frames. We described the design principles of this framework and all ingredients of the system, namely the local reference frames, the composition rules, OR-nodes, etc. We also provided the detailed description of each step of the learning and inference algorithms, including pre-processing of the input data, inference of the first layer, collecting of the co-occurrence statistics, clustering of statistical maps and forming

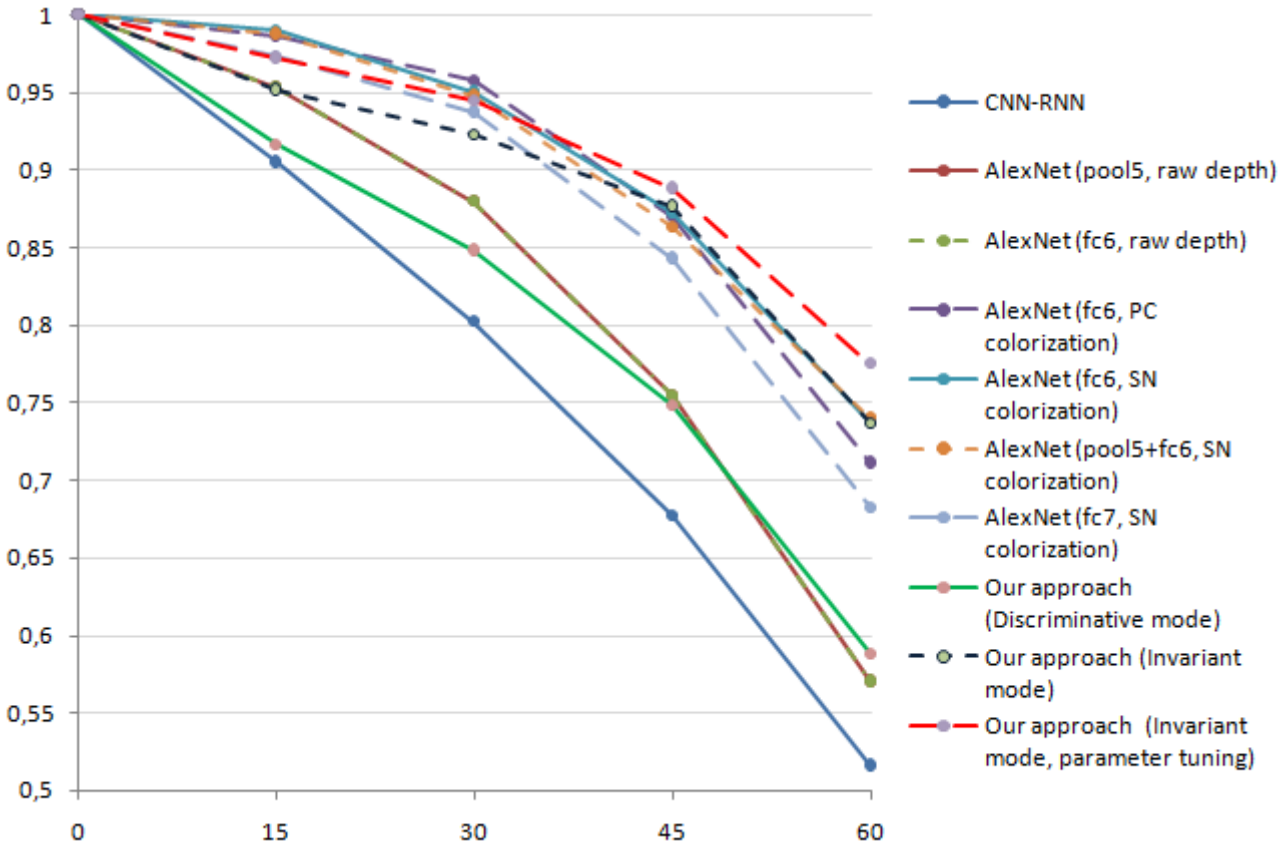


Figure 6.14: Relative performance of different methods under changes of view. Recognition performance of each method from the front view is assumed to be 1.0

doublets, inference of doublets from the training data, making OR-Nodes, and pooling.

The the most difficult part of this research work is related to orientations of part realizations and the local reference frames. We specify how LRFs should be computed, how to handle the situations where it is not possible to compute them, how to describe spatial relations of LRFs, and how they are inherited by part realizations of higher layers. Another difficult problem is the clustering algorithm for statistical maps since these maps contain a very rich information, and clusters are not always well-separated, especially on the lower layers.

The work presented in this chapter addressed the second research goal of this thesis. We complemented the principles of hierarchical compositionality by the principle of part-based reference frames and developed a novel system on basis of these principles. Results of the experimental evaluation demonstrated that the goals we mainly achieved. In

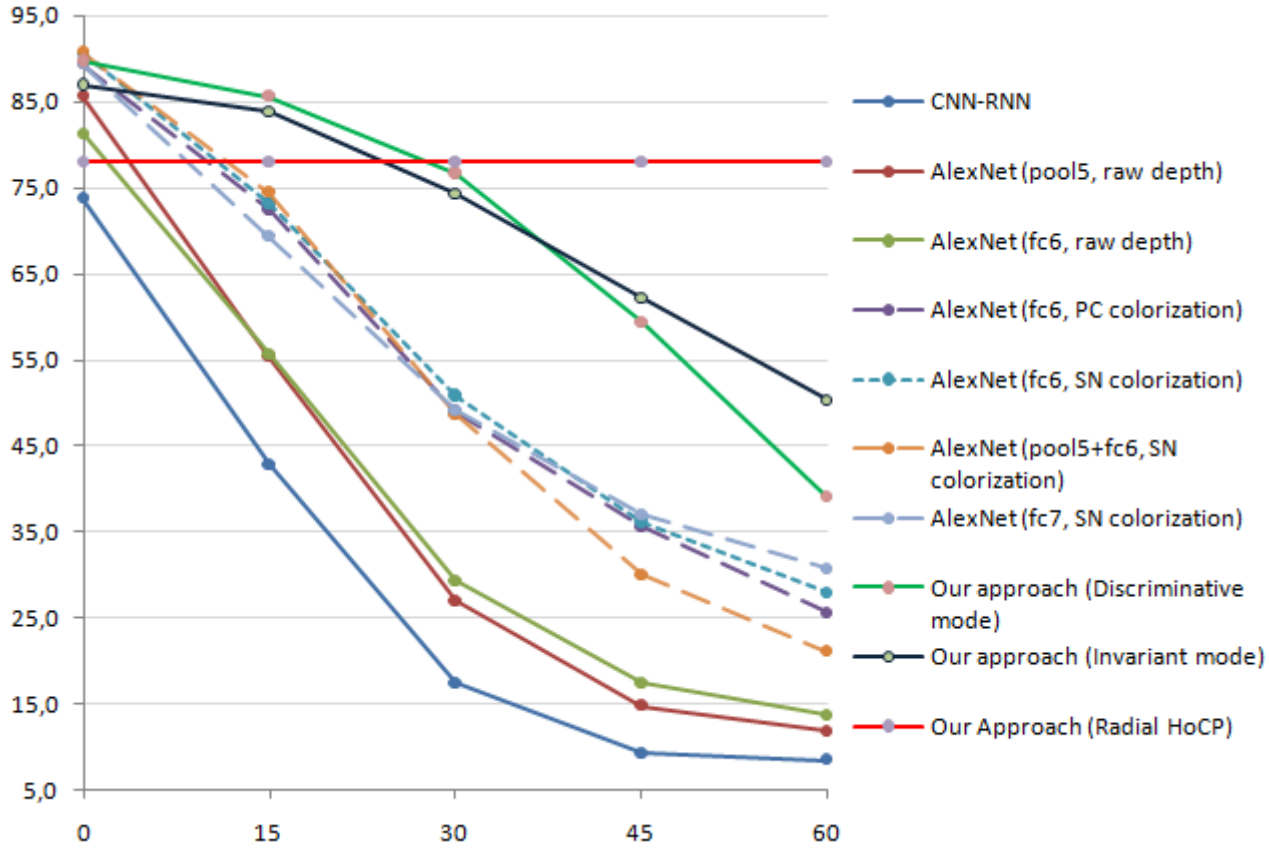


Figure 6.15: Performance of different methods under in-plane rotations

recognition from the front view, our framework demonstrated the accuracy nearly at the level of the state-of-the-art CNN pre-trained on ImageNet. Our framework outperforms the competing CNNs in recognition from unseen in-plane rotations by a large margin. As for the changes of view, the achieved results were not very conclusive. Our system outperformed both CNN working with raw depth data. It also demonstrated a greater robustness (in relative values) than CNNs for the large view changes (45-60 degrees). However, for small view changes AlexNet classifying colorized depth images shows the greater robustness than our method.

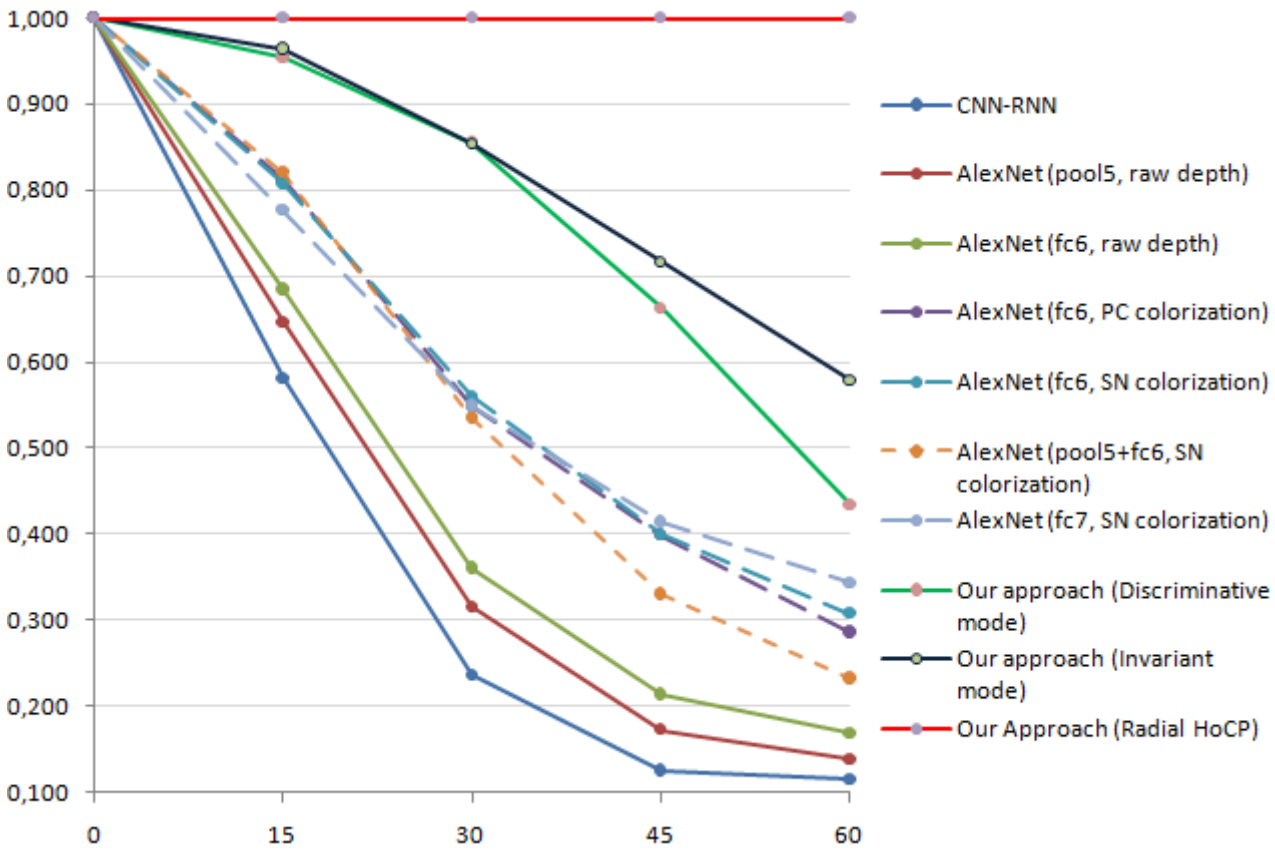


Figure 6.16: Relative performance of different methods under in-plane rotations. Recognition result of each method without in-plane rotation is assumed to be 1.0

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this thesis we addressed the complicated problem of object categorization based on surface shape features. We identified that one of the main problems of deep learning methods applied to 3D shape analysis is the lack of robustness to different geometric transformations. We proposed two novel frameworks for multi-layer learning of surface shape features based on the principles of hierarchical compositionality. The main purpose of the first framework was to transfer the principles of hierarchical compositionality to the domain of surface shape analysis. The second framework was purposed to enable learning of surface shape features having a large discriminative power and a greater robustness to rigid body transformations than the features learned by state-of-the-art CNNs.

7.1 Conclusions

The following conclusions can be made from this work:

- We transferred the principles of hierarchical compositionality to the domain of surface shape analysis by developing a novel framework for multi-layer learning of surface shape features. This thesis also demonstrates that this framework enables efficient category recognition from the large-scale Washington RGB-D dataset.
- We demonstrated that the usage of the Quadratic-chi affinity measure within SVM can significantly improve results of classification of HoCPs.

- We demonstrated that the principles of hierarchical compositionality, complemented by the principle of intrinsic part-based reference frames, enable development of a multi-layer framework capable of learning surface shape features of large discriminative power and a greater robustness to in-plane rotations compared to features learned by the state-of-the-art CNN-based architecture (AlexNet).

- We demonstrated that this framework has a greater robustness to changes of view, than CNN-based architectures working with raw depth images. However, our experiments show that the colorization of depth images substantially increases the robustness of AlexNet to view changes. That is why, our framework demonstrated a greater robustness than AlexNet only for large changes of view, and only according to one of the measures (i.e. the ratio of the recognition accuracy from the unseen view and the recognition accuracy from the front view).

7.2 Summary

Chapter 1 is the introduction to this thesis. We presented the background, the motivation, the main research goals of the thesis, as well as the hypotheses and contributions of this thesis.

Chapter 2 is dedicated to the representation of 2D images. We first describe the domain of handcrafted image features and the methods exploiting them and have demonstrated that the handcrafted features having large discriminative power and robust to different transformations have been very popularly employed. Then, we introduce the concept of deep learning and described the main types of ANNs. Finally, we described the theoretical work on hierarchical compositionality and made an overview of the existing compositional hierarchical methods. We pointed that compositional hierarchical methods have not been compared with modern CNN-based method in the mainstream challenges for object categorization, however, these systems have some interesting theoretical properties.

Chapter 3 is dedicated to surface shape analysis. It outlines the handcrafted shape features and demonstrates that some of these features are invariant to different geometric transformations, including rigid body transformations and isometric deformations of shapes. Then, this chapter describes recent deep learning methods used for shape analysis and outlines their limitations. The main conclusion from this chapter is that the application of deep learning methods to surface shape analysis problems is burdened by the limitations of the existing methods, e.g. lack of invariance to different geometric transformations. On the other hand, we emphasize that the situation is changing very rapidly since the number of deep learning methods specially designed for the 3D domain grows rapidly.

Chapter 4 provides a high-level description of the learning and inference algorithms used in the view-based and the surface-based compositional hierarchies.

Chapter 5 describes the view-based compositional hierarchy providing the detailed description of the learning and inference algorithms. The key stage of the learning algorithms is the part selection. The chapter proposes several importance measures and evaluates different approaches to part selection, i.e. minimization of different multi-objective cost functions and the divide-and-conquer approaches which assume that different sub-algorithms should perform part selection based on different criteria, and then the results should be merged. Our experiments show that the divide-and-conquer approaches lead to substantially better results in object categorization. This chapter also describes a novel way of classifying the category descriptors (HoCP), using EMD and Quadratic-chi kernels for SVM. This chapter confirms the first hypothesis of this thesis.

Chapter 6 is dedicated to the surface-based framework. It introduces the principle of intrinsic hierarchical reference frames and describes all steps of the learning and inference procedures. There are two most difficult theoretical questions described in this chapter. The first question relates to local reference frames. We specify how to compute LRFs, how to describe spatial relations between them, and how they are inherited by part realizations of higher layers. The second question is related to clustering algorithm for

statistical maps, which is a difficult problem since statistical maps contain very rich information about relative positions and relative orientations. Additionally, this chapter proposes a novel protocol for evaluating the robustness of computer vision methods to geometric transformations. The results of experimental evaluation of the surface-based hierarchy show that it has a large discriminative power and a greater robustness to in-plane rotations than CNN-based methods. As for the view changes, the hierarchy can be considered (according to the relative performance) to be more robust for large view changes (around 45-60 degrees), while the CNN-based method is more robust to small view-changes.

7.3 Future Work

From our point of view, the proposed surface-based compositional hierarchy has a large potential to be further developed and applied to various computer vision and shape analysis tasks. So far we demonstrated the usefulness of this framework for object category recognition from range images, representing partial views of objects. In the experimental settings described in this thesis we perform learning of surface shape features from a set of range images and then use these learned features for category recognition from the same type of input data.

One of the promising directions of the future work is to extend our system to enable vocabulary learning from full 3D shape models. Such an extension would require introducing another set of composition rules for higher layers of the hierarchy, and change local reference frames from the surface-based to volumetric ones, for instance, computed using the principal component analysis (PCA). Our assumption is that features learned from a dataset of full shape models may be successfully applied to the 3D shape retrieval task.

An interesting direction of the future work is to conduct the research on object and category descriptors exploiting the proposed compositional hierarchical shape vocabulary. Actually, in this thesis we exploit only one type of category descriptors, namely the

Histogram of Compositional Parts (HoCP). It may be, however, a very promising direction of work to study other types of category descriptors and to compare their performance against the performance of HoCPs. It is also a scientifically interesting question whether or not the features learned by our framework may be used within the existing global object and category descriptors (e.g. the Covariance descriptor [198] or other global descriptors presented in Chapter 6).

Another possible direction of the future work is to apply this framework to object detection in cluttered scenes. Since objects in cluttered scenes may have arbitrary positions and orientations relative to a camera, the robustness of an object detection system to in-plane and out-plane rotations becomes one of the most important properties. In such settings the surface-based compositional hierarchy, proposed in this thesis, may be beneficial. It is also important to conduct the research on category recognition under occlusions, and to figure out whether the proposed compositional hierarchical system is robust in object/category recognition under occlusions.

Additionally learning of 3D shape models representing closed surfaces may open the opportunities for designing a system for making predictions of the hidden (e.g. occluded or self-occluded) parts of the objects. This property may be interesting for robotics scenarios, for example for robot grasping, where the hidden (occluded) part of the object should be reconstructed for grasp planning.

LIST OF REFERENCES

[1] William T. Freeman. The generic viewpoint assumption in a framework for visual perception. *Nature*, 368(6471):542–545, 1994.

[2] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[3] Simon Thorpe, Denis Fize, Catherine Marlot, et al. Speed of processing in the human visual system. *nature*, 381(6582):520–522, 1996.

[4] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[5] Alberto Garcia-Garcia, Francisco Gomez-Donoso, Jose Garcia-Rodriguez, S Orts-Escolano, M Cazorla, and J Azorin-Lopez. Pointnet: A 3D convolutional neural network for real-time object class recognition. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1578–1584. IEEE, 2016.

[6] Ziyan Wang, Ruogu Lin, Jiwen Lu, Jianjiang Feng, et al. Correlated and individual multi-modal deep learning for RGB-D object recognition. *arXiv preprint arXiv:1604.01655*, 2016.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[9] Andrew E Johnson. *Spin-images: a representation for 3-D surface matching*. PhD thesis, Microsoft Research, 1997.

[10] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[11] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

[12] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[16] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[19] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. Technical report, 2015.

[20] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

[21] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3D shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[22] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.

[24] Richard Socher and Brody Huval and Bharath Bhat and Christopher D. Manning and Andrew Y. Ng. Convolutional-Recursive Deep Learning for 3D Object Classification. In *Advances in Neural Information Processing Systems 25*. 2012.

[25] Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3D shape retrieval. *Neurocomputing*, 204:41–50, 2016.

[26] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

[27] Davide Boscaini, Jonathan Masci, Simone Melzi, Michael M Bronstein, Umberto Castellani, and Pierre Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, pages 13–23. Wiley Online Library, 2015.

[28] Stuart Geman, Daniel F Potter, and Zhiyi Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002.

[29] Shimon Ullman and Boris Epshtein. Visual classification by a hierarchy of extended fragments. In *Toward Category-Level Object Recognition*, pages 321–344. Springer, 2006.

[30] B. Ommer and J. M. Buhmann. Learning the compositional nature of visual objects. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

[31] Sanja Fidler and Ales Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

[32] Zijian Xu, Hong Chen, Song-Chun Zhu, and Jiebo Luo. A hierarchical compositional model for face representation and sketching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):955–969, 2008.

[33] Long Leo Zhu, Chenxi Lin, Haoda Huang, Yuanhao Chen, and Alan Yuille. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *European Conference on Computer Vision*, pages 759–773. Springer, 2008.

[34] Renaud Detry, Nicolas Pugeault, and Justus H Piater. A probabilistic framework for 3D visual object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1790–1803, 2009.

[35] Zhangzhang Si and Song-Chun Zhu. Learning And-Or templates for object recognition and detection. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2189–2205, 2013.

[36] Umit Rusen Aktas, Mete Ozay, Aleš Leonardis, and Jeremy L Wyatt. A graph theoretic approach for object shape representation in compositional hierarchies using a hybrid generative-descriptive model. In *Computer Vision–ECCV 2014*, pages 566–581. Springer, 2014.

[37] Jifeng Dai, Yi Hong, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Unsupervised learning of dictionaries of hierarchical compositional models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2505–2512, 2014.

[38] Mete Ozay, Krzysztof Walas, and Ales Leonardis. A hierarchical approach for joint multi-view object pose estimation and categorization. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5480–5487. IEEE, 2014.

[39] Mete Ozay, Umit Rusen Aktas, Jeremy L Wyatt, and Ales Leonardis. Compositional hierarchical representation of shape manifolds for classification of non-manifold shapes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1662–1670, 2015.

[40] Domen Tabernik, Aleš Leonardis, Marko Boben, Danijel Skočaj, and Matej Kristan. Adding discriminative power to a generative hierarchical compositional model using histograms of compositions. *Computer Vision and Image Understanding*, 138:102–113, 2015.

[41] Alan Yuille and Roozbeh Mottaghi. Complexity of representation and inference in compositional models with part sharing. *Journal of Machine Learning Research*, 17(11):1–28, 2016.

[42] Sanja Fidler, Marko Boben, and Ales Leonardis. Learning a hierarchical compositional shape vocabulary for multi-class object representation. *arXiv preprint arXiv:1408.5516*, 2014.

[43] Wenze Hu and Song-Chun Zhu. Learning 3D object templates by quantizing geometry and appearance spaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(6):1190–1205, 2015.

[44] Daniel Frederic Potter. *Compositional Pattern Recognition*. PhD thesis, Brown University, 1999.

[45] Shih-Hsiu Huang. *Compositional approach to recognition using multi-scale computations*. PhD thesis, Brown University, 2001.

[46] Geoffrey Hinton. Some demonstrations of the effects of structural descriptions in mental imagery. *Cognitive Science*, 3(3):231–250, 1979.

[47] Paul J Besl. *Surfaces in range image understanding*. Springer-Verlag New York, Inc., 1988.

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

[49] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[50] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. TORCH7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[51] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on GPUs with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3. Citeseer, 2011.

[52] Yanhua Cheng, Rui Cai, Xin Zhao, and Kaiqi Huang. Convolutional fisher kernels for RGB-D object recognition. In *3D Vision (3DV), 2015 International Conference on*, pages 135–143. IEEE, 2015.

[53] Paul J Besl and Ramesh C Jain. Invariant surface characteristics for 3D object recognition in range images. *Computer vision, graphics, and image processing*, 33(1):33–80, 1986.

[54] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[55] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[56] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004.

[57] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages II: 2169–2178, 2006.

[58] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[59] Alexander Thomas, Vittorio Ferrar, Bastian Leibe, Tinne Tuytelaars, Bernt Schiel, and Luc Van Gool. Towards multi-view object class detection. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1589–1596. IEEE, 2006.

[60] Christoffer Valgren and Achim J Lilienthal. Sift, surf and seasons: Long-term outdoor localization using local features. In *EMCR*, 2007.

[61] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[62] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Computer vision and image understanding*, 113(3):345–352, 2009.

[63] Zheng Yi, Cao Zhiguo, and Xiao Yang. Multi-spectral remote image registration based on sift. *Electronics Letters*, 44(2):1, 2008.

[64] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[65] Luo Juan and Oubong Gwun. A comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[66] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, pages 490–503. Springer, 2006.

[67] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 494–501. ACM, 2007.

[68] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[69] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition,*

2003. Proceedings. 2003 IEEE Computer Society Conference on, volume 2, pages II–II. IEEE, 2003.

[70] Robert Fergus, Pietro Perona, and Andrew Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71(3):273–303, 2007.

[71] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[72] Otávio AB Penatti, Eduardo Valle, and Ricardo da S Torres. Comparative study of global color and texture descriptors for web image retrieval. *Journal of Visual Communication and Image Representation*, 23(2):359–380, 2012.

[73] Alireza Khotanzad and Yaw Hua Hong. Invariant image recognition by Zernike moments. *IEEE Transactions on pattern analysis and machine intelligence*, 12(5):489–497, 1990.

[74] Piotr Bilinski, François Bremond, and Mohamed Becha Kaaniche. Multiple object tracking with occlusions using hog descriptors and multi resolution images. In *Crime Detection and Prevention (ICDP 2009), 3rd International Conference on*, pages 1–6. IET, 2009.

[75] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An HOG-LBP human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39. IEEE, 2009.

[76] Mihran Tuceryan, Anil K Jain, et al. Texture analysis. *Handbook of pattern recognition and computer vision*, 2:207–248, 1993.

[77] Manish H Bharati, J Jay Liu, and John F MacGregor. Image texture analysis: methods and comparisons. *Chemometrics and intelligent laboratory systems*, 72(1):57–71, 2004.

[78] H-J Bunge. *Texture analysis in materials science: mathematical methods*. Elsevier, 2013.

[79] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[80] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[81] Guoying Zhao and Matti Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):915–928, 2007.

[82] Junge Zhang, Kaiqi Huang, Yinan Yu, and Tieniu Tan. Boosted local structured HOG-LBP for object localization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1393–1400. IEEE, 2011.

[83] Dimitri A Lisin, Marwan A Mattar, Matthew B Blaschko, Erik G Learned-Miller, and Mark C Benfield. Combining local and global image features for object class recognition. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 47–47. IEEE, 2005.

[84] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[85] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[86] Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1(1):1–47, 1991.

[87] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

[88] Edmund T. Rolls and Gustavo Deco. *Computational neuroscience of vision*. Oxford University Press, Oxford, 2002.

[89] Tai Sing Lee and David Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.

[90] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio J Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1847–1871, 2013.

[91] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[92] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[93] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[94] Quoc V Le, MarcAurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S Corrado, Jeff Dean, and Andrew Y Ng. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2012.

[95] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

[96] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

[97] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[98] Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Progress in Brain Research*, 165:535–547, 2007.

[99] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[100] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[101] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[102] Max Schwarz, Hannes Schulz, and Sven Behnke. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE, 2015.

[103] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[104] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DECAF: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[105] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, pages II: 1331–1338, 2005.

[106] Ya Jin and Stuart Geman. Context and hierarchy in a probabilistic image model. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2145–2152, 2006.

[107] Sanja Fidler, Marko Boben, and Ales Leonardis. Evaluating multi-class learning strategies in a generative hierarchical framework for object detection. In *Advances in Neural Information Processing Systems*, pages 531–539, 2009.

[108] Bjorn Ommer and Joachim Buhmann. Learning the compositional nature of visual object categories for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):501–516, 2010.

[109] Iasonas Kokkinos and Alan Yuille. Inference and learning with hierarchical shape models. *International Journal of Computer Vision*, 93(2):201–225, 2011.

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

[111] Ge Guo, Yizhou Wang, Tingting Jiang, Alan L Yuille, Fang Fang, and Wen Gao. A shape reconstructability measure of object part importance with applications to object detection and localization. *International Journal of Computer Vision*, 108(3):241–258, 2014.

[112] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984.

[113] Hong Chen, Zi Jian Xu, Zi Qiang Liu, and Song Chun Zhu. Composite templates for cloth modeling and sketching. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 943–950. IEEE, 2006.

[114] Yuanhao Chen, Long Zhu, Chenxi Lin, Hongjiang Zhang, and Alan L Yuille. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In *Advances in Neural Information Processing Systems*, pages 289–296, 2007.

[115] Song-Chun Zhu, David Mumford, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.

[116] Liang Lin, Shaowu Peng, Jake Porway, Song-Chun Zhu, and Yongtian Wang. An empirical study of object category recognition: Sequential testing with generalized samples. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[117] Liang Lin, Tianfu Wu, Jake Porway, and Zijian Xu. A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*, 42(7):1297–1307, 2009.

[118] Yang Lu, Tianfu Wu, and Song Chun Zhu. Online object tracking, learning and parsing with And-Or graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3462–3469, 2014.

[119] Bo Li, Tianfu Wu, and Song-Chun Zhu. Integrating context and occlusion for car detection by hierarchical And-Or model. In *European Conference on Computer Vision*, pages 652–667. Springer, 2014.

[120] Tianfu Wu, Bo Li, and Song-Chun Zhu. Learning And-Or model to represent context and occlusion for car detection and viewpoint estimation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1829–1843, 2016.

[121] Long Zhu, Yuanhao Chen, A Torralba, W. Freeman, and A Yuille. Part and Appearance Sharing: Recursive Compositional Models for Multi-View Multi-Object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on*, pages 1919–1926, 2010.

[122] Björn Ommer and Joachim M Buhmann. Object categorization by compositional graphical models. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 235–250. Springer, 2005.

[123] Sanja Fidler, Marko Boben, and Ales Leonardis. Optimization framework for learning a hierarchical shape vocabulary for object class detection. In *BMVC*, 2009.

[124] Sanja Fidler, Marko Boben, and Aleš Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[125] Guillaume Bouchard and Bill Triggs. Hierarchical part-based visual object categorization. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 710–715. IEEE, 2005.

[126] Silvio Savarese and Fei-Fei Li. 3D generic object categorization, localization and pose estimation. In *ICCV*, pages 1–8, 2007.

[127] Boris Epshtein and Shimon Ullman. Feature hierarchies for object classification. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 220–227. IEEE, 2005.

[128] Long Zhu and Alan L Yuille. A hierarchical compositional system for rapid object detection. In *NIPS*, volume 5, page 3, 2005.

[129] Fabien Scalzo and Justus H Piater. Statistical learning of visual feature hierarchies. In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, pages III: 44–44, 2005.

[130] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[131] Samuel Krempp, Donald Geman, and Yali Amit. Sequential learning of reusable parts for object detection. Technical report, Citeseer, 2002.

[132] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[133] Shimon Ullman, Michel Vidal-Naquet, and Erez Sali. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, 2002.

[134] Bjorn Ommer, Michael Sauter, and Joachim M. Buhmann. Learning Top-Down Grouping of Compositional Hierarchies for Recognition. In *Proceedings of the 2006 Conf. on Computer Vision and Pattern Recognition Workshop*, CVPRW '06, pages 194–211, Washington, DC, USA, 2006. IEEE Computer Society.

[135] Sanja Fidler, Marko Boben, and Ales Leonardis. Learning hierarchical compositional representations of object structure. In S. Dickinson, A. Leonardis, B. Schiele, and M.J. Tarr, editors, *Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009.

[136] Andrew Zisserman, David Forsyth, Joseph Mundy, Charlie Rothwell, Jane Liu, and Nic Pillow. 3d object recognition using invariance. *Artificial Intelligence*, 78(1):239–288, 1995.

[137] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[138] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

[139] Ingo Wald and Vlastimil Havran. On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 61–69. IEEE, 2006.

[140] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3):201–227, 1992.

[141] Z Lian, J Zhang, S Choi, H ElNaghy, J El-Sana, T Furuya, A Giachetti, et al. Shrec'15 track: Non-rigid 3D shape retrieval. *3DOR*, 10:101–108, 2015.

[142] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *Asian conference on computer vision*, pages 525–538. Springer, 2012.

[143] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

[144] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 689–696. IEEE, 2009.

[145] Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[146] G Medioni and R Nevatia. Description of 3-d surfaces using curvature properties. In *Proc. DARPA Image Understanding Workshop*, pages 291–299. Science Applications Mc Lean, VA,, 1984.

[147] Baba C Vemuri, Amar Mitiche, and Jake K Aggarwal. Curvature-based representation of objects from range data. *Image and vision computing*, 4(2):107–114, 1986.

[148] Ivan Sipiran and Benjamin Bustos. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 27(11):963–976, 2011.

[149] Tsz-Wai Rachel Lo and J Paul Siebert. Local feature extraction and matching on range images: 2.5D SIFT. *Computer Vision and Image Understanding*, 113(12):1235–1250, 2009.

[150] Farzin Mokhtarian, Nasser Khalili, and Peter Yuen. Multi-scale free-form 3D object recognition using 3D models. *Image and Vision Computing*, 19(5):271–281, 2001.

[151] Bogdan Matei, Ying Shan, Harpreet S Sawhney, Yi Tan, Rakesh Kumar, Daniel Huber, and Martial Hebert. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1111–1126, 2006.

[152] Jiaxi Hu and Jing Hua. Salient spectral geometric features for shape matching and retrieval. *The Visual Computer*, 25(5):667–675, 2009.

[153] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608. IEEE, 2011.

[154] Lei Yunqi, Lai Haibin, and Jiang Xutuan. 3d face recognition by surf operator based on depth image. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 9, pages 240–244. IEEE, 2010.

[155] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[156] Xinju Li and Igor Guskov. Multiscale features for approximate alignment of point-based surfaces. In *Symposium on geometry processing*, volume 255, page 217, 2005.

[157] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1584–1601, 2006.

[158] Sotiris Malassiotis and Michael G Strintzis. Snapshots: A novel local surface descriptor and matching algorithm for robust 3D surface alignment. *IEEE Transactions on pattern analysis and machine intelligence*, 29(7), 2007.

[159] Umberto Castellani, Marco Cristani, Simone Fantoni, and Vittorio Murino. Sparse points matching by combining 3D mesh saliency with statistical descriptors. In *Computer Graphics Forum*, volume 27, pages 643–652. Wiley Online Library, 2008.

[160] Takeshi Masuda. Log-polar height maps for multiple range image registration. *Computer Vision and Image Understanding*, 113(11):1158–1169, 2009.

[161] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3D local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.

[162] Yueming Wang, Gang Pan, Zhaohui Wu, and Shi Han. Sphere-spin-image: A viewpoint-invariant surface representation for 3D face recognition. In *International Conference on Computational Science*, pages 427–434. Springer, 2004.

[163] H Quynh Dinh and Steven Kropac. Multi-resolution spin-images. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 863–870. IEEE, 2006.

[164] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009.

[165] Iasonas Kokkinos, Michael M Bronstein, Roee Litman, and Alex M Bronstein. Intrinsic shape context descriptors for deformable shapes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 159–166. IEEE, 2012.

[166] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision*, pages 356–369. Springer, 2010.

[167] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 373–380. IEEE, 2009.

[168] Fridtjof Stein and Gérard Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145, 1992.

[169] Yiyong Sun and Mongi A Abidi. Surface matching by 3D point's fingerprint. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 263–269. IEEE, 2001.

[170] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3D surf for robust three dimensional classification. In *European Conference on Computer Vision*, pages 589–602. Springer, 2010.

[171] Roee Litman and Alexander M Bronstein. Learning spectral descriptors for deformable shape correspondence. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):171–180, 2014.

[172] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 821–826, 2011.

[173] Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1704–1711. IEEE, 2010.

[174] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1626–1633. IEEE, 2011.

[175] James A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999.

[176] A Ben Hamza and Hamid Krim. Geodesic object representation and recognition. In *International conference on discrete geometry for computer imagery*, pages 378–387. Springer, 2003.

[177] Varun Jain and Hao Zhang. A spectral approach to shape-based retrieval of articulated 3D models. *Computer-Aided Design*, 39(5):398–407, 2007.

[178] Dirk Smeets, Thomas Fabry, Jeroen Hermans, Dirk Vandermeulen, and Paul Suetens. Isometric deformation modelling for object recognition. In *International Conference on Computer Analysis of Images and Patterns*, pages 757–765. Springer, 2009.

[179] Dirk Smeets, Jeroen Hermans, Dirk Vandermeulen, and Paul Suetens. Isometric deformation invariant 3D shape recognition. *Pattern Recognition*, 45(7):2817–2831, 2012.

[180] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[181] Zhouhui Lian, Afzal Godil, Benjamin Bustos, Mohamed Daoudi, Jeroen Hermans, Shun Kawamura, Yukinori Kurita, Guillaume Lavoué, Hien Van Nguyen, Ryutarou Ohbuchi, et al. A comparison of methods for non-rigid 3D shape retrieval. *Pattern Recognition*, 46(1):449–461, 2013.

[182] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3D shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156. IEEE, 2004.

[183] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Burtscher, Qiang Chen, Nihad Karim Chowdhury, Bin Fang, et al. A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding*, 131:1–27, 2015.

[184] Eric Paquet, Marc Rioux, Anil Murching, Thumpudi Naveen, and Ali Tabatabai. Description of shape information for 2-d and 3-d objects. *Signal processing: Image communication*, 16(1):103–122, 2000.

[185] C. Zhang and T Chen. Efficient feature extraction for 2D/3D objects in mesh representation. In *International Conference on Image Processing*, volume 3, pages 935–938. IEEE, 2001.

[186] Jonathan Corney, Heather Rea, Doug Clark, John Pritchard, Michael Breaks, and Roddy MacLeod. Coarse filters for shape matching. *IEEE Computer Graphics and Applications*, 22(3):65–74, 2002.

[187] Michael Elad, Ayellet Tal, and Sigal Ar. Content based retrieval of vrml objectsan iterative and interactive approach. In *Multimedia 2001*, pages 107–118. Springer, 2002.

[188] Yi Fang, Mengtian Sun, and Karthik Ramani. Temperature distribution descriptor for robust 3D shape retrieval. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 9–16. IEEE, 2011.

[189] Yi Fang, Jin Xie, Guoxian Dai, Meng Wang, Fan Zhu, Tiantian Xu, and Edward Wong. 3D deep shape descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2319–2328, 2015.

[190] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[191] Y. Liu, H. B. Zha, and H. Qin. Shape topics: A compact representation and new algorithms for 3D partial shape retrieval. In *CVPR*, pages II: 2025–2032, 2006.

[192] Dengsheng Zhang, Guojun Lu, et al. A comparative study on shape retrieval using fourier descriptors with different shape signatures. In *Proc. of International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)*, pages 1–9. Citeseer, 2001.

[193] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[194] Marcin Novotni and Reinhard Klein. 3D Zernike descriptors for content based shape retrieval. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 216–225. ACM, 2003.

[195] Hamid Laga, Hiroki Takahashi, and Masayuki Nakajima. Spherical wavelet descriptors for content-based 3D model retrieval. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, pages 15–15. IEEE, 2006.

[196] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-spectra as fingerprints for shape matching. In Leif Kobbelt and Vadim Shapiro, editors, *Symposium on Solid and Physical Modeling*, pages 101–106. ACM, 2005.

[197] Raif M Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233. Eurographics Association, 2007.

[198] Hedi Tabia, Hamid Laga, David Picard, and Philippe-Henri Gosselin. Covariance descriptors for 3D shape matching and retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4185–4192. IEEE, 2014.

[199] Ding-Yun Chen and Ming Ouhyoung. A 3D object retrieval system based on multi-resolution reeb graph. In *Proc. of Computer Graphics Workshop*, volume 16, 2002.

[200] Hari Sundar, Deborah Silver, Nikhil Gagvani, and Sven Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International, 2003*, pages 130–139. IEEE, 2003.

[201] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. GIFT: A real-time and scalable 3D shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5023–5032, 2016.

[202] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[203] Mohamed Chaouch and Anne Verroust-Blondet. Alignment of 3D models. *Graphical Models*, 71(2):63–76, 2009.

[204] Jie Feng, Yan Wang, and Shih-Fu Chang. 3D shape retrieval using a single depth image from low-cost sensors. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016.

[205] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013.

[206] Nima Sedaghat, Mohammadreza Zolfaghari, and Thomas Brox. Orientation-boosted voxel nets for 3D object recognition. *arXiv preprint arXiv:1604.03351*, 2016.

[207] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[208] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[209] Raoul Wessel and Reinhard Klein. Learning the compositional structure of man-made objects for 3D shape retrieval. In M. Daoudi, T. Schreck, M. Spagnuolo, I. Pratikakis, R. C. Veltkamp, and T. Theoharis, editors, *3DOR*, pages 39–46. Eurographics Association, 2010.

[210] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *European Conference on Computer Vision*, pages 1–14. Springer, 2010.

[211] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[212] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[213] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1697–1702. IEEE, 2012.

[214] Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering. In *AAAI*, pages 858–863, 2012.

[215] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[216] Domen Tabernik, Matei Kristan, Marko Boben, and Ales Leonardis. Learning statistically relevant edge structure improves low-level visual descriptors. In *ICPR*, pages 1471–1474. IEEE, 2012.

[217] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[218] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. In *Computer Vision–ECCV 2010*, pages 749–762. Springer, 2010.

[219] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[220] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[221] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[222] A. Zamolotskikh and P. Cunningham. An Assessment of Alternative Strategies for Constructing EMD-Based Kernel Functions for Use in an SVM for Image Classification. In *Content-Based Multimedia Indexing, 2007. CBMI '07. Int. Workshop on*, pages 11–17, 2007.

[223] Fengxue Li, Huaping Liu, Xinying Xu, and Fuchun Sun. Multi-modal local receptive field extreme learning machine for object recognition. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1696–1701. IEEE, 2016.

[224] Yanhua Cheng, Xin Zhao, Rui Cai, Zhiwei Li, Kaiqi Huang, and Yong Rui. Semi-supervised multimodal deep learning for RGB-D object recognition.

[225] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[226] Gabriel Peyré, Mickaël Péchaud, Renaud Keriven, and Laurent D Cohen. Geodesic methods in computer vision and graphics. *Foundations and Trends® in Computer Graphics and Vision*, 5(3–4):197–397, 2010.

[227] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[228] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 902–907. IEEE, 1995.

[229] James J Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3993–3998. IEEE, 2004.

[230] David L Davies and Donald W Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):224–227, 1979.

[231] Christopher Bingham. An antipodally symmetric distribution on the sphere. *The Annals of Statistics*, pages 1201–1225, 1974.

[232] Karsten Kunze and Helmut Schaeben. The Bingham distribution of quaternions and its spherical radon transform in texture analysis. *Mathematical Geology*, 36(8):917–943, 2004.

[233] Jack Glover and Sanja Popovic. Bingham procrustean alignment for object detection in clutter. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2158–2165. IEEE, 2013.

[234] Sebastian Riedel, Zoltan-Csaba Marton, and Simon Kriegel. Multi-view orientation estimation using bingham mixture models. In *Automation, Quality and Testing, Robotics (AQTR), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[235] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[236] Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoua, P Dp Suetens, et al. Shape retrieval on non-rigid 3D watertight meshes. In *Eurographics Workshop on 3D Object Retrieval (3DOR)*, 2011.

[237] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. Shape retrieval contest 2007: Watertight models track. *SHREC competition*, 8(7), 2007.

[238] Juan Zhang, Kaleem Siddiqi, Diego Macrini, Ali Shokoufandeh, and Sven Dickinson. Retrieving articulated 3D models using medial surfaces and their graph spectra. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 285–300. Springer, 2005.