UNIVERSITY OF CALIFORNIA

Los Angeles

Recursive Composition for Modeling, Inference and Learning in Computer Vision

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Long Zhu

2008

The dissertation of Long Zhu is approved.

Luminita Vese

Pietro Perona

Songchun Zhu

Yingnian Wu

Alan Yuille, Committee Chair

University of California, Los Angeles

2008

To my parents, and my wife Cindy

TABLE OF CONTENTS

I Background **1**

1 Overview **2**

1.1 Motivation 2

1.2 Thesis Summary 3

1.3 Outline of the Thesis 6

1.4 The Track of our Research Program 7

2 Vision Tasks, Evaluation Criterions and Datasets **9**

2.1 Object Categorization, Detection, Segmentation and Parsing 9

2.1.1 Object Categorization 9

2.1.2 Object Detection 9

2.1.3 Object Segmentation 10

2.1.4 Object Parsing: Matching and Alignment 10

2.1.5 Image Parsing: Segmentation and Scene Labeling 11

2.2 Datasets 11

2.2.1 Caltech-101 Dataset 11

2.2.2 Weizmann Horse Dataset 11

2.2.3 Multi-view Face Alignment Dataset 12

2.2.4 Berkeley Human Baseball Dataset 12

2.2.5 MSRC Dataset 12

II Non-Recursive Representation and Learning **14**

3 Probabilistic Grammar-Markov Model **15**

3.1 Introduction 16

3.2 Background 20

3.3 The Image Representation: Features and Oriented Triplets 22

3.3.1 The Image Features 23

3.3.2 The Oriented Triplets 24

3.4 Probabilistic Grammar-Markov Model 25

3.5 The Distribution defined on the PGMM 27

3.5.1 Generating the leaf nodes: $P(y|\Omega)$ 30

3.5.2 Generating the observable leaf nodes: $P(u|y,\omega^g)$ 31

3.5.3 Generating the positions and orientation of the leaf nodes: $P(z,\theta|y,\omega^g)$. 31

3.5.4 The Appearance Distribution $P(A|y,\omega^A)$. 34

3.5.5 The Priors: $P(\Omega), P(\omega^A), P(\omega^g)$. 34

3.5.6 The Correspondence Problem: 34

3.6 Learning and Inference of the model 36

3.6.1 Dynamic Programming for the Max and Sum 37

3.6.2 EM Algorithm for Parameter Learning 40

3.6.3 Structure Pursuit 41

3.6.3.1 The appearance and triplet vocabularies 42

3.6.3.2 Structure Induction Algorithm 44

3.7 Experimental Results 46
3.7.1 Learning Individual Objects Models 47
3.7.2 Invariance to Rotation and Scale 49
3.7.3 The Advantages of Variable Graph Structure 54
3.8 Discussion 57

III Object Parsing by Recursive Deformable Template 59

4 Learning a Recursive Deformable Template Model 60
4.1 Introduction 61
4.2 Background 64
4.3 Recursive Deformable Template Model (RDTM) 67
4.3.1 The Graphical Structure of the RDTM 67
4.3.2 The state variables and the potential functions 67
4.3.3 Constructing the Hierarchy by One-example Learning 71
4.4 Inference: Parsing the Model 72
4.5 Structure-Perceptron Learning 76
4.5.1 Background on Perceptron and Structure-Perceptron Learning 76
4.5.2 Structure-Perceptron Learning 77
4.5.3 Averaging Parameters 78
4.5.4 Feature Selection 80
4.6 Experimental Results 82
4.6.1 Dataset and Evaluation Criterions 82

4.6.2 Experiment I: One-example Learning 84
4.6.3 Experiment II: Contributions of Object Parts: Complexity and Performance Analysis 86
4.6.4 Experiment III: Evaluations of Structure-Perceptron Learning for Deformable Object Detection, Segmentation and Parsing . 89
4.6.5 Experiment IV: Diagnosis of structure-perceptron learning . . 91
4.6.6 Experiment V: Multi-view Face Alignment 92
4.7 Conclusion 94

5 Unsupervised Learning: Recursive Composition, Suspicious Coincidence and Competitive Exclusion 96
5.1 Introduction 97
5.2 Background: Hierarchical Structure Learning 100
5.3 The Recursive Deformable Template Model (RDTM) and the Inference algorithm 101
5.3.1 The Recursive Deformable Template Model (RDTM) 101
5.3.2 The Inference Algorithm for parsing when RDTM is known . 104
5.4 Unsupervised Structure Learning 104
5.4.1 The Bottom-Up Process 105
5.4.2 The Top-Down Process 109
5.5 Experimental Results 111
5.6 Conclusion 117

6 AND/OR Graph Learning 119

6.1 Introduction 120
6.2 Background 123
6.2.1 Object Representation 123
6.2.2 Human Body Parsing 124
6.2.3 Max Margin Structure Learning 125
6.3 The AND/OR Graph Representation 126
6.3.1 The Topological Structure of the AND/OR Graph 126
6.3.2 The Representational Power of the AND/OR Graph Representation 127
6.3.3 The State Variables 129
6.3.4 The Potential Functions for the AND/OR Graph 130
6.4 The Inference/Parsing Algorithm 133
6.5 Max Margin AND/OR Graph Learning 136
6.5.1 Primal and Dual Problems 136
6.5.2 Optimization of the Dual 139
6.6 Experiments 143
6.6.1 Datasets and Implementation Details. 143
6.6.2 Performance of the AND/OR graph on the Horse dataset . . . 145
6.6.3 Computational Complexity Analysis 148
6.6.4 Human Body Parsing 149
6.7 Discussion 151

IV Image Parsing by Recursive Segmentation - Recognition Tem-

plate **155**

7 Image Understanding by Recursive Segmentation and Recognition Template **156**

7.1 Introduction 157

7.2 Hierarchical Image Model 160

7.2.1 The Model 160

7.2.2 Parsing by Dynamic Programming 166

7.2.3 Learning the Model 167

7.3 Experimental Results 169

7.4 Conclusion 172

V Concluding Remarks **174**

8 Conclusions and Future Directions **175**

References **176**

LIST OF FIGURES

1.1 This figure shows recursive deformable templates for horses at different levels. The object template consists of a small number of small sub-templates at lower level. The sub-templates consist of smaller subsub-templates, and so on. Observe how the vocabulary contains "generic" shapes at low levels, but horse specific parts at the higher levels. These templates are learnt automatically as described in chapter 5. 5

3.1 Probabilistic Context Free Grammar. The grammar applies production rules with probability to generate a tree structure. Different random sampling will generate different tree structures. The production rules are applied independently on different branches of the tree. There are no sideways relations between nodes. 18

3.2 This chapter uses triplets of nodes as building blocks. We can grow the structure by adding new triangles. The junction tree (the far right panel) is used to represent the combination of triplets to allow efficient inference. 19

3.3 Ten of the object categories from Caltech 101 which we learn in this chapter. 20

3.4 The oriented triplet is specified by the internal angles β, the orientation of the vertices θ, and the relative angles α between them. 22

3.5 This figure illustrates the features and triplets without orientation (left two panels) and oriented triplets (next two panels). 23

3.6 Graphical Models. Squares, triangles, and circles indicate AND, OR, and LEAF nodes respectively. The horizontal lines denote MRF connections. The far right panel shows the background node generating leaf nodes. The models for $O1$ for panels 2,3 and 4 correspond to the triplets combinations in figure (3.2). See text for notation. 26

3.7 This figure illustrates the dependencies between the variables. The variables Ω specify the probability for topological structure y. The spatial assignments z of the leaf nodes are influenced by the topological structure y and the MRF variables ω. The probability distribution for the image features x depends on y, ω and z. 29

3.8 This figure illustrates structure pursuit. a)image with triplets. b) one triplet induced. c) two triplets induced. d) three triplets induced. Yellow triplets: all triplets from triplet vocabulary. Blue triplets: structure induced. Green triplets: possible extensions for next induction. Circles with radius: image features with different sizes. 43

3.9 Structure Induction Algorithm 45

3.10 We report the classification performance for 26 classes which have at least 80 images. The average classification rate is 87.6%. 49

3.11 Individual Models learnt for Faces, Motorbikes, Airplanes, Grand Piano and Rooster. The circles represent the AF's. The numbers inside the circles give the a index of the nodes, see Table (3.1). The Markov Random Field of one aspect of Faces, Roosters, and Grand Pianos are shown on the right. 52

3.12 Parsed Results for Faces, Motorbikes and Airplanes. The circles represent the AF's. The numbers inside the circles give the a index of the nodes, see Table (3.1). 53

3.13 Parsed Results: Invariant to Rotation and Scale. 53

3.14 Analysis of the effects of adding OR nodes. Observe that performance rapidly improves, compared to the single MRF model with only one aspect, as we add extra aspects. But this improvement reaches an asymptote fairly quickly. (This type of result is obviously dataset dependent). 54

3.15 Hybrid Model learnt for Faces, Motorbikes and Airplanes. 56

4.1 Alternative representations for deformable objects. Left panel: standard models use "flat" Markov Random Field (MRF) models relying on sparse cues and with limited spatial interactions. Middle panel: a RDTM has a hierarchical representation of a large variety of different images cues and spatial interactions at a range of scales. Right panel: The points along the object boundary correspond to the nodes in the "flat" MRF models or the leaf nodes of the hierarchical model. 62

4.2 The hierarchical graph of the RDTM is constructed by a hierarchical clustering algorithm (see text for details). Black dots indicate the positions of the leaf nodes in the hierarchy. Color dots indicate the subparts which correspond to particular nodes of the hierarchy. The appearance and shape deformation are modeled at multiple levels in the hierarchy. 68

4.3 Representation based on oriented triplet. This gives the cliques for the four children of a node. In this example, four triplets are computed. Each circle corresponds to one node in the hierarchy which has a descriptor (indicated by blue line) of position, orientation and scale. The potentials of the cliques are Gaussians defined over features extracted from triple nodes, such as internal angles of the triangle and relative angles between the feature orientation and the orientations of the three edges of the triangle. These exacted features are invariant to scale and rotation. 71

4.4 This snapshot illustrates the bottom-up inference algorithm. The bottom-up process starts from level 1 and constructs proposals from children nodes. Only proposals above threshold are kept. Similar proposals in the same local window defined over space, orientation and scale are grouped together. See the proposals inside the windows. The proposal with the highest score within this cluster is kept to propagate to upper level. 74

4.5 The inference algorithm. $\oplus$ denotes the operation of combining two proposals. 76

4.6 Algorithm I: a simple training algorithm of structure-perceptron learning 79

4.7 Algorithm II: a modification of algorithm I. 79

4.8 Examples of the Weizmann horse data set. This figure shows input image, ground truth of segmentation, parsing (position of leaf nodes) and detection, from left to right respectively. 81

4.9 This shows the exemplars used for the horse (left) and the cow (right). 84

4.10 Segmentation and parsing results on the horse and cows datasets. The first column shows the raw images. The second one show the edge maps. The third one shows the parsed result. The last one shows the segmentation results. 85

4.11 This figure shows the Precision-Recall curves for different levels. Level 4 is the top level. Level 0 is the bottom level. "Human" curve is provided as an ideal decision maker for comparison. 87

4.12 Examples of Parsing and Segmentation. Column 1 , 2 and 3 show the raw images, parsing and segmentation results respectively. Column 4 to 6 show extra examples. Parsing is illustrated by dotted points which indicate the positions of leaf nodes (object parts). Note that the points in different images with the same color correspond to the same semantical part. 90

4.13 The average position errors (y-axis) across iterations (x-axis) are compared between Algorithm-II(average) and Algorithm-I (non-average). 91

4.14 The average positions errors on training, validation and testing dataset are reported. 92

4.15 Weights of Features. The most useful features overall are gray value, magnitude and orientation of gradient, and difference of intensity along horizontal and vertical directions (Ix and Iy). DooG1 Ch5 means Difference of offset Gaussian (DooG) at scale 1 (13*13) and channel (orientation) 5 ($\frac{4}{6}\pi$). 93

4.16 Multi-view Face Alignment. 94

5.1 Left:The learning is unsupervised in the sense that we are given a training dataset of images containing the object in cluttered backgrounds but we do not know the position or boundary of the object. Right: The hierarchical representation of the object. The boxes represent non-leaf nodes. The circles denote leaf nodes that directly relate to properties of the input image. The topology of the structure is not given, but learnt in an unsupervised way. 98

5.2 This figure illustrates the procedure of the unsupervised learning. The first column shows the responses of the features from the "vocabulary" at level l. the second column shows the compositions at step 1). the third column shows the clustering at step 2. The noise(non-regular) pattern bounded by dotted line is pruned out by step 4. The last column plots the results after step 5. At step 5, the compositions, which are constructed by different components, but parse the same areas in the image, are grouped together (the maximum is kept 107

5.3 Bottom-up and Top-down Learning. 110

5.4 In the top two rows, Columns 1 ,3 and 5 show parse results of our method followed by segmentation results. The parse result is illustrated by the colored dots which correspond to the leaf nodes of the object. The correspondences are consistent for different images. Rows 3 and 4 show the parse results of our method and OBJ CUT (cropped from [1]) on 5 images used in [1] respectively. Note our method obtains more complete boundary. 113

5.5 Top: The hierarchy shows the learnt hierarchical model. The colored rectangles highlight the identities of the structures. All parts are obtained at the bottom-up stage except that the rectangles in the dotted boxes show the parts of the model that were learnt in the top-down stage. Bottom: The rows illustrate structures at levels 4,3,3,2,2 (i.e. top row is level 4, next row is level 3,...). The first column gives the structure (with three children colour coded). The second column shows the structure detected on a specific image. The third, fourth, and fifth columns show the proposals for the sub-structure – colour coded. 115

5.6 Top: This figure shows elements of the vocabulary for horses at different levels (mean values only). Observe how the vocabulary contains "generic" shapes at low levels, but finds horse specific parts at the higher levels. Bottom: This figure shows the histogram of concepts at the different levels of the dictionary. A point in a curve quantifies the number of concepts which have a certain number of instances. 116

5.7 This figure illustrates the generality of our approach. We show some typical training images which contain cluttered background, different shapes and deformations. The red-sketch images show the learnt models. 118

6.1 The AND/OR representation allows us to model enormous poses of the object. A parse tree which is an instantiation of the AND/OR graph represents a specific pose of the human body. The nodes and edges in red indicate one parse tree. In this chapter, there are 98 poses which can be modeled by the parse trees of the whole AND/OR graph. . . . 121

6.2 The AND/OR graph is an efficient way to represent different appearances of an object. The graph was hand specified.. The bottom level of the graph indicates points along the boundary of human body. The higher levels indicate combinations of elementary configurations. The graph that we used contains eight levels (three lower levels are not depicted here due to lack of space). Color points distinguish different body parts. The arms are not modeled in this chapter (or in related work in the literature). 128

6.3 The AND/OR graph for horses. There are 40 poses allowed in this chapter. The first (typical) pose in the top node will be used for single hierarchy by fixing the child of all OR nodes. 128

6.4 The inference algorithm. The operation LocalMaximum implements surround suppression. 135

6.5 Working Set Optimization 141

6.6 Sequential Minimal Optimization (SMO) 142

6.7 Pair Selection in SMO 142

6.8 This figure is best viewed in color. Columns from (a) to (d) show the parsing and segmentation results obtained by hierarchy and AND/OR graph models respectively. The colored dots correspond to the leaf nodes of the object. 147

6.9 The first column shows the parse results of the human body. Color points indicate the positions of body parts. The same color points in different images correspond to the same parts. The second column show the segmentations of human body. The next four columns show extra examples. 150

6.10 We compare our results with that of Srinivasan and Shi [2]. The performance of parsing (position error) and segmentation (overlap score) are shown in the top and bottom figures respectively. Note that [2] select the best one (manually)of the top parses. 152

6.11 Convergence Analysis. We study the behavior of max margin training. The first panel shows the convergence curve in terms of the objective function defined in equation (6.19). The second panel shows the converge curves of the average position error evaluated on training and testing set. 153

7.1 The left panel shows the structure of the Hierarchical Image Model. The grey circles are the nodes of the hierarchy. All nodes, except the top node, have only one parent nodes. All nodes except the leafs are connected to four child nodes. The middle panel shows a dictionary of 30 segmentation templates. The color of the sub-parts of each template indicates the object class. Different sub-parts may share the same label. For example, three sub-parts may have only two distinct labels. The last panel shows that the ground truth pixel labels (upper right panel) can be well approximated by composing a set of labeled segmentation templates (bottom right panel). 161

7.2 This figure illustrates how the segmentation templates and object labels (S-R pair) represent image regions in a coarse-to-fine way. The left figure is the input image which is followed by global, mid-level and local S-R pairs. The global S-R pair gives a coarse description of the object identity (horse), its background (grass), and its position in the image (central). The mid-level S-R pair corresponds to the region bounded by the black box in the input image. It represents (roughly) the shape of the horse's leg. The four S-R pairs at the lower level combine to represent the same leg more accurately. 163

7.3 Structure-perceptron learning 169

7.4 This figure is best viewed in color. The colors indicate the labels of 21 object classes as in [3]. The columns (except the fourth "accuracy" column) show the input images, ground truth, the labels obtained by HIM and the boosting classifier respectively. The "accuracy" column shows the global accuracy obtained by HIM (left) and the boosting classifier (right). In these 7 examples, HIM improves boosting by 1%, -1% (an outlier!), 1%, 10%, 18%, 20% and 32% in terms of global accuracy. 171

7.5 The S-R pairs can be used to parse the object into parts. The colors indicate the identities of objects. The shapes (spacial layout) of the segmentation templates distinguish the constituent parts of the object. Observe that the same S-R pairs (e.g. stomach above grass, and tail to the left of grass) correspond to the same object part in different images. 173

LIST OF TABLES

3.1 The notations used for the PGMM. 28

3.2 We have learnt probability grammars for 13 objects in the Caltech database, obtaining scores over 90% for most objects. A score of 90%, means that we have a classification rate of 90% and a false positive rate of $10\%(10\% = (100 - 90)\%)$. We compare our results with constellation model 50

3.3 Localization rate is used to measure the proportion of AF's of the model that lie within the groundtruth bounding box. 51

3.4 Invariant to Rotation and Scale 51

3.5 The PGMM are learnt on different training datasets which consist of a random mixture of images containing the object and images which do not. 55

3.6 The PGMM can learn a hybrid class which consists of faces, airplanes, and motorbikes. 56

4.1 The performance of the RDTM with one-example learning. 84

4.2 Analysis of the Bottom-Up Processing. The numbers of clusters for each node and proposals for each cluster at different levels are compared in columns 2 and 3. Time costs for each node and the whole image are listed in the last two columns. The last row shows the numbers averaged over the nodes of all the levels of the hierarchy. 87

4.3 Comparisons of one-example learning and structure-perceptron learning 88

4.4 Comparisons of Segmentation Performance on Weizmann Horse Dataset 88

5.1 The table shows the notation for the learning algorithm. 107

5.2 Results on horse dataset. Columns 2 and 3 give the size of images taken for training and testing. Column 4 quantifies the segmentation accuracy. The last column shows the average time taken for one image. Note that [1] is tested on only 5 images and [4] assumes the position of the object is roughly given. 112

5.3 Columns from 2 to 5 show the numbers of proposals after composition (step 1), clusters after clustering (step 2), after pruning (step 4) and after competitive exclusion (step 5) respectively. The next column shows the time (seconds) taken for each level. Level 0 shows the results of the grouping of the edge points(360 degrees are divided into 4 angle ranges). all numbers are calculated over 12 real images in the training dataset. 117

6.1 Performance for parsing, segmentation and detection. The table compares the results for the hierarchial model (without OR nodes), AND/OR graph and other alternative methods. 148

6.2 Complexity Analysis. This table shows the numbers of proposals and time costs at different levels. 149

7.1 Performance Comparisons for average accuracy and global accuracy. "Classifier only" are the results where the pixel labels are predicted by the classifier obtained by boosting only. 172

ACKNOWLEDGMENTS

I would like to thank my advisor Alan Yuille for his vision, patience and advice during my studies at UCLA. I would like to thank Songchun Zhu, Yingnian Wu, Zhuowen Tu, Luminita Vese and Pietro Perona for helpful discussions.

In the last two years, it has been my great honor to collaborate with Yuanhao Chen and Chenxi Lin. I also would like to thank my officemate Songfeng Zheng, Ziqiang Liu and all other people in the CIVS lab at UCLA.

Finally, I would like to thank my family.

VITA

1997 — 2001 B.S. (Computer Science) , Northeastern University, China

2003 — 2008 Graduate Student Researcher, Department of Statistics, UCLA.

PUBLICATIONS

S. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, H. Shum, "Statistical Learning of Multi-view Face Detection," in Proceedings of The European Conference on Computer Vision, 2002.

R. Xiao, L. Zhu, H. Zhang, "Boosting Chain Learning for Object Detection," in Proceedings of IEEE International Conference on Computer Vision, 2003.

L. Zhu and A. L. Yuille, "A hierarchical compositional system for rapid object detection," in Advances in Neural Information Processing Systems, 2005.

L. Zhu, Y. Chen, and A. L. Yuille, "Unsupervised learning of a probabilistic grammar for object detection and parsing," in Advances in Neural Information Processing Systems, 2006,

L. Zhu, and A. Yuille, "Unsupervised learning of probabilistic grammar-markov models for object categories," IEEE Transactions on Pattern Analysis and Machine Intelli-

gence, 2008.

Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang, "Rapid inference on a novel and/or graph for object detection, segmentation and parsing," in Advances in Neural Information Processing Systems, 2007.

L. Zhu, Y. Chen, X. Ye, and A. L. Yuille, "Structure-perceptron learning of a hierarchical log-linear model," in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008.

L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille, "Max margin and/or graph learning for parsing the human body," in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008.

Y. Chen, L. Zhu, A. L. Yuille, and H. Zhang, "Unsupervised Learning of Probabilistic Object Models for Object Classification, Segmentation and Recognition," in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008.

L. Zhu, C. Lin, Y. Chen, H. Huang, and A. L. Yuille, "Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion," in Proceedings of The 10th European Conference on Computer Vision, 2008.

L. Zhu, Y. Chen, Y. Lin, and A. L. Yuille, "Recursive Segmentation and Recognition Templates for 2D Parsing," in Advances in Neural Information Processing System, 2008.

ABSTRACT OF THE DISSERTATION

Recursive Composition for Modeling, Inference and Learning in Computer Vision

by

Long Zhu

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2008

Professor Alan Yuille, Chair

Image understanding depends critically on learning visual models which can be formulated in terms of probabilistic structured representations. But the two-dimensional nature of images makes it much hard to design efficient image understanding systems and the form of the representations is also unclear. The key assumption used to address these problems in this thesis is that the visual world is recursively compositional. This enables us to construct representations, by recursively combining elementary stochastic templates, and makes inference and unsupervised learning practical. By introducing elementary visual constituents of recursion properties, we design recursively compositional systems for two basic image understanding problems, deformable object parsing and image parsing.

We first propose a recursive deformable template model to represent objects in a hierarchical form. The object template consists of a small number of small sub-templates which is composed by smaller subsub-templates, and so on. The composition of templates and their deformation are imposed at different levels to capture both short-range and long-range correlations. The recursive design enables us to perform rapid parsing by dynamic programming, efficient unsupervised learning by recursive composition

and supervised training by structured -perceptron. The learnt model together with the inference algorithms are capable of performing different vision tasks simultaneously, such as object detection, segmentation and parsing (e.g. matching/alignment of object parts).

Next we extend the recursive deformable template model to a novel AND/OR graph representation for parsing articulated objects into parts and recovering their poses. The recursion design in the AND/OR graph allows us to handle an enormous variety of articulated poses with a compact graphical model where the rapid inference can be performed. We present a novel structure-learning method, Max Margin AND/OR Graph (MM-AOG), to learn the parameters of the AND/OR graph model discriminatively.

Finally, we present a recursive segmentation and recognition template model for 2D image parsing. This representation consists of recursive segmentation-recognition templates which account for image segmentation and object recognition simultaneously at multiple layers. The recursive templates result in a coarse-to-fine representation which is capable of capturing long-range dependency and exploiting different levels of contextual information. The recursive structure also allows us to design a rapid inference algorithm, based on dynamic programming, which enables us to parse the image rapidly in polynomial time, and learn the model efficiently in a discriminative manner.

We demonstrate the significant benefits of the recursive design on several vision tasks including deformable articulated object detection, parsing and segmentation, and image segmentation and scene understanding. Our experiments on the challenging public datasets show that the recursively compositional systems achieve the state-of-the-art performance.

Part I

Background

CHAPTER 1

Overview

1.1 Motivation

Vision is a pre-eminent pattern theoretic problem. The difficulty of vision arises from the complexity and ambiguity of natural images (notoriously models designed using synthetic stimuli almost never scale-up to work on realistic images). The importance, and difficulty, of visual perception can be appreciated by realizing that the optic nerve is the largest nerve in the body and the visual cortex is estimated to be roughly half the size of the entire cortex. Hence designing a system that can perform the main tasks of vision detecting and recognizing objects and interpreting images and scenes is equivalent to a computational understanding of at least half the cortex. Moreover, recent computational studies suggest that similar underlying principles underly all aspects of cognition and that a deep understanding of vision will translate to other domains.

But how is vision possible? And, in particular, how can an infant or a robot learn to decode the complexity of real world images? It has been argued that vision, and other cognitive tasks, can be formulated in terms of probabilistic inference on structured representations [5]. But how can these representations and probability distributions be learnt in an unsupervised way from real images? How can we perform efficient inference on these probability distributions? How can the system scale to deal with the large numbers of different objects in the world and the enormous variety of images and visual scenes? These are critical questions that we address in this thesis.

This thesis will present a strategy for addressing all of these problems which are tested on large image datasets. The key assumption is that the visual world is recursively compositional. This enables us to construct representations, by combining elementary stochastic structures (e.g. templates) enabling scalability, and makes inference and unsupervised learning practical. By introducing elementary visual constituents of recursion and composition properties, we design recursively compositional systems for two basic image understanding problems, deformable object parsing and image parsing. We note that our approach leads to networks that differ from traditional artificial neural networks but which can instead propose neural architectures based on computational requirements, in the spirit of Valiants work on circuits of the mind [6]. We hope, and anticipate, that the techniques we develop can be applied to other pattern theory problems.

1.2 Thesis Summary

Image understanding depends critically on learning visual models which can be formulated in terms of probabilistic structured representations. But the two-dimensional nature of images makes it much hard to design efficient image understanding systems and the form of the representations is also unclear. The key assumption to address these problems is that the visual world is recursively compositional. This enables us to construct representations, by recursively combining elementary stochastic templates , and makes inference and unsupervised learning practical. By introducing elementary visual constituents of recursion properties, we design recursively compositional systems for two basic image understanding problems, deformable object parsing and image parsing.

We first introduce our previous attempt to object modeling. We address the issues of object representation and unsupervised learning. We present a Probabilistic

Grammar-Markov Model (PGMM) which couples probabilistic context free grammars and Markov Random Fields. PGMMs are designed so that they can perform rapid inference, parameter learning, and the more difficult task of structure induction. But PGMMs do not rely on the recursion.

After introducing PGMMs, we propose a recursive deformable template model to represent objects in a hierarchical form. The object template consists of a small number of small sub-templates which is composed by smaller subsub-templates, and so on. See the examples of recursive deformable templates in figure 1.1. The composition of templates and their deformation are imposed at different levels to capture both short-range and long-range correlations. The recursive design enables us to perform rapid inference (parsing) by dynamic programming, efficient unsupervised learning by recursive composition, supervised training by structured -perceptron. The learnt models together with the inference algorithms are capable of performing different vision tasks simultaneously, such as object detection, segmentation and parsing (e.g. matching/alignment of object parts).

Next we extend the recursive deformable template model to a novel AND/OR graph representation for parsing articulated objects into parts and recovering their poses. The recursion design in the AND/OR graph allows us to handle an enormous variety of articulated poses with a compact graphical model where the rapid inference can be performed. We present a novel structure-learning method, Max Margin AND/OR Graph (MM-AOG), to learn the parameters of the AND/OR graph model discriminatively.

Finally, we present a recursive segmentation and recognition template model for 2D image parsing. This representation consists of recursive segmentation-recognition templates which account for image segmentation and object recognition simultaneously at multiple layers. The recursive templates result in a coarse-to-fine represen-

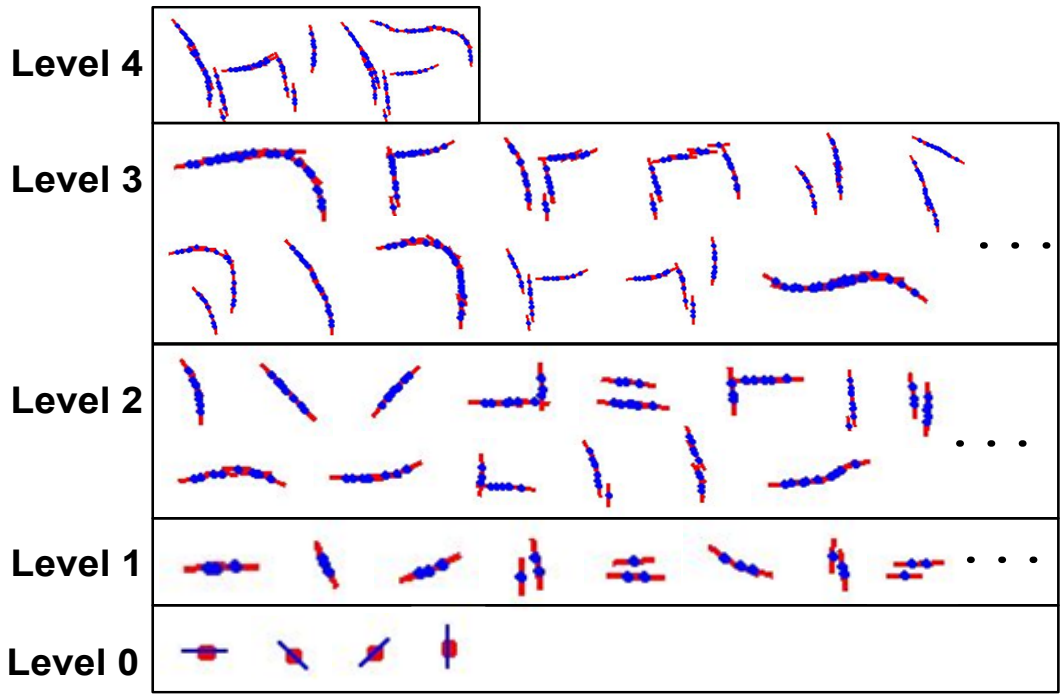


Figure 1.1: This figure shows recursive deformable templates for horses at different levels. The object template consists of a small number of small sub-templates at lower level. The sub-templates consist of smaller subsub-templates, and so on. Observe how the vocabulary contains "generic" shapes at low levels, but horse specific parts at the higher levels. These templates are learnt automatically as described in chapter 5.

tation which is capable of capturing long-range dependency and exploiting different levels of contextual information. The recursive structure also allows us to design a rapid inference algorithm, based on dynamic programming, which enables us to parse the image rapidly in polynomial time, and learn the model efficiently in a discriminative manner.

We demonstrate the advantages of the recursive design on several vision tasks including deformable articulated object detection, parsing and segmentation, and image segmentation and scene understanding. Our experiments on challenging public datasets show that the recursively compositional systems achieve the state-of-the-art performance. We stress that we are using a general approach which is suitable to all problems, rather than a limited task specific approach.

1.3 Outline of the Thesis

Part I of this thesis consists of Chapters 1 and 2, which are the overview and general background.

Chapter 2 briefly introduces the vision tasks that we address in this thesis and their evaluation criteria. We also describe some public datasets which are used for evaluations.

In Part II, we describe our first attempt to addressing the object modeling and unsupervised learning which motivates our later development on recursive design.

Part III consists of Chapter 4 to 7 in which we develop the recursion and composition based framework including the recursive deformable template model, compositional inference, supervised structure-perceptron learning, and unsupervised learning by recursive composition. We show that our experimental results are superior to those obtained by many other competing methods in deformable object segmentation and

parsing. Chapter 7 further extends the representation to a novel AND/OR graph model and presents max-margin structure learning for estimating the parameters of the model. The learnt AND/OR model results in significant improvement for the task of articulated object parsing, e.g, human body parsing.

Part IV applies the recursion composition principle to the image parsing problem. We present a recursive segmentation and recognition template model for 2D image parsing. The recursive structure naturally leads to a rapid inference algorithm, based on dynamic programming, which enables us to parse the image rapidly in polynomial time. We describe the structure-perceptron training of the model in a discriminative manner. The proposed approach achieves the state-of-the-art performance on the tasks of image segmentation and scene labeling.

Part V concludes the thesis and suggests future research directions.

1.4 The Track of our Research Program

We started our journey for building hierarchical compositional system in NIPS 2005 [7]. Though it is not presented in this thesis, it was valuable as a starting point. The system had a compositional form, but had no recursion in the design. Nevertheless, the idea of non-maximum suppression in this work, which was proposed for rapid inference, inspired the recursive design in the later models.

Chapter 3 is based on our work in NIPS 2006 [8] and [9] where we abandoned the spirit of recursion and composition temporarily, but provided a triplet shape descriptor which is used later in our recursive models as a basic building block for shape representation. We also studied the issues occurring in the unsupervised structure induction which strongly pushed us back to the track of the recursion and composition design. This lead to additional collaborative work (POMs) [10] that is part of the thesis of

Chen and is not reported in this thesis.

Chapters 4 and 5 draw from our work in NIPS 2007 [11] and CVPR 08 [12] where we returned to the hierarchical design, proposed the recursive deformable template model fully based on the recursion and composition principle, and demonstrated a successful system for deformable object parsing and segmentation. Chapter 6 is based on our work to be presented in ECCV 2008 [13] where two learning principles are proposed to overcome the issues in unsupervised learning discovered in our first attempt [7]. Charter 7 is based on our work in CVPR 2008 [14] which is an extension from the task of deformable object parsing to articulated object parsing.

Chapter 8 is based on our work to be presented at NIPS 2008 [15] which is another design following the same recursion and composition principle, but used for a different task, i.e. image parsing.

CHAPTER 2

Vision Tasks, Evaluation Criterions and Datasets

In this thesis, we are interested in both object modeling and image understanding. We study a range of basic vision tasks. In particular, for object modeling, we exploit the tasks of object categorization, detection, segmentation and parsing. For image understanding, we focus on the task of image segmentation and scene labeling. This chapter will introduce these tasks and their evaluation criterions. The datasets used for testing the algorithms on these tasks will be described.

2.1 Object Categorization, Detection, Segmentation and Parsing

2.1.1 Object Categorization

The task of object categorization is to classify an image containing the object versus purely background images. In this task, we are not required to predict the position of the object. The **classification rate** (=1-error rate) is used to evaluate the categorization performance.

2.1.2 Object Detection

Object detection is a basic vision task where we are interested in locating the object by the estimation of the center and size of the object. We use **detections rate** to quantify the proportion of successful detections. We rate *detection* to be successful if the area

of intersection of the labeled object region (obtained the object detectors) and the true object region is greater than half the area of the union of these regions.

2.1.3 Object Segmentation

Object segmentation requires the segmentation algorithms to label every pixel in the input image to be object or non-object (background). We use **segmentation accuracy** to quantify the proportion of the correct pixel labels (object or non-object). Although segmentation accuracy is widely used as a measure for segmentation, it has the disadvantage that it depends on the relative size of the object and the background. Therefore, to overcome the shortcoming of segmentation accuracy, we also report **precision/recall**, see [16], where $precision = \frac{P \cap TP}{P}$ and $recall = \frac{P \cap TP}{TP}$ (P is the set of pixels which are classifier as object by the segmentation algorithm and TP is the set of object pixels in ground truth). We note that segmentation accuracy is commonly used in the computer vision community, while precision/recall is more standard in machine learning.

2.1.4 Object Parsing: Matching and Alignment

Object parsing (e.g. matching and alignment) is a task to predict the pose (position, orientation and orientation) of the object and its parts. To evaluate the performance of parsing and matching/alignment, we use the **average position error** measured in terms of pixels. This quantifies the average distance between the positions of object parts (e.g. eyes, mouth, nose for face alignment, and head, leg, tail for horse parsing) of the ground truth and those estimated by the parsing algorithms.

2.1.5 Image Parsing: Segmentation and Scene Labeling

In this thesis, image parsing is to perform both image segmentation and object recognition. More specifically, we want to label every image pixels to one of object classes. To evaluate the performance of parsing we use the **global accuracy** measured in terms of all pixels and the **average accuracy** over a number of object classes (global accuracy pays most attention to frequently occurring objects and penalizes infrequent objects).

2.2 Datasets

2.2.1 Caltech-101 Dataset

The Caltech-101 dataset [17] consists of images from 101 object categories and an additional background class. Each category contains 40 to 800 images (typically less than 100). Most categories have about 50 images. The size of each image is roughly 300 x 200 pixels. There is significant variation in color, pose and lighting. But the size and position of objects do not vary a lot. In most cases, objects appear in the center of the image and occupy one third of the whole image. To make this dataset more challenging, we randomized the scale and orientation of objects for some experiments. This dataset is typically used to .levaluate the performance of object categorization.

2.2.2 Weizmann Horse Dataset

The Weizmann Horse dataset [18] was designed for the evaluation of object segmentation. Recently, this dataset has also been used for testing object parsing. There are 328 horse images with cluttered background, shape variations, textured patterns, and changes in viewing angles. But in most images, the object appears in the center of the image and the background is relative small. These datasets are designed to evaluate

segmentation, so the groundtruth only gives the regions of the object and the background. To supplement this groundtruth, we required students to manually parse the images by locating the positions of object parts (e.g. head, leg, tail, etc.) in the images.

2.2.3 Multi-view Face Alignment Dataset

We use the dataset [19] for testing algorithm on multi-view face alignment. This dataset contains 280 image with ground truth of standard 65 key points which lie along the boundaries of face components with semantic meaning, i.e, eyes, nose, mouth and cheek.

2.2.4 Berkeley Human Baseball Dataset

The Berkeley Human Baseball dataset [20] is designed for the task of human body parsing. To supplement this groundtruth, we required students to manually parse the images by locating the positions of the parts of human body. In the experiment of human body parsing, Srinivasan and Shi [2] only used 5 joint nodes (head-torso, torso-left thigh, torso-right thigh, left thigh-left lower leg, right thigh-right lower leg) per image. In our case, there are 27 nodes along the boundary of human body per image used to give more detailed parsing. Therefore, we also asked students to label the parts of the human body as ground truth (i.e. to identify different parts of the human). This dataset contains a large variance of poses of human body and the appearance of clothes changes a lot from image to image.

2.2.5 MSRC Dataset

We use a standard public dataset, the MSRC 21-class Image Dataset [21], to perform experimental evaluations for the image parsing. This dataset is designed to evaluate

scene labeling including both image segmentation and multi-class object recognition. The ground truth only gives the labeling of the image pixels. There are a total of 591 images.

Part II

Non-Recursive Representation and Learning

CHAPTER 3

Probabilistic Grammar-Markov Model

In this chapter, we address the issues of object representation and unsupervised learning, but abandon the spirit of recursion and composition temporarily. We introduce a Probabilistic Grammar-Markov Model (PGMM) which couples probabilistic context free grammars and Markov Random Fields. These PGMMs are generative models defined over attributed features and are used to detect and classify objects in natural images. PGMMs are designed so that they can perform rapid inference, parameter learning, and the more difficult task of structure induction. PGMMs can deal with unknown 2D pose (position, orientation, and scale) in both inference and learning, different appearances, or aspects, of the model. The PGMMs can be learnt in an unsupervised manner where the image can contain one of an unknown number of objects of different categories or even be pure background. We first study the weakly supervised case, where each image contains an example of the (single) object of interest, and then generalize to less supervised cases. The goal of this chapter is theoretical but, to provide proof of concept, we demonstrate results from this approach on a subset of the Caltech dataset (learning on a training set and evaluating on a testing set). Our results are generally comparable with the current state of the art, and our inference is performed in less than five seconds. PGMMs and structure induction learning methods discussed in this chapter are out first attempt to the object modeling and learning. Their limitations of non-recursive design and greedy learning will motivate our later work.

3.1 Introduction

Remarkable progress in mathematics and computer science of probability is leading to a revolution in the scope of probabilistic models. There are exciting new probability models defined on structured relational systems, such as graphs or grammars [22, 23, 24, 25, 26, 27]. Unlike more traditional models, such as Markov Random Fields (MRF's) [28] and Conditional Random Fields (CRF's) [23], these models are not restricted to having fixed graph structures. Their ability to deal with varying graph structure means that they can be applied to model a large range of complicated phenomena as has been shown by their applications to natural languages [29], machine learning [27], and computer vision [30].

Our longterm goal is to provide a theoretical framework for the unsupervised learning of probabilistic models for generating, and interpreting, natural images [30]. This is somewhat analogous to Klein and Manning's work on unsupervised learning of natural language grammars [24]. In particular, we hope that this chaper can help bridge the gap between computer vision and related work on grammars in machine learning [29],[22],[27]. There are, however, major differences between vision and natural language processing. Firstly, images are arguably far more complex than sentences so learning a probabilistic model to generate natural images is too ambitious to start with. Secondly, even if we restrict ourselves to the simpler task of generating an image containing a single object we must deal with: (i) the cluttered background (similar to learning a natural language grammar when the input contains random symbols as well as words), (ii) the unknown 2D pose (size, scale, and position) of the object, and (iii) different appearances, or *aspects*, of the object (these aspect deal with changes due to different 3D poses of the object, different photometric appearance, different 2D shapes, or combinations of these factors). Thirdly, the input is a set of image intensities and is considerably more complicated than the limited types of speech tags (e.g.

nouns, verbs, etc) used as input in [24].

In this chapter, we address an important subproblem. We are given a set of images containing one of an unknown number of objects (with variable 2D pose) of different categories, or even pure background. The object is allowed to have several different appearances, or aspects. We call this *unsupervised learning* by contrast to *weakly supervised learning* where each image contains an example of a single object (but the position and boundary of the object are unknown). We represent these images in terms of attributed features (AF's). The task is to learn a probabilistic model for generating the AF's (both those of the object and the background). We require that the probability model must allow: (i) rapid inference (i.e. interpret each image), (ii) rapid parameter learning, and (iii) *structure induction*, where the structure of the model is unknown and must be grown in response to the data.

To address this subproblem, we develop a Probabilistic Grammar Markov Model (PGMM) which is motivated by this goal and its requirements. The PGMM combines elements of MRF's [28] and probabilistic context free grammars (PCFG's) [29]. The requirement that we can deal with a variable number of AF's (e.g. caused by different aspects of the object) motivates the use of grammars (instead of fixed graph models like MRF's). But PCFG's, see figure (3.1), are inappropriate because they make independence assumptions on the production rules and hence must be supplemented by MRF's to model the spatial relationships between AF's of the object. The requirement that we deal with 2D pose (both for learning and inference) motivates the use of oriented triangles of AF's as our basic building blocks for the probabilistic model, see figure (3.2). These oriented triangles are represented by features, such as the internal angles of the triangle, which are invariant to the 2D pose of the object in the image. The requirement that we can perform rapid inference on new images is achieved by combining the triangle building blocks to enable dynamic programming. The ability

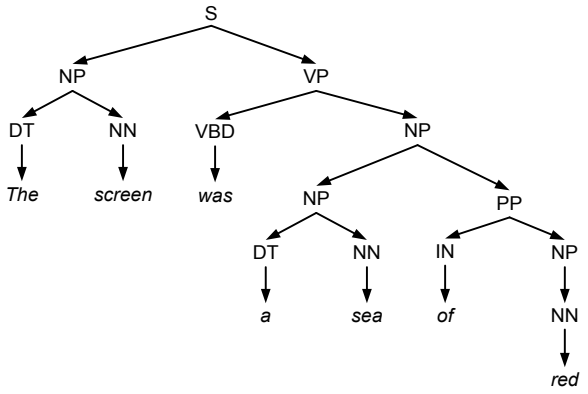


Figure 3.1: Probabilistic Context Free Grammar. The grammar applies production rules with probability to generate a tree structure. Different random sampling will generate different tree structures. The production rules are applied independently on different branches of the tree. There are no sideways relations between nodes.

to perform rapid inference ensures that parameter estimation and structure learning is practical.

We decompose the learning task into: (a) learning the structure of the model, and (b) learning the parameters of the model. Structure learning is the more challenging task [29],[22],[27] and we propose a structure induction (or structure pursuit) strategy which proceeds by building an AND-OR graph [25, 26] in an iterative way by adding more triangles or OR-nodes (for different aspects) to the model. We use clustering techniques to make proposals for adding triangles/OR-nodes and validate or reject these proposals by model selection. The clustering techniques relate to Barlow's idea of suspicious coincidences [31].

We evaluate our approach by testing it on parts of the Caltech-4 (faces, motorbikes, airplanes and background) [32] and Caltech-101 database [17]. Performance on this database has been much studied [32, 33, 34, 35, 36]. But we stress that the goal here is to develop a novel theory and test it, rather than simply trying to get better

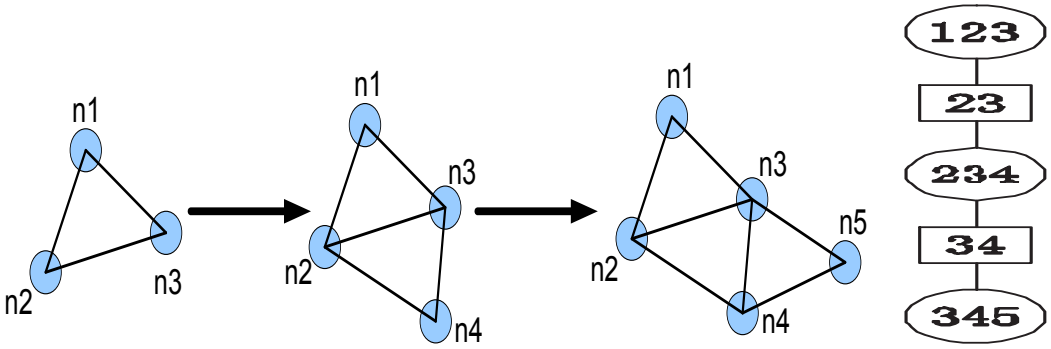


Figure 3.2: This chapter uses triplets of nodes as building blocks. We can grow the structure by adding new triangles. The junction tree (the far right panel) is used to represent the combination of triplets to allow efficient inference.

performance on a standard database. Nevertheless our experiments show three major results. Firstly, we can learn PGMMs for a number of different objects and obtain performance results close to the state of the art. Moreover, we can also obtain good localization results (which is not always possible with other methods). The speed of inference is under five seconds. Secondly, we demonstrate our ability to do learning and inference independent of the scale and orientation of the object (we do this by artificially scaling and rotating images from Caltech 101, lacking a standard database where these variations occur naturally). Thirdly, the approach is able to learn from noisy data (where half of the training data is only background images) and to deal with object classes, which we illustrate by learning a hybrid class consisting of faces, motorbikes and airplane.

This chapter is organized as follows. We first review the background in section (3.2). Section (3.3) describes the features we use to represent the images. In section (3.4) we give an overview of PGMMs. Section (3.5) specifies the probability distributions defined over the PGMM. In section (3.6), we describe the algorithms for inference, parameter learning, and structure learning. Section (3.7) illustrates our approach by learning models for 38 objects, demonstrating invariance to scale and

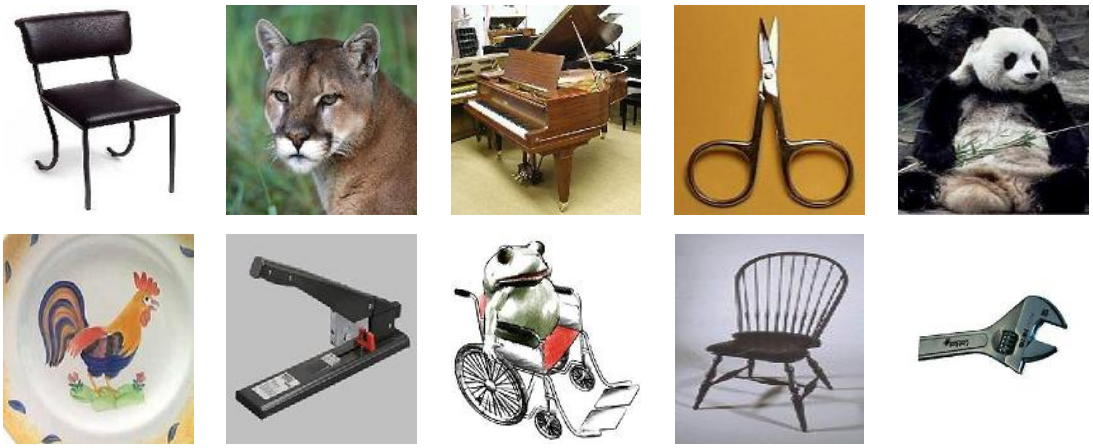


Figure 3.3: Ten of the object categories from Caltech 101 which we learn in this chapter.

rotation, and performing learning for object classes.

3.2 Background

This section gives a brief review of the background in machine learning and computer vision.

Structured models define a probability distribution on structured relational systems such as graphs or grammars. This includes many standard models of probability distributions defined on graphs – for example, graphs with fixed structure, such as MRF's [28] or Conditional Random Fields [23], or Probabilistic Context Free Grammars (PCFG's) [29] where the graph structure is variable. Attempts have been made to unify these approaches under a common formulation. For example, Case-Factor Diagrams [22] have recently been proposed as a framework which subsumes both MRF's and PCFG's. In this chapter, we will be concerned with models that combine probabilistic grammars with MRF's. The grammars are based on AND-OR graphs [22, 25, 26], which relate to mixtures of trees [37]. This merging of MRF's with prob-

abilistic grammars results in structured models which have the advantages of variable graph structure (e.g. from PCFG's) combined with the rich spatial structure from the MRF's.

There has been considerable interest in inference algorithms for these structured models, for example McAllester et al. [22] describe how dynamic programming algorithms (e.g. Viterbi and inside-outside) can be used to rapidly compute properties of interest for Case-Factor diagrams. But inference on arbitrary models combining PCFG's and MRF's remains difficult.

The task of learning, and particularly structure induction, is considerably harder than inference. For MRF models, the number of graph nodes is fixed and structure induction consists of determining the connections between the nodes and the corresponding potentials. For these graphs, an effective strategy is feature induction [38] which is also known as feature pursuit [39]. A similar strategy is also used to learn CRF's [40] where the learning is fully supervised. For Bayesian network, there is work on learning the structure using the EM algorithm [41].

Learning the structure of grammars in an unsupervised way is more difficult. Klein and Manning [24] have developed unsupervised learning of PCFG's for parsing natural language, but here the structure of grammar is specified. Zettlemoyer and Collins [27] perform similar work based on lexical learning with lambda-calculus language.

To our knowledge, there is no unsupervised learning algorithm for structure induction for any PGMM. But an extremely compressed version of part of our work appeared in [8].

There has been a considerable amount of work for learning MRF models for visual tasks such as object detection. An early attempt was described in [42]. The constellation model [32] is a nice example of a weakly supervised algorithm which represents objects by a fully connected (fixed) graph. Huttenlocher and collaborators [35, 34]

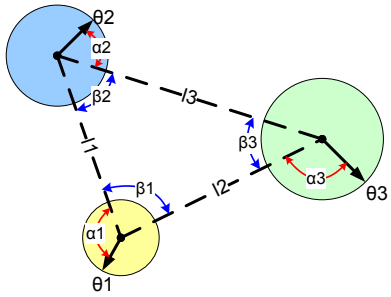


Figure 3.4: The oriented triplet is specified by the internal angles β, the orientation of the vertices θ, and the relative angles α between them.

explore different simpler MRF structures, such as k-fans models, which enable rapid inference.

There is also a large literature [32, 33, 34, 35, 36] on computer vision models for performing object recognition many of which have been evaluated on the Caltech databases [32]. Indeed, there is a whole range of computer vision methods which have been evaluated on the Caltech database [17]. A review of performance and critiques of the database are given in [43]. A major concern is that the nature of this dataset enables over-generalization, for example, the models can use features that occur in the background of the image and not within the object.

3.3 The Image Representation: Features and Oriented Triplets

In this chapter we will represent images in terms of isolated attributed features, which will be described in section (3.3.1). A key ingredient of our approach is to use conjunctions of features and, in particular, triplets of features with associated angles at the vertices which we call oriented triplets, see figures (3.4,3.5). The advantages of using conjunctions of basic features is well-known in natural language processing and leads to unigram, bigram, and trigram features [29].

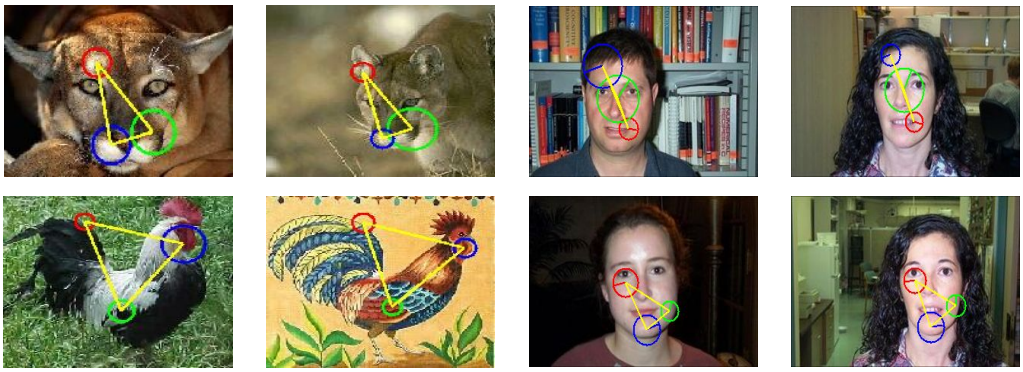


Figure 3.5: This figure illustrates the features and triplets without orientation (left two panels) and oriented triplets (next two panels).

There are several reasons for using oriented triplets in this chapter. Firstly, they contain geometrical properties which are invariant to the scale and rotation of the triplet. These properties include the angles between the vertices and the relative angles at the vertices, see figures (3.4,3.5). These properties can be used both for learning and inference of a PGMM when the scale and rotation are unknown. Secondly, they lead to a representation which is well suited to dynamic programming, similar to the junction tree algorithm [44], which enables rapid inference, see figures (3.6,3.2). Thirdly, they are well suited to the task of structure pursuit since we can combine two oriented triplets by a common edge to form a more complex model, see figures (3.2,3.6).

3.3.1 The Image Features

We represent an image by attributed features $\{x_i : i = 1, .., N_\tau\}$, where N_τ is the number of features in image I_τ with $\tau \in \Lambda$, where Λ is the set of images. Each feature is represented by a triple $x_i = (z_i, \theta_i, A_i)$, where z_i is the location of the feature in the image, θ_i is the orientation of the feature, and A_i is an appearance vector.

These features are computed as follows. We apply the Kadir-Brady [45] operator K_b to select circular regions $\{C^i(I_\tau) : i = 1, ..., N_\tau\}$ of the input image I_τ such that

$K_b(C^i(I_\tau)) > T, \forall\, i$, where T is a fixed threshold. We scale these regions to a constant size to obtain a set of scaled regions $\{\hat{C}^i(I_\tau) : i = 1, ..., N_\tau\}$. Then we apply the SIFT operator $L(.)$ [46] to obtain Lowe's feature descriptor $L_i = L(\hat{C}^i(I_\tau))$ together with an orientation θ_i (also computed by [46]) and set the feature position z_i to be the center of the window C^i. Then we perform PCA on the appearance attributes (using the data from all images $\{I_\tau : \tau \in \Lambda\}$) to obtain a 15 dimensional subspace (a reduction from 128 dimensions). Projecting L_i into this subspace gives us the appearance attribute A_i.

The motivation for using these operators is as follows. Firstly, the Kadir-Brady operator is an interest operator which selects the parts of the image which contain interesting features (e.g. edges, triple points, and textured structures). Secondly, the Kadir-Brady operator adapts geometrically to the size of the feature, and hence is scale-invariant. Thirdly, the SIFT operator is also (approximately) invariant to a range of photometric and geometric transformations of the feature. In summary, the features occur at interesting points in the image and are robust to photometric and geometric transformations.

3.3.2 The Oriented Triplets

An oriented triplet of three feature points has geometry specified by $(z_i, \theta_i, z_j, \theta_j, z_k, \theta_k)$ and is illustrated in figures (3.4,3.5). We construct a 15 dimensional invariant triplet vector $\vec{l}$ which is invariant to the scale and rotation of the oriented triplet.

$$\begin{aligned}\vec{l}(z_i, \theta_i, z_j, \theta_j, z_k, \theta_k) = (&l_1/L, l_2/L, l_3/L,\\ &\cos\alpha_1, \sin\alpha_1, \cos\alpha_2, \sin\alpha_2, \cos\alpha_3, \sin\alpha_3,\\ &\cos\beta_1, \sin\beta_1, \cos\beta_2, \sin\beta_2, \cos\beta_3, \sin\beta_3),\end{aligned} \tag{3.1}$$

where l_1, l_2, l_3 are the length of the three edges, $L = l_1 + l_2 + l_3$, $\alpha_1, \alpha_2, \alpha_3$ are the relative angles between the orientations $\theta_i, \theta_j, \theta_k$ and the orientations of the three edges

of the triangle, and $\beta_1, \beta_2, \beta_3$ are the angles between edges of the triangle (hence $\beta_1 + \beta_2 + \beta_3 = \pi$).

This representation is over-complete. But we found empirically that it was more stable than lower-dimensional representations. If rotation and scale invariance are not needed, then we can use alternative representations of triplets such as $(l_1, l_2, l_3, \theta_1, \theta_2, \theta_3, \beta_1, \beta_2, \beta_3)$. Previous authors [47, 48] have used triples of features but, to our knowledge, oriented triplets are novel.

3.4 Probabilistic Grammar-Markov Model

We now give an overview of the Probabilistic Grammar-Markov Model (PGMM), which has characteristics of both a probabilistic grammar, such as a Probabilistic Context Free Grammar (PCFG), and a Markov Random Field (MRF). The probabilistic grammar component of the PGMM specifies different topological structures, as illustrated in the five leftmost panels of figure (3.6), enabling the ability to deal with variable number of attributed features. The MRF component specifies spatial relationships and is indicated by the horizontal connections.

Formally we represent a PGMM by a graph $G = (V, E)$ where V and E denote the set of vertices and edges respectively. The vertex set V contains three types of nodes, "OR" nodes, "AND" nodes and "LEAF" nodes which are depicted in figure (3.6) by triangles, rectangles and circles respectively. The edge set E contains vertical edges defining the topological structure and horizontal edges defining spatial constraints (e.g. MRF's).

The leaf nodes are indexed by a and will correspond to AF's in the image. They have attributes (z_a, θ_a, A_a), where z_a denotes the spatial position, θ_a the orientation, and A_a the appearance. There is also a binary-valued *observability variable* u_a which

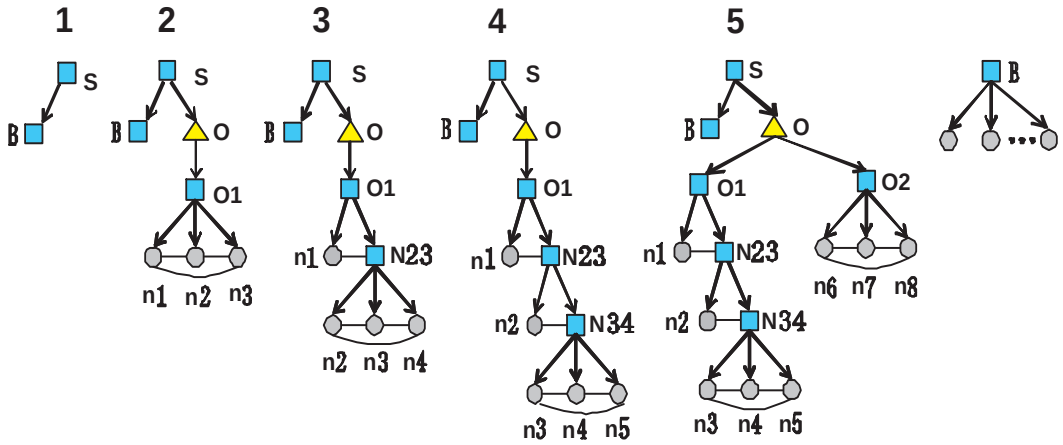


Figure 3.6: Graphical Models. Squares, triangles, and circles indicate AND, OR, and LEAF nodes respectively. The horizontal lines denote MRF connections. The far right panel shows the background node generating leaf nodes. The models for $O1$ for panels 2,3 and 4 correspond to the triplets combinations in figure (3.2). See text for notation.

indicates whether the node is observable in the image (a node may be unobserved because it is occluded, or the feature detector has too high a threshold). We set y to be the parse structure of the graph when the OR nodes take specific assignments. We decompose the set of leaves $L(y) = L_B(y) \bigcup L_O(y)$, where $L_B(y)$ are the leaves due to the background model, see the far right panel of figure (3.6), and $L_O(y)$ are the leaves due to the object. We order the nodes in $L_O(y)$ by "drop-out", so that the closer the node to the root the lower its number, see figure (3.6).

In this chapter, the only OR node is the object category node O. This corresponds to different aspects of the object. The remaining non-terminal nodes are AND nodes. They include a background node B, object aspect nodes O_i and *clique nodes* of form $N_{a,a+1}$ (containing points n_a, n_{a+1}). Each aspect O_i corresponds to a set of object leaf nodes $L_O(y)$ with corresponding cliques $C(L_O(y))$. As shown in figure (3.6), each clique node $N_{a,a+1}$ is associated with a leaf node n_{a+2} to form a *triplet-clique*

$C_a\{n_a, n_{a+1}, n_{a+2}\}$.

The directed (vertical) edges connect nodes at successive levels of the tree. They connect: (a) the root node S to the object node and the background node, (b) the object node to aspect nodes, (c) a non-terminal node to three leaf nodes, see panel (ii) of figure (3.6), or (d) a non-terminal node to a clique node and a leaf node, see panel (iii) of figure (3.6). In case (c) and (d), they correspond to a triplet-clique of point features.

Figure (3.6) shows examples of PGMMs. The top rectangle node S is an AND node. The simplest case is a pure background model, in panel (1), where S has a single child node B which has an arbitrary number of leaf nodes corresponding to feature points. In the next model, panel (2), S has two child nodes representing the background B and the object category O. The category node O is an OR node which is represented by a triangle. The object category node O has child node, O_1, which has a triplet of child nodes corresponding to point features. The horizontal line indicates spatial relations of this triplet. The next two models, panels (3) and (4), introduce new feature points and new triplets. We can also introduce a new aspect of the object O_2, see panel (5), to allow for the object to have a different appearance.

3.5 The Distribution defined on the PGMM

The structure of the PGMM is specified by figure (3.7). The PGMM specifies the probability distribution of the AF's observed in an image in terms of parse graph y and model parameters Ω, ω for the grammar and the MRF respectively. The distribution involves additional hidden variables which include the pose G and the observability variables $u = \{u_a\}$. We set $z = \{z_a\}$, $A = \{A_a\}$, and $\theta = \{\theta_a\}$. See table (3.1) for the notation used in the model.

Table 3.1: The notations used for the PGMM.

Notation	Meaning
Λ	the set of images
$x_i = (z_i, \theta_i, A_i)$	an attributed feature (AF)
$\{x_i : i = 1, ..., N_\tau\}$	attributed features of image I_τ
N_τ	the number of features in image I_τ
z_i	the location of the feature
θ_i	the orientation of the feature
A_i	the appearance vector of the feature
y	the topological structure
a	the index of the node
n_a	the leaf nodes of PGMM
$C_a = \{n_a, n_{a+1}, n_{a+2}\}$	a triplet clique
$\vec{l}_C()$	the invariance triplet vector of clique C
$u = \{u_a\}$	observability variables
Ω	the parameters of grammatical part
ω	(ω^g, ω^A)
ω^g	the parameters of spatial relation of leaf nodes
ω^A	the parameters of appearances of the AF's
$V = \{i(a)\}$	the correspondence variables

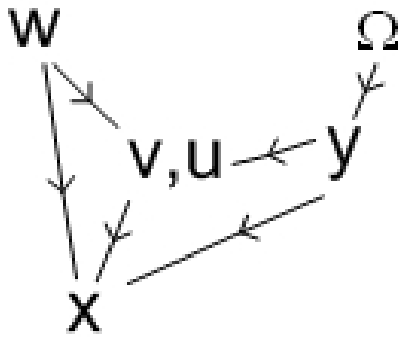


Figure 3.7: This figure illustrates the dependencies between the variables. The variables Ω specify the probability for topological structure y. The spatial assignments z of the leaf nodes are influenced by the topological structure y and the MRF variables ω. The probability distribution for the image features x depends on y, ω and z.

We define the full distribution to be:

$$P(u, z, A, \theta, y, \omega, \Omega) = P(A|y, \omega^A)P(z, \theta|y, \omega^g)P(u|y, \omega^g)P(y|\Omega)P(\omega)P(\Omega). \tag{3.2}$$

The observed AF's are those for which $u_a = 1$. Hence the observed image features $x = \{(z_a, A_a, \theta_a) : \ s.t. \ u_a = 1\}$. We can compute the joint distribution over the observed image features x by:

$$P(x, y, \omega, \Omega) = \sum_{\{(z_a, A_a, \theta_a) \ s.t. u_a = 0\}} P(u, z, A, \theta, y, \omega, \Omega). \tag{3.3}$$

We now briefly explain the different terms in equation (3.2) and refer to the following subsections for details.

$P(y|\Omega)$ is the grammatical part of the PGMM (with prior $P(\Omega)$). It generates the topological structure y which specifies which aspect model O_i is used and the number of background nodes. The term $P(u|y, \omega^g)$ specifies the probability that the leaf nodes are observed (background nodes are always observed). $P(z, \theta|\omega^g)$ specifies the prob-

ability of the spatial positions and orientations of the leaf nodes. The distributions on the object leaf nodes are specified in terms of the invariant shape vectors defined on the triplet cliques, while the background leaf nodes are generated independently. Finally, the distribution $P(A|y, \omega^A)$ generates the appearances of the AF's. $P(\omega^g, \omega^A)$ is the prior on ω.

3.5.1 Generating the leaf nodes: $P(y|\Omega)$

This distribution $P(y|\Omega)$ specifies the probability distribution of the leaf nodes. It determines how many AF's are present in the image (except for those which are unobserved due to occlusion or falling below threshold). The output of y is the set of numbered leaf nodes. The numbering determines the object nodes $L_O(y)$ (and the aspects of the object) and the background nodes $L_B(O)$. (The attributes of the leaf nodes are determined in later sections).

$P(y|\Omega)$ is specified by a set of production rules. In principle, these production rules can take any form such as those used in PCFG's [29]. Other possibilities are Dechter's And-Or graphs [25], case-factor diagrams [22], composite templates [26], and compositional structures [49]. In this chapter, however, we restrict our implementation to rules of form:

$$
\begin{aligned}
S &\rightarrow \{B, O\} \; with \; prob \; 1, \\
O &\rightarrow \{O_j : j = 1, ..., \rho\} \; with \; prob, \Omega_j^O, \; j = 1, ..., \rho \\
O_j &\rightarrow \{n_a, N_{a+1,a+2}\} \; with \; prob. \; 1, \; a = \beta_j, \\
N_{a,a+1} &\rightarrow \{n_a, N_{a+1,a+2}\} \; with \; prob. \; 1, \; \beta_i + 1 \leq a \leq \beta_{j+1} - 4. \\
N_{\beta_{j+1}-3,\beta_{j+1}-2} &\rightarrow \{n_{\beta_{j+1}-2}, n_{\beta_{j+1}-1}\} \; with \; prob \; 1, \\
B &\rightarrow \{n_{\beta_{\rho+1}}, ..., n_{\beta_{\rho+1}+m}\} \; with \; prob \; \Omega^B e^{-m\Omega^B} \; (m = 0, 1, 2...).
\end{aligned} \tag{3.4}
$$

Here $\beta_1 = 1$. The nodes $\beta_j, ..., \beta_{j+1} - 1$ correspond to aspect O_j. Note that

these $\{\beta_j\}$ are parameters of the model which will be learnt. ρ is the number of aspects and $\{\Omega_j^O\}$ and Ω^B are parameters that specify the distribution (all these will be learnt). We write $\Omega = \{\Omega^B, \Omega_1^O, ..., \Omega_\rho^O, \beta_1, ..., \beta_{\rho+1}, \rho\}$. These rules are illustrated in figure (3.6) (note that, for simplicity of the figure, we represent the combination $N_{a,a+1} \mapsto \{n_a, N_{a+1,a+2}\}$ *and* $N_{a+1,a+2}$ by $N_{a,a+1} \mapsto (n_a, n_{a+1}, n_{a+2})$).

3.5.2 Generating the observable leaf nodes: $P(u|y, \omega^g)$

The distribution $P(u|y, \omega^g)$ specifies whether objects leafs are observable in the image (all background nodes are assumed to be observed). The observation variable u allows for the possibility that an object leaf node a is unobserved due to occlusion or because the feature detector response falls below threshold. Formally, $u_a = 1$ if the object leaf node a is observed and $u_a = 0$ otherwise. We assume that the observability of nodes are independent:

$$P(u|y, \omega^g) = \prod_{a \in L_O(y)} \lambda_\omega^{u_a} (1 - \lambda_\omega)^{(1-u_a)} = \exp\left\{ \sum_{a \in L_O(y)} \{\delta_{u_a,1} \log \lambda_\omega + \delta_{u_a,0} \log(1 - \lambda_\omega)\} \right\}, \tag{3.5}$$

where λ_ω is the parameter of the bernoulli distribution and $\delta_{u_a,1}$ are the Kronecker delta function (i.e. $\delta_{u_a,1} = 0$ unless $u_a = 1$).

3.5.3 Generating the positions and orientation of the leaf nodes: $P(z, \theta|y, \omega^g)$.

$P(z, \theta|y, \omega^g)$ is the distribution of the spatial positions z and orientations θ of the leaf nodes. We assume that the spatial positions and orientations of the background leaf nodes are independently generated from a uniform probability distribution.

The distribution on the position and orientations of the object leaf nodes is required to satisfy two properties: (a) it is invariant to the 2D pose (position, orientation, and

scale), and (b) it is easily computable. In order to satisfy both these properties we make an approximation. We first present the distribution that we use and then explain its derivation and the approximation involved.

The distribution is given by:

$$P(z,\theta|y,\omega^g) = K \times P(l(z,\theta)|y,\omega^g), \tag{3.6}$$

where $P(l(z,\theta)|y,\omega^g)$ (see equation (3.7)) is a distribution over the invariant shape vectors l computed from the spatial positions z and orientations θ. We assume that K is a constant. This is an approximation because the full derivation, see below, has $K(z,\theta)$.

We define the distribution $P(z,\theta|y,\omega^g)$ over l to be a Gaussian distribution defined on the cliques:

$$P(l|y,\omega^g) = \frac{1}{Z}\exp\left\{\sum_{a\in Cliques(y)} \psi_a(\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2}),\omega_a^g)\right\}, \tag{3.7}$$

where the triplet cliques are $C_1,...,C_{\tau-2}$, where $C_a = (n_a, n_{a+1}, n_{a+2})$. The invariant triplet vector $\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2})$ is given by equation (3.1).

The potential $\psi_a(\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2}),\omega_a^g)$ specifies geometric regularities of clique C_a which are invariant to the scale and rotation. They are of form:

$$\begin{aligned}\psi_a(\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2}),\omega_a^g) = \\ -(1/2)(\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2}) - \vec{\mu}_a^z)^T(\Sigma_a^z)^{-1} \\ (\vec{l}(z_a,\theta_a,z_{a+1},\theta_{a+1},z_{a+2},\theta_{a+2}) - \vec{\mu}_a^z).\end{aligned} \tag{3.8}$$

where $\omega_a^g = (\mu_a^z, \Sigma_a^z)$ and $\omega^g = \{\omega_a^g\}$.

Now we derive equation (3.6) for $P(z,\theta|y,\omega^g)$ and explain the nature of the approximation. First, we introduce a pose variable G which specifies the position, orien-

tation, and scale of the object. We set:

$$P(z, \theta, \vec{l}, G|y, \omega^g) = P(z, \theta|l, G)P(l|y, \omega^g)P(G), \tag{3.9}$$

where the distribution $P(z, \theta|l, G)$ is of form:

$$P(z, \theta|l, G) = \delta(z - z(G, l))\delta(\theta - \theta(G, l)). \tag{3.10}$$

$P(z, \theta|l, G)$ specifies the positions and orientations z, θ by deterministic functions $z(l, G), \theta(l, G)$ of the pose G and shape invariant vectors l. We can invert this function to compute $l(z, \theta)$ and $G(z, \theta)$ (i.e. to compute the invariant feature vectors and the pose from the spatial positions and orientations z, θ).

We obtain $P(z, \theta|y, \omega^g)$ by integrating out l, G:

$$P(z, \theta|y, \omega^g) = \int dG \int dl P(z, \theta, l, G|y, \omega^g). \tag{3.11}$$

Substituting equations (3.10) and (3.9) into equation (3.11) yields:

$$\begin{aligned} P(z, \theta|y, \omega^z) &= \int dG \int dl \delta(z - z(l, G))\delta(\theta - \theta(l, G))P(l|y, \omega^g)P(G) \\ &= \int \int d\rho d\gamma \frac{\partial(l, G)}{\partial(\rho, \gamma)} \delta(z - \rho)\delta(\theta - \gamma)P(l(z, \theta)|y, \omega^g)P(G(z, \theta)), \\ &= \frac{\partial(l, G)}{\partial(\rho, \gamma)}(z, \theta)P(l(z, \theta)|y, \omega^g)P(G(z, \theta)), \end{aligned} \tag{3.12}$$

where we performed a change of integration from variables (l, G) to new variables (ρ, γ) with $\rho = z(l, G)$, $\gamma = \theta(l, G)$ and where $\frac{\partial(l,G)}{\partial(\rho,\gamma)}(z, \theta)$ is the Jacobian of this transformation (evaluated at (z, θ)).

To obtain the form in equation (3.6) we simply equation (3.12) by assuming that $P(G)$ is the uniform distribution and by making the approximation that the Jacobian factor is independent of (z, θ) (this approximation will be valid provided the size and shapes of the triplets do not vary too much).

3.5.4 The Appearance Distribution $P(A|y, \omega^A)$.

We now specify the distribution of the appearances $P(A|y, \omega^A)$. The appearances of the background nodes are generated from a uniform distribution. For the object nodes, the appearance A_a is generated by a Gaussian distribution specified by $\omega_a^A = (\mu_a^A, \Sigma_a^A)$:

$$P(A_a|\omega_a^A) = \frac{1}{\sqrt{2\pi}|\Sigma_{A,a}|} \exp\{-(1/2)(A_a - \mu_a^A)^T(\Sigma_a^A)^{-1}(A_a - \mu_a^A)\}. \quad (3.13)$$

3.5.5 The Priors: $P(\Omega), P(\omega^A), P(\omega^g)$.

The prior probabilities are set to be uniform distributions, expect for the priors on the appearance covariances Σ_a^A which are set to zero mean Gaussians with fixed variance.

3.5.6 The Correspondence Problem:

Our formulation of the probability distributions has assumed an ordered list of nodes indexed by a. But these indices are specified by the model and cannot be observed from the image. Indeed performing inference requires us to solve a correspondence problem between the AF's in the image and those in the model. This correspondence problem is complicated because we do not know the aspect of the object and some of the AF's of the model may be unobservable.

We formulate the correspondence problem by defining a new variable $V = \{i(a)\}$. For each $a \in L_O(y)$, the variable $i(a) \in \{0, 1, ..., N_\tau\}$, where $i(a) = 0$ indicates that a is unobservable (i.e. $u_a = 0$). For background leaf nodes, $i(a) \in \{1, ..., N_\tau\}$. We constrain all image nodes to be matched so that $\forall j \in \{1, ..., N_\tau\}$ there exists a unique $b \in L(y)$ $s.t.$ $i(b) = j$ (we create as many background nodes as is necessary to ensure this). To ensure uniqueness, we require that object triplet nodes all have unique matches in the image (or are unmatched) and that background nodes can only match AF's which are not matched to object nodes or to other background nodes. (It

is theoretically possible that object nodes from different triplets might match the same image AF. But this is extremely unlikely due to the distribution on the object model and we have never observed it).

Using this new notation, we can drop the u variable in equation (3.5) and replace it by V with prior:

$$P(V|y,\omega^g) = \frac{1}{\hat{Z}} \prod_a \exp\{-\log\{\lambda_\omega/(1-\lambda_\omega)\}\delta_{i(a),0}\} \tag{3.14}$$

This gives the full distribution (see equation (3.2) which is defined over u variable):

$$P(\{z_i, A_i, \theta_i\}|V, y, \omega^g, \omega^A, \Omega)P(V|y,\omega^g)P(y|\Omega)P(\omega)P(\Omega), \tag{3.15}$$

with

$$\begin{aligned} &P(\{z_i, A_i, \theta_i\}|V, y, \omega^g, \omega^A, \Omega) \\ &= \frac{1}{Z} \prod_{a \in L_O(y): i(a) \neq 0} P(A_{i(a)}|y, \omega^A, V) \prod_{c \in C(L_O(y))} P(\vec{l_c}(\{z_{i(a)}, \theta_{i(a)}\})|y, \omega^g, V). \end{aligned} \tag{3.16}$$

We have the constraint that $|L_B(y)| + \sum_{a \in L_O(y)}(1 - \delta_{i(a),0}) = N_\tau$. Hence $P(y|\Omega)$ reduces to two components: (i) the probability of the aspect $P(L_O(y)|\Omega)$ and the probability $\Omega^B e^{-\Omega^B |L_B(y)|}$ of having $|L_B(y)|$ background nodes.

There is one problem with the formulation of equation (3.16). There are variables on the right hand side of the equation which are not observed – i.e. z_a, θ_a such that $i(a) = 0$. In principle, these variables should be removed from the equation by integrating them out. In practice, we replace their values by their best estimates from $P(\vec{l_c}(\{z_{i(a)}, \theta_{i(a)}\})|y, \omega^g)$ using our current assignments of the other variables. For example, suppose we have assigned two vertices of a triplet to two image AF's and decide to assign the third vertex to be unobserved. Then we estimate the position and orientation of the third vertex by the most probable value given the position and orientation assignments of the first two vertices and relevant clique potential. This is sub-optimal

but intuitive and efficient. (It does require that we have at least two vertices assigned in each triplet).

3.6 Learning and Inference of the model

In order to learn the models, we face three tasks: (I) structure learning, (II) parameter learning to estimate (Ω, ω), and (III) inference to estimate (y, V) (from a single image).

Inference requires estimating the parse tree y and the correspondences $V = \{i(a)\}$ from input x. The model parameters (Ω, ω) are fixed. This requires solving

$$\begin{aligned}(y^*, V^*) &= \arg\max_{y,V} P(y, V|x, \omega, \Omega) \\ &= \arg\max_{y,V} P(x, \omega, \Omega, y, V). \end{aligned} \tag{3.17}$$

As described in section (3.6.1) we use dynamic programming to estimate y^*, V^* efficiently.

Parameter learning occur when the structure of the model is known but we have to estimate the parameters of the model. Formally we specify a set W of parameters (ω, Ω) which we estimate by MAP. Hence we estimate

$$\begin{aligned}(\omega^*, \Omega^*) &= \arg\max_{\omega,\Omega \in W} P(\omega, \Omega|x) \propto P(x|\omega, \Omega)P(\omega, \Omega) \\ &= \arg\max_{\omega,\Omega \in W} P(\omega, \Omega) \prod_{\tau \in \Lambda} \sum_{y_\tau, V_\tau} P(x_\tau, y_\tau, V_\tau|\omega, \Omega). \end{aligned} \tag{3.18}$$

This is performed by an EM algorithm, see section (3.6.2), where the summation over the $\{V_\tau\}$ is performed by dynamic programming (the summation over the y's corresponds to summing over the different aspects of the object). The ω, Ω are calculated using sufficient statistics.

Structure Learning involves learning the model structure. Our strategy is to grow the structure of the PGMM by adding new aspect nodes, or by adding new cliques to

existing aspect nodes. We use clustering techniques to propose ways to grow the structure, see section (3.6.3). For each proposed structure, we have a set of parameters W which extends the set of parameters of the previous structure. For each new structure, we evaluate the fit to the data by computing the *score*:

$$score = \max_{\omega,\Omega} P(\omega, \Omega) \prod_{\tau \in \Lambda} \sum_{y_\tau} \sum_{V_\tau} P(x_\tau, y_\tau, V_\tau | \omega, \Omega). \tag{3.19}$$

We then apply standard model selection by using the score to determine if we should accept the proposed structure or not. Evaluating the score requires summing over the different aspects and correspondence $\{V_\tau\}$ for all the images. This is performed by using dynamic programming.

3.6.1 Dynamic Programming for the Max and Sum

Dynamic programming plays a core role for PGMMs. All three tasks – inference, parameter learning, and structure learning – require dynamic programming. Firstly, inference uses dynamic programming via the max rule to calculate the most probable parse tree y^*, V^* for input x. Secondly, in parameter learning, the E-step of the EM algorithm relies on dynamic programming to compute the sufficient statistics by the sum rule and take the expectations with respect to $\{y_\tau\}, \{V_\tau\}$. Thirdly, structure learning summing over all configurations $\{y_\tau\}, \{V_\tau\}$ uses dynamic programming as well.

The structure of a PGMM is designed to ensure that dynamic programming is practical. Dynamic programming was first used to detect objects in images by Coughlan et al. [50]. In this chapter, we use the ordered clique representation to use the configurations of triangles as the basic variables for dynamic programming similar to the junction tree algorithm [44].

We first describe the use of dynamic programming using the max rule for inference (i.e. determining the aspect and correspondence for a single image). Then we will

describe the modification to the sum rule used for parameter learning and structure pursuit.

To perform inference, we need to estimate the best aspect (object model) $L_O(y)$ and the best assignment V. We loop over all possible aspects and for each aspect we select the best assignment by dynamic programming (DP). For DP we keep a table of the possible assignments including the unobservable assignment. As mentioned above, we perform the sub-optimal method of replacing missing values $z_a, \theta_a \; s.t. \; i(a) = 0$ by their most probable estimates.

The conditional distribution is obtained from equations (3.4,3.7,3.13,3.14).

$$P(y, V, x|\omega, \Omega) = \frac{1}{Z} \exp\{ \sum_{a \in C(L_O(y))} \psi_a(\vec{l}(z_{i(a)}, \theta_{i(a)}, z_{i(a+1)}, \theta_{i(a+1)}, z_{i(a+2)}, \theta_{i(a+2)}), \omega_a^g) \\ -(1/2) \sum_{a \in L_O(y)} \{1 - \delta_{i(a),0}\} (A_{i(a)} - \mu_a^A)^T (\Sigma_a^A)^{-1} (A_{i(a)} - \mu_a^A) \\ - \sum_{a \in L_O(y)} \log\{\lambda_\omega/(1-\lambda_\omega)\} \delta_{i(a),0} - \Omega^B (N_\tau - |L_O(y)|) + \sum_{j \in [1,\rho]} I(\beta_j, L_O(y)) \log \Omega_j^O \}. \tag{3.20}$$

where $I(\beta_j, L_O(y))$ is an indicator which indicates the aspect j is active or not. $I(\beta_j, L_O(y))$ equals one if $\beta_j \in L_O(y)$, otherwise zero.

We can re-express this as

$$P(y, V, x|\omega, \Omega) = \prod_{a=1}^{|L_O|-2} \hat{\pi}_a[(z_{i(a)}, A_{i(a)}, \theta_{i(a)}), (z_{i(a+1)}, A_{i(a+1)}, \theta_{i(a+1)}), \\ (z_{i(a+2)}, A_{i(a+2)} \theta_{i(a+2)})], \tag{3.21}$$

where the $\hat{\pi}_a[.]$ are determined by equation (3.20).

We maximize equation (3.20) with respect to y and V. The choice of y is the choice of aspect (because the background nodes are determined by the constraint that all AF's in the image are matched). For each aspect, we use dynamic programming to

maximize over V. This can be done recursively by defining a function $h_a[(z_{i(a)}, A_{i(a)}, \theta_{i(a)}), (z_{i(a+1)}, A_{i(a+1)}, \theta_{i(a+1)})]$ by a forward pass:

$$h_{a+1}[(z_{i(a+1)}, A_{i(a+1)}, \theta_{i(a+1)}), (z_{i(a+2)}, A_{i(a+2)}, \theta_{i(a+2)})] =$$
$$\max_{i(a)} \hat{\pi}_a[(z_{i(a)}, A_{i(a)}, \theta_{i(a)}), (z_{i(a+1)}, A_{i(a+1)}, \theta_{i(a+1)}), (z_{i(a+2)}, A_{i(a+2)}, \theta_{i(a+2)})]$$
$$h_a[(z_{i(a)}, A_{i(a)}, \theta_{i(a)}), (z_{i(a+1)}, A_{i(a+1)}, \theta_{i(a+1)})] \quad (3.22)$$

The forward pass computes the maximum value of $P(y, V, x|\omega, \Omega)$. The backward pass of dynamic programming compute the most probable value V^*. The forward and backward passes are computed for all possible aspects of the model. As stated earlier in section (3.5.6), we make an approximation by replacing the values $z_{i(a)}, \theta_{i(a)}$ of unobserved object leaf nodes (i.e. $i(a) = 0$) by their most probable values.

We perform the max rule, equation (3.22), for each possible topological structure y. In this chapter, the number of topological structures is very small (i.e. less than twenty) for each object category and so it is possible to enumerate them all. The computational complexity of the dynamic programming algorithm is $O(MN^K)$ where M is the number of cliques in the aspect model for the object, $K = 3$ is the size of the maximum clique and N is the number of image features.

We will also use the dynamic programming algorithm (using the sum rule) to help perform parameter learning and structure learning. For parameter learning, we use the EM algorithm, see next subsection, which requires calculating sums over different correspondences and aspects. For structure learning we need to calculate the score, see equation (3.19), which also requires summing over different correspondences and aspects. This requires replacing the $\max$ in equation (3.22) by $\sum$. If points are unobserved, then we restrict the sum over their positions for computational reasons (summing over the positions close to their most likely positions).

3.6.2 EM Algorithm for Parameter Learning

To perform EM to estimate the parameters ω, Ω from the set of images $\{x_\tau : \tau \in \Lambda\}$. The criterion is to find the ω, Ω which maximize:

$$P(\omega, \Omega | \{x_\tau\}) = \sum_{\{y_\tau\}, \{V_\tau\}} P(\omega, \Omega, \{y_\tau\}, \{V_\tau\} | \{x_\tau\}), \tag{3.23}$$

where:

$$P(\omega, \Omega, \{y_\tau\}, \{V_\tau\} | \{x_\tau\}) = \frac{1}{Z} P(\omega, \Omega) \prod_{\tau \in \Lambda} P(y_\tau, V_\tau | x_\tau, \omega, \Omega). \tag{3.24}$$

This requires us to treat $\{y_\tau\}, \{V_\tau\}$ as missing variables that must be summed out during the EM algorithm. To do this we use the EM algorithm using the formulation described in [51]. This involves defining a free energy $F[q, \omega, \Omega]$ by:

$$\begin{aligned} F[q(.,.), \omega, \Omega] = & \sum_{\{y_\tau\}, \{V_\tau\}} q(\{y_\tau\}, \{V_\tau\}) \log q(\{y_\tau\}, \{V_\tau\}) \\ & - \sum_{\{y_\tau\}, \{V_\tau\}} q(\{y_\tau\}, \{V_\tau\}) \log P(\omega, \Omega, \{y_\tau\}, \{V_\tau\} | \{x_\tau\}), \end{aligned} \tag{3.25}$$

where $q(\{y_\tau\}, \{V_\tau\})$ is a normalized probability distribution. It can be shown [51] that minimizing $F[q(.,.), \omega, \Omega]$ with respect to $q(.,.)$ and (ω, Ω) in alternation is equivalent to the standard EM algorithm. This gives the E-step and the M-step:

E-step:

$$q^t(\{y_\tau\}, \{V_\tau\}) = P(\{y_\tau\}, \{V_\tau\} | \{x_\tau\}, \omega^t, \Omega^t), \tag{3.26}$$

M-step:

$$(\omega^{t+1}, \Omega^{t+1}) = \arg\min_{\omega, \Omega} \{ - \sum_{\{y_\tau\}, \{V_\tau\}} q^t(\{y_\tau\}, \{V_\tau\}) \log P(\omega, \Omega, \{y_\tau\}, \{V_\tau\} | \{x_\tau\}) \} \tag{3.27}$$

The distribution $q(\{y_\tau\}, \{V_\tau\}) = \prod_{\tau \in \Lambda} q_\tau(\{y_\tau\}, \{V_\tau\})$ because there is no dependence between the images. Hence the E-step reduces to:

$$q^t_\tau(\{y_\tau\}, \{V_\tau\}) = P(\{y_\tau\}, \{V_\tau\} | \{x_\tau\}, \omega^t, \Omega^t), \tag{3.28}$$

which is the distribution of the aspects and the correspondences using the current estimates of the parameters ω^t, Ω^t.

The M-step requires maximizing with respect to the parameters ω, Ω after summing over all possible configurations (aspects and correspondences). The summation can be performed using the sum version of dynamic programming, see equation (3.22). The maximization over parameters is straightforward because they are the coefficients of Gaussian distributions (mean and covariances) or exponential distributions. Hence the maximization can be done analytically.

For example, consider a simple exponential distribution $P(h|\alpha) = \frac{1}{Z(\alpha)} \exp\{f(\alpha)\phi(h)\}$, where h is the observable, α is the parameters, $f(.)$ and $\phi(.)$ are arbitrary functions and $Z(\alpha)$ is the normalization term. Then $\sum_h q(h) \log P(h|\alpha) = f(\alpha) \sum_h q(h)\phi(h) - \log Z(\alpha)$. Hence we have

$$\frac{\partial \sum_h q(h) \log P(h|\alpha)}{\partial \alpha} = \frac{\partial f(\alpha)}{\partial \alpha} \sum_h q(h)\phi(h) - \frac{\partial \log Z(\alpha)}{\partial \alpha}. \tag{3.29}$$

If the distributions are of simple forms, like the Gaussians used in our models, then the derivatives of $f(\alpha)$ and $\log Z(\alpha)$ are straightforward to compute and the equation can be solved analytically. The solution is of form:

$$\mu(t) = \sum_h q^t(h)h, \;\; \sigma^2(t) = \sum_h q^t(h)\{h - \mu(t)\}^2. \tag{3.30}$$

Finally, the EM algorithm is only guaranteed to converge to a local maxima of $P(\omega, \Omega|\{x_\tau\})$ and so a good choice of initial conditions is critical. The triplet vocabularies, described in subsection (3.6.3.1), give good initialization (so we do not need to use standard methods such as multiple initial starting points).

3.6.3 Structure Pursuit

Structure pursuit proceeds by adding a new triplet clique to the PGMM. This is done either by adding a new aspect node O_j and/or by adding a new clique node $N_{a,a+1}$.

This is illustrated in figure (3.6) where we grow the PGMM from panel (1) to panel (5) in a series of steps. For example, the steps from (1) to (2) and from (4) to (5) correspond to adding a new aspect node. The steps from (2) to (3) and from (3) to (4) correspond to adding new clique nodes. Adding new nodes requires adding new parameters to the model. Hence it corresponds to expanding the set W of non-zero parameters.

Our strategy for structure pursuit is as follows, see figures (3.8,3.9). We first use clustering algorithms to determine a triplet vocabulary. This triplet vocabulary is used to propose ways to grow the PGMM, which are evaluated by how well the modified PGMM fits the data. We select the PGMM with the best score, see equation (3.19). The use of these triplet vocabularies reduces the, potentially enormous, number of ways to expand the PGMM down to a practical number. We emphasize that the triplet vocabulary is only used to assist the structure learning and it does not appear in the final PGMM.

3.6.3.1 The appearance and triplet vocabularies

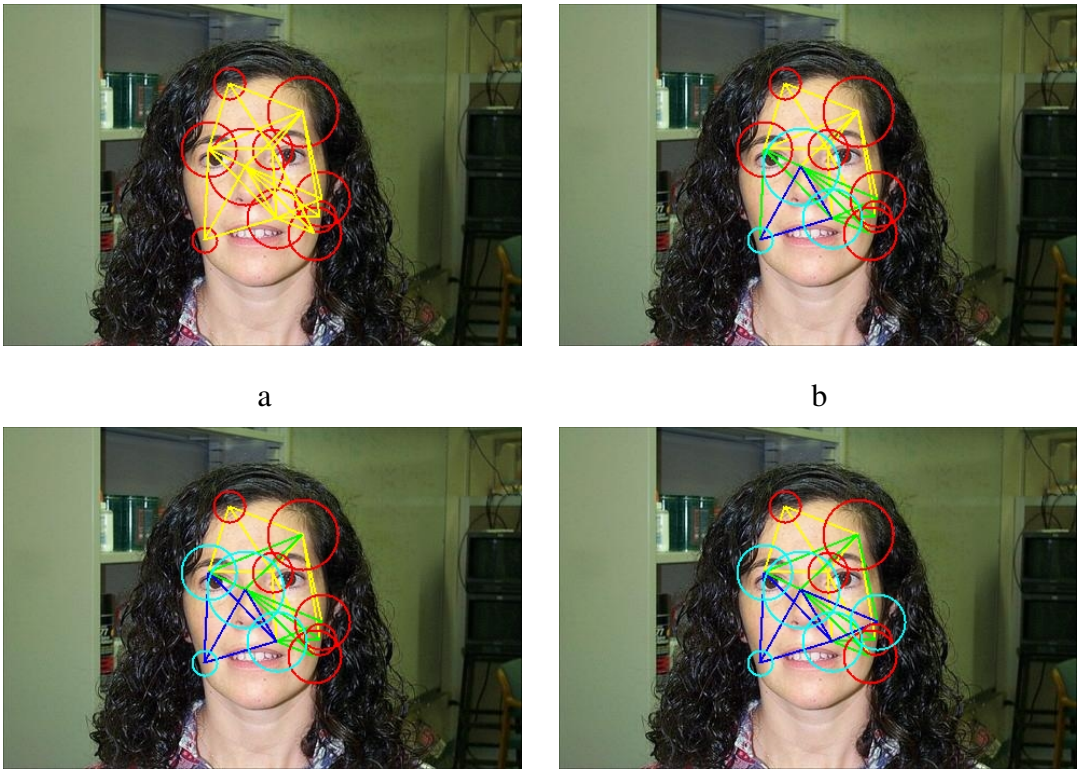
We construct appearance and triplet vocabularies using the features $\{x_i^\tau\}$ extracted from the image dataset as described in section (3.3.1).

To get the appearance vocabulary Voc_A, we perform k-means clustering on the appearances $\{A_i^\tau\}$ (ignoring the spatial positions and orientations $\{(z_i^\tau, \theta_i^\tau)\}$). The means $\mu^{A,a}$ and covariances $\Sigma^{A,a}$ of the clusters, define the appearance vocabulary:

$$Voc_A = \{(\mu^{A,a}, \Sigma^{A,a}) : a \in \Lambda_A\}. \tag{3.31}$$

where Λ_A is a set of indexes for the appearance ($|\Lambda_A|$ is given by the number of means).

To get the triplet vocabulary, we first quantize the appearance data $\{A_i^\tau\}$ to the means $\mu^{A,a}$ of the appearance vocabulary using nearest neighbor (with Euclidean dis-



a b

c d

Figure 3.8: This figure illustrates structure pursuit. a)image with triplets. b) one triplet induced. c) two triplets induced. d) three triplets induced. Yellow triplets: all triplets from triplet vocabulary. Blue triplets: structure induced. Green triplets: possible extensions for next induction. Circles with radius: image features with different sizes.

tance). This gives a set of modified data features $\{(z_i^\tau, \theta_i^\tau, \mu^{A,a(i,\tau)}\}$, where $a(i,\tau) = \arg\min_{a \in \Lambda_A} |A_i^\tau - \mu^{A,a}|$.

For each appearance triplet $(\mu^{A,a}, \mu^{A,b}, \mu^{A,c})$, we obtain the set of positions and orientations of the corresponding triplets of the modified data features:

$$\{(z_i^\tau, \theta_i^\tau), (z_j^\tau, \theta_j^\tau), (z_k^\tau, \theta_k^\tau) : \ s.t. (\mu^{A,a(i,\tau)}, \mu^{A,a(j,\tau)}, \mu^{A,a(k,\tau)}) = (\mu^{A,a}, \mu^{A,b}, \mu^{A,c})\} \tag{3.32}$$

We compute the ITV $\vec{l}$ of each triplet and perform k-means clustering to obtain a set of means $\mu_{abc}^{g,s}$ and covariances $\Sigma_{abc}^{g,s}$ for $s \in d_{abc}$, where $|d_{abc}|$ denotes the number of clusters. This gives the triplet vocabulary:

$$D = \{\mu_{abc}^{g,s}, \Sigma_{abc}^{g,s}, (\mu^{A,a}, \mu^{A,b}, \mu^{A,c}), (\Sigma^{A,a}, \Sigma^{A,b}, \Sigma^{A,c}) : \ s \in d_{abc}, \ a \leq b \leq c \ a, b, c \in \Lambda_A\}. \tag{3.33}$$

The triplet vocabulary contains geometric and appearance information (both mean and covariance) about the triplets that commonly occur in the images. This triplet vocabulary will be used to make proposals to grow the structure of the model (including giving initial conditions for learning the model parameters by the EM algorithm).

3.6.3.2 Structure Induction Algorithm

We now have the necessary background to describe our structure induction algorithm. The full procedure is described in the pseudo code in figure (3.9). Figure (3.6) shows an example of the structure being induced sequentially.

Initially we assume that all the data is generated by the background model. In the terminology of section (3.6), this is equivalent to setting all of the model parameters Ω to be zero (except those for the background model). We can estimate the parameters of this model and score the model as described in section (3.6).

Input: Training Image $\tau = 1, .., M$ and the triplet vocabulary Voc_2. Initialize G to be the root node with the background model, and let $G^* = G$.

Algorithm for Structure Induction:

- **STEP 1**:
  - OR-NODE EXTENSION

    For $T \in Voc_2$
    - $G' = G \bigcup T$ (OR-ing)
    - Update parameters of G' by EM algorithm
    - If $Score(G') > Score(G^*)$ Then $G^* = G'$
  - AND-NODE EXTENSION

    For Image $\tau = 1, .., M$
    - P = the highest probability parse for Image τ by G
    - For each Triple T in Image τ

      if $T \bigcap P \neq \emptyset$
      - $G' = G \bigcup T$ (AND-ing)
      - Update parameters of G' by EM algorithm
      - If $Score(G') > Score(G^*)$ Then $G^* = G'$
- **STEP 2:** $G = G^*$. Go to STEP 1 until $Score(G) - Score(G^*) < Threshold$

Output: G

Figure 3.9: Structure Induction Algorithm

Next we seek to expand the structure of this model. To do this, we use the triplet vocabularies to make proposals. Since the current model is the background model, the only structure change allowed is to add a triplet model as one child of the category node O (i.e. to create the background plus triple model described in the previous section, see figure (3.6)). We consider all members of the triplet vocabulary as candidates, using their cluster means and covariances as initial setting on their geometry and appearance properties in the EM algorithm as described in subsection (3.6.2). Then, for all these triples we construct the background plus triplet model, estimate their parameters and score them. We accept the one with highest score as the new structure.

As the graph structure grows, we now have more ways to expand the graph. We can add a new triplet as a child of the category node. This proceeds as in the previous paragraph. Or we can take two members of an existing triplet, and use them to construct a new triplet. In this case, we first parse the data using the current model. Then we use the triplet vocabulary to propose possible triplets, which partially overlap with the current model (and give them initial settings on their parameters as before). See figure (3.8). Then, for all possible extensions, we use the methods in section (3.6) to score the models. We select the one with highest score as the new graph model. If the score increase is not sufficient, we cease building the graph model. See the structured models in figure (3.11).

3.7 Experimental Results

Our experiments were designed to give proof of concept for the PGMM. Firstly, we show that our approach gives comparable results to other approaches for classification (testing between images containing the object versus purely background images) when tested on the Caltech-4 (faces, motorbikes, airplanes and background) [32] and Caltech 101 images [17] (note that most of these approaches are weakly supervised and so are

given more information than our unsupervised method). Moreover, our approach can perform additional tasks such as localization (which are impossible for some methods like bag of key points [36]). Our inference algorithm is fast and takes under five seconds (the CPU is AMD Opteron processor 880, 2.4G Hz). Secondly, we illustrate a key advantage of our method that it can both learn and perform inference when the 2D pose (position, orientation, and scale) of the object varies. We check this by creating a new dataset by varying the pose of objects in Caltech 101. Thirdly, we illustrate the advantages of having variable graph structure (i.e. OR nodes) in several ways. We first quantify how the performance of the model improves as we allow the number of OR nodes to increase. Next we show that learning is possible even when the training dataset consists of a random mixture of images containing the objects and images which do not (and hence are pure background). Finally we learn a *hybrid model*, where we are given training examples which contain one of several different types of object and learn a model which has different OR nodes for different objects.

3.7.1 Learning Individual Objects Models

In this section, we demonstrate the performance of our models for objects chosen from the Caltech datasets. We first choose a set of 13 object categories (as reported in [8]). Three classes of faces, motorbikes and airplanes are coming from [32]. We use the identical splitting for training and testing as used in [32]. The remaining categories are selected from Caltech-101 dataset [17]. To avoid concerns about selection bias, and to extend the number of object categories, we perform additional experiments on all object categories from [17] for which there are at least 80 images (80 is a cutoff factor chosen to ensure that there are a sufficient amount of data for training and testing). This gives an additional set of 26 categories (the same parameter settings were used on both sets). Each dataset was randomly split into two sets with equal size (one for

training and the other for testing). Note that the Caltech datasets, the objects typically appear in standardized orientations. Hence rotation invariance is not necessary. To check this, we also implemented a simpler version of our model which was not rotation invariant by modifying the $\vec{l}$ vector, as described in subsection (3.3.2). The results of this simplified model were practically identical to the results of the full model, that we now present.

K-means clustering was used to learn the appearance and triplet vocabularies where, typically, K is set to 150 and 1000 respectively. Each row in figure 3.5 corresponds to some triplets in the same group.

We illustrate the results of the PGMMs in Table (3.2) and Figure (3.10). A score of 90% means that we get a true positive rate of 90% and a false positive rate of 10%. This is for classifying between images containing the object and purely background images [32]. For comparison, we show the performance of the Constellation Model [32]. These results are slightly inferior to the bag of keypoint methods [36] (which requires weak supervision). We also evaluate the ability of the PGMMs to localize the object. To do this, we compute the proportion of AF's of the model that lie within the groundtruth bounding box. Our localization results are shown in Table (3.3). Note that some alternative methods, such as the bag of keypoints, are unable to perform localization.

The models for individual objects classes, learnt from the proposed algorithm, are illustrated in figure (3.11). Observe that the generative models have different tree-width and depth. Each subtree of the object node defines a Markov Random Field to describe one aspect of the object. The computational cost of the inference, using dynamic programming, is proportional to the height of the subtree and exponential to the maximum width (only three in our case). The detection time is less than five seconds (including the processing of features and inference) for the image with the

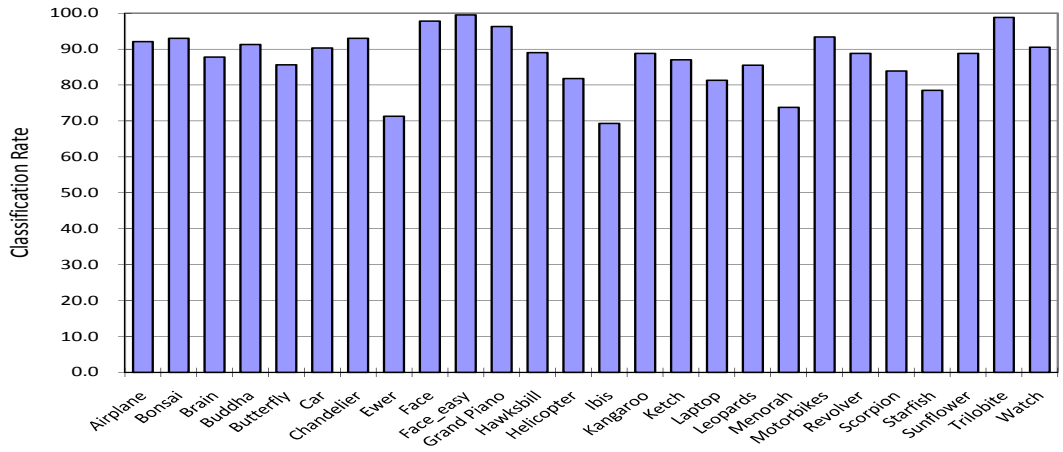


Figure 3.10: We report the classification performance for 26 classes which have at least 80 images. The average classification rate is 87.6%.

size of $320 * 240$. The training time is around two hours for 250 training images. The parsed results are illustrated in figure (3.12).

3.7.2 Invariance to Rotation and Scale

This section shows that the learning and inference of a PGMM is independent of the pose (position, orientation, and scale) of the object in the image. This is a key advantage of our approach and is due to the triplet representation.

To evaluate PGMMs for this task, we modify the Caltech 101 dataset by varying either the orientation, or the combination of orientation and scale. We performed learning and inference using images with 360-degree in-plane rotation, and another dataset with rotation and scaling together (where the scaling range is from 60% of the original size to 150% – i.e. $180 * 120 - 450 * 300$).

The PGMM showed only slight degradation due to these pose variations. Table (3.4) shows the comparison results. The parsing results (rotation+scale) are illustrated in figure (3.13).

Table 3.2: We have learnt probability grammars for 13 objects in the Caltech database, obtaining scores over 90% for most objects. A score of 90%, means that we have a classification rate of 90% and a false positive rate of $10\%(10\% = (100 - 90)\%)$. We compare our results with constellation model

Dataset	Size	Ours	Constellation Model
Faces	435	97.7	96.4
Motorbikes	800	92.9	92.5
Airplanes	800	91.8	90.2
Chair	62	90.9	–
Cougar Face	69	90.9	–
Grand Piano	90	96.3	–
Panda	38	90.9	–
Rooster	49	92.1	–
Scissors	39	94.9	–
Stapler	45	90.5	–
Wheelchair	59	93.6	–
Windsor Chair	56	92.4	–
Wrench	39	84.6	–

Table 3.3: Localization rate is used to measure the proportion of AF's of the model that lie within the groundtruth bounding box.

Dataset	Localization Rate
Faces	96.3
Motorbikes	98.6
Airplanes	91.5

Table 3.4: Invariant to Rotation and Scale

Method	Accuracy
Scale Normalized	97.8
Rotation Only	96.3
Rotation + Scale	96.3

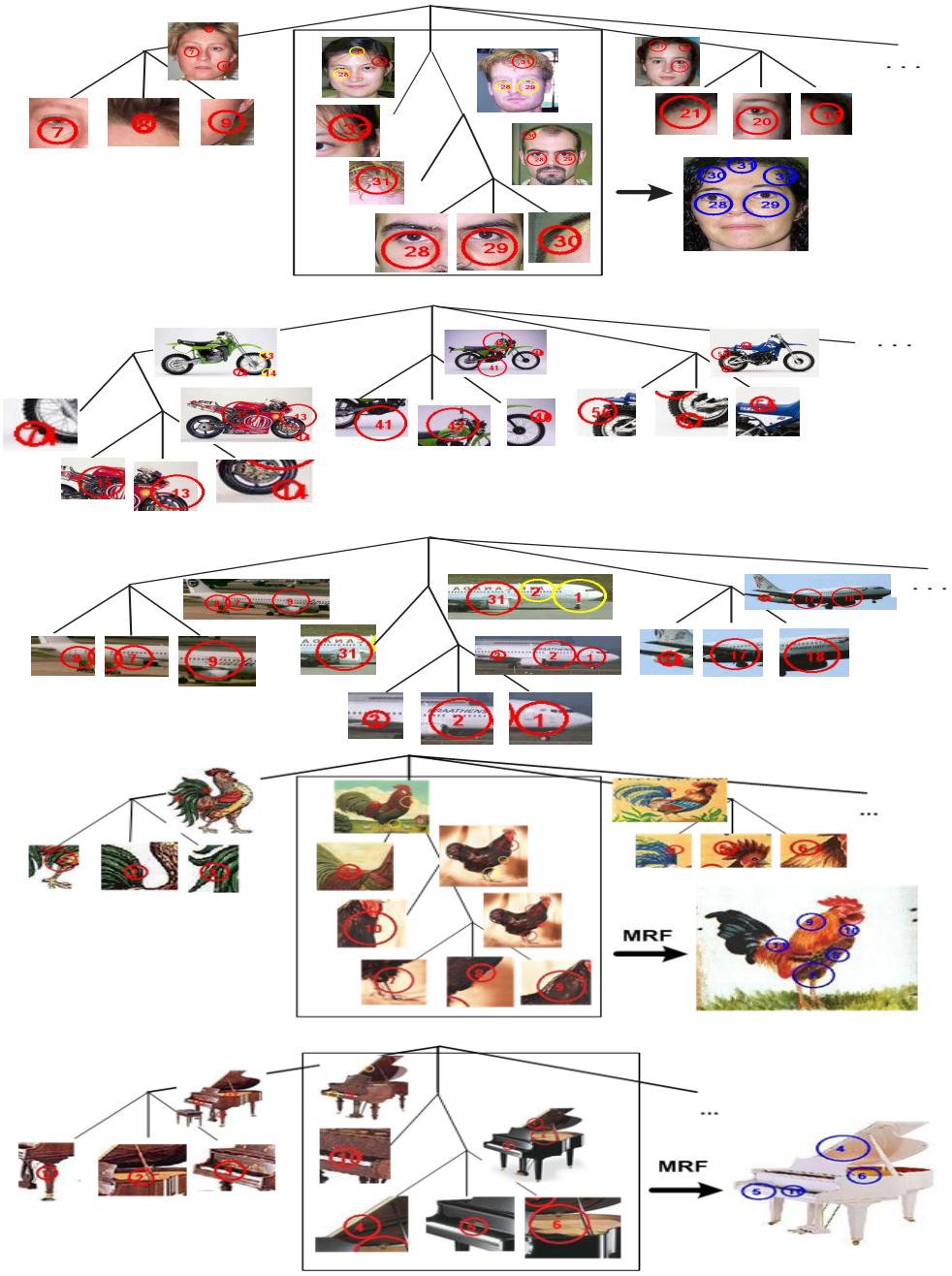


Figure 3.11: Individual Models learnt for Faces, Motorbikes, Airplanes, Grand Piano and Rooster. The circles represent the AF's. The numbers inside the circles give the a index of the nodes, see Table (3.1). The Markov Random Field of one aspect of Faces, Roosters, and Grand Pianos are shown on the right.

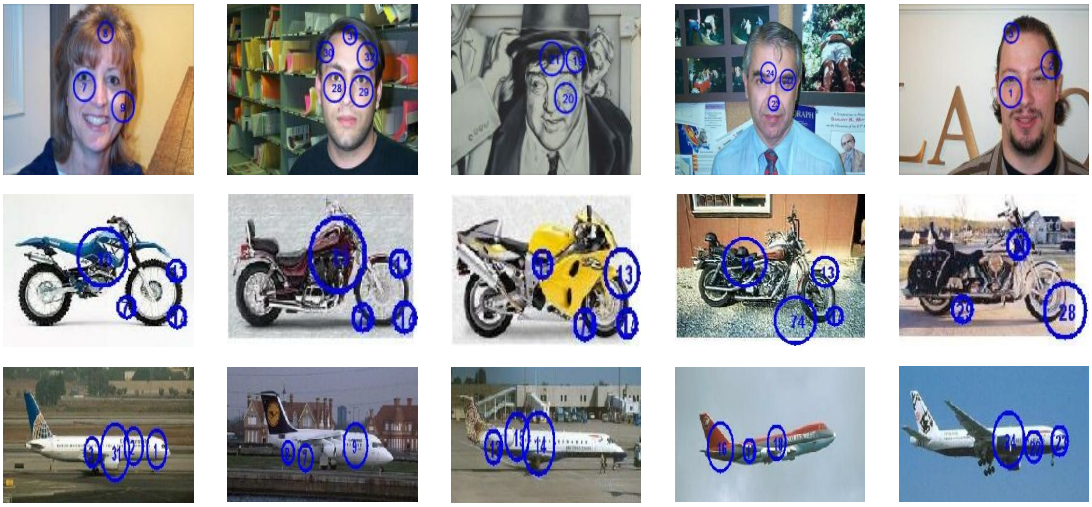


Figure 3.12: Parsed Results for Faces, Motorbikes and Airplanes. The circles represent the AF's. The numbers inside the circles give the a index of the nodes, see Table (3.1).

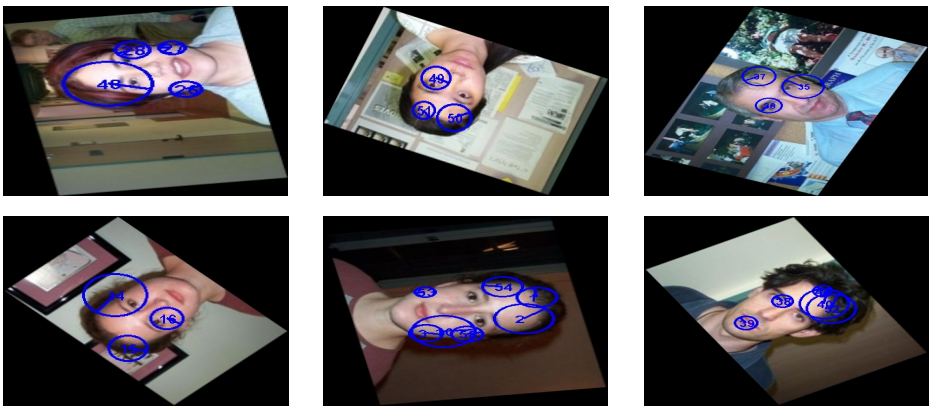


Figure 3.13: Parsed Results: Invariant to Rotation and Scale.

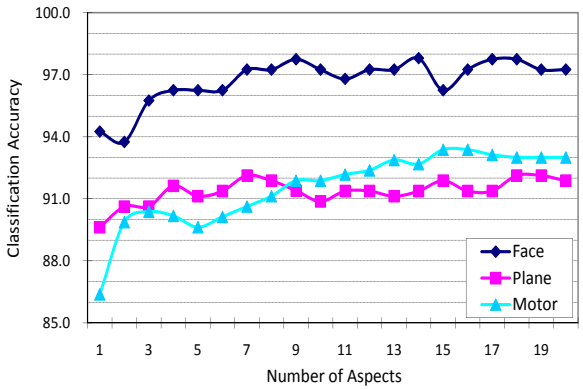


Figure 3.14: Analysis of the effects of adding OR nodes. Observe that performance rapidly improves, compared to the single MRF model with only one aspect, as we add extra aspects. But this improvement reaches an asymptote fairly quickly. (This type of result is obviously dataset dependent).

3.7.3 The Advantages of Variable Graph Structure

Our basic results for classification and localization, see section (3.7.1), showed that our PGMMs did learn variable graph structure (i.e. OR nodes). We now explore the benefits of this ability.

Firstly, we can quantify the use of the OR nodes for the basic tasks of classification. We measure how performance degrades as we restrict the number of OR nodes, see figure (3.14). This shows that performance increases as the number of OR nodes gets bigger, but this increase is jagged and soon reaches an asymptote.

Secondly, we show that we can learn a PGMM even when the training dataset consists of a random mixture of images containing the object and images which do not. Table (3.5) shows the results. The PGMM can learn in these conditions because it uses some OR nodes to learn the object (i.e. account for the images which contain the object) and other OR nodes to deal with the remaining images. The overall performance of this PGMM is only slightly worse that the PGMM trained on standard images (see

Table 3.5: The PGMM are learnt on different training datasets which consist of a random mixture of images containing the object and images which do not.

	Training Set		Testing Set		
Dataset	Object	Background	Object	Background	Classification Rate
Faces	200	0	200	200	97.8
Faces	200	50	200	200	98.3
Faces	200	100	200	200	97.7
Motor	399	0	399	200	93.7
Motor	399	50	399	200	93.2
Motor	399	100	399	200	93.0
Plane	400	0	400	200	92.1
Plane	400	50	400	200	90.5
Plane	400	100	400	200	90.2

section (3.7.1)).

Thirdly, we show that we can learn a model for an object class. We use a hybrid class which consists of faces, airplanes, and motorbikes. In other words, we know that one object is present in each image but we do not know which. In the training stage, we randomly select images from the datasets of faces, airplanes, and motorbikes. Similarly, we test the hybrid model on examples selected randomly from these three datasets.

The learnt hybrid model is illustrated in figure (3.15). It breaks down nicely into OR's of the models for each object. Table (3.6) shows the performance for the hybrid model. This demonstrates that the proposed method can learn a model for the class with extremely large variation.

Table 3.6: The PGMM can learn a hybrid class which consists of faces, airplanes, and motorbikes.

Dataset	Single Model	Hybrid Model
Faces	97.8	84.0
Motorbikes	93.4	82.7
Airplanes	92.1	87.3
Overall	–	84.7

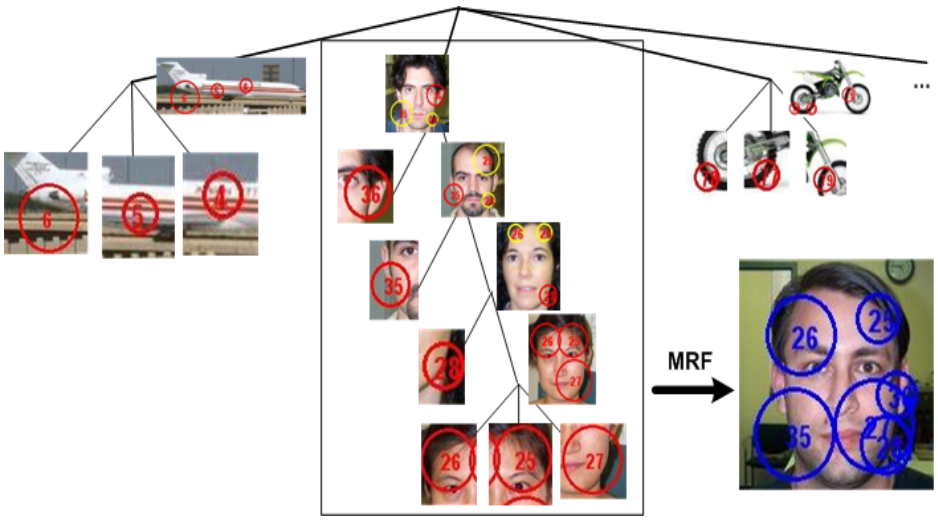


Figure 3.15: Hybrid Model learnt for Faces, Motorbikes and Airplanes.

3.8 Discussion

This chapter introduced PGMMs and showed that they can be learnt in an unsupervised manner and perform tasks such as classification and localization and of objects in unknown backgrounds. We also showed that PGMMs were invariant to 2D pose (position, scale and rotation) for both learning and inference. PGMMs could also deal with different appearances, or aspects, of the object and also learn hybrid models which include several different types of object.

More technically, PGMMs combine elements of probabilistic grammars and markov random fields (MRFs). The grammar component enables them to adapt to different aspects while the MRF enables them to model spatial relations. The nature of PGMMs enables rapid inference and parameter learning by exploiting the topological structure of the PGMM which enables the use of dynamic programming. The nature of PGMMs also enables us to perform structure induction to learn the structure of the model, in this case by using oriented triplets as elementary building blocks that can be composed to form bigger structures.

Our experiments demonstrated proof of concept of our approach. We showed that: (a) we can learn probabilistic models for a variety of different objects and perform rapid inference (less than five seconds), (b) that our learning and inference is invariant to scale and rotation, (c) that we can learn models in noisy data, for hybrid classes, and that the use of different aspects improves performance.

PGMMs are the first step in our program for unsupervised learning of object models. There are two critical limitations in PGMMs. I) PGMMs are unable to deal with large shape deformation. In other words, long-range shape correlation can not be easily encoded in PGMMs. II) Unsupervised learning of PGMMs is a greedy approach which highly relies on the good initialization of the triplet. In part II of this thesis, we

will resort to recursively compositional design to remedy these issues.

Part III

Object Parsing by Recursive Deformable Template

CHAPTER 4

Learning a Recursive Deformable Template Model

In this chapter, we address the problems of detecting, segmenting, parsing, and matching deformable objects. We propose a recursive deformable template model (RDTM) to represent objects in a hierarchical form. The object template consists of a small number of small sub-templates which is composed by smaller subsub-templates, and so on. RDTM represents both shape and appearance features at multiple levels of a hierarchy. This enables us to combine appearance cues at multiple scales and to model shape deformations at a range of scales. We provide a bottom-up algorithm which performs approximate inference for this hierarchical model. The algorithm is designed to be very fast while maintaining high precision and recall. We introduce the *structure-perceptron* algorithm to estimate the parameters of the RDTM in a discriminative way. The learning is able to estimate the appearance and shape parameters simultaneously. The structure-perceptron learning is able to perform feature selection (e.g. like AdaBoost) which enables us to specify a large dictionary of appearance and shape features and allow the algorithm to select which features to use and weight their importance. We have tested RDTM's for detection, segmentation, matching (alignment) and parsing. We show that the algorithm achieves state of the art performance for different tasks evaluated on datasets with groundtruth (when compared to algorithms which are specialized to the specific tasks).

4.1 Introduction

Detecting and parsing deformable objects in cluttered images is an important but unsolved problem in computer vision. They have many applications including object recognition, pose estimation and tracking. These tasks are difficult due to four major reasons – shape deformation, appearance variation, cluttered backgrounds, and occlusion. Although there have been some partial successes – see [50, 52, 53, 54] and others reviewed in section (4.2) – none are close to the performance level and computational speed achieved for detecting rigid objects by using techniques such as AdaBoost [55, 56]. In our opinion, serious disadvantages of the current approaches are that they are based on representations of the object that only use sparse image cues, short range spatial interactions, or some combination. See figure (4.1). Hence these representations fail to capture important information about the object, which reduces their performance and restricts the set of tasks that they can achieve (e.g. you cannot perform segmentation using only sparse image cues). In practice, the choice of object representations is strongly restricted by the availability of effective inference algorithms. For example, the object representations used in [50] and [52] were chosen so that dynamic programming (DP) and belief propagation (BP) could be used respectively. In addition, current methods do not learn the models. This limitation constrains the use of richer image features. Hence we argue that progress in this area requires a strategy that simultaneously develops powerful representations, suitable inference and learning algorithms.

In this chapter, we propose a new class of object models – *Recursive Deformable Template Model (RDTM)* – which represent objects in a hierarchical form. The object template consists of a small number of small sub-templates which is composed by smaller subsub-templates, and so on. RDTM represents a large variety of cues and spatial interactions at a range of scales. See figure (4.1, 4.2). The RDTM is very versa-

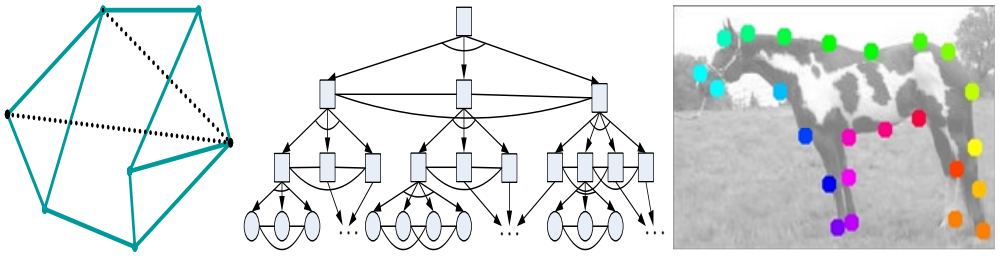


Figure 4.1: Alternative representations for deformable objects. Left panel: standard models use "flat" Markov Random Field (MRF) models relying on sparse cues and with limited spatial interactions. Middle panel: a RDTM has a hierarchical representation of a large variety of different images cues and spatial interactions at a range of scales. Right panel: The points along the object boundary correspond to the nodes in the "flat" MRF models or the leaf nodes of the hierarchical model.

tile since it gives a rich representation of the object which makes it suitable for a range of visual tasks such as detection, segmentation, parsing, and matching/alignment. We also describe a novel bottom-up inference algorithm – *compositional inference* – which esuriently is a prunded version of DP and enables us to detect and parse the RDTM rapidly. Finally, we extend the recent *structure perceptron learning algorithm* [57] in order to perform supervised learning of the parameters of RDTM.

We perform inference on a RDTM using a bottom-up strategy where the bottom-up process rapidly makes a number of proposals for the state of the object. The bottom-up proposals are state representations of subparts of the hierarchy. These proposals are generated based on the principle of composition by combining local cues for the location and pose of different subparts of the object hierarchically. To keep the number of proposals small, we use a *threshold* to reject proposals and use *surround suppression* to select the local winner and keep the remainder in a cluster suitable for later processing at upper levels. Proposals at the top level of the hierarchy correspond to complete state representations of the object. This strategy was inspired by a compositional algorithm

[7, 8] which did not use a hierarchical model and was only tested on a small number of images. But we have other algorithm on AND/OR graph [11].

We learn the parameters of the RDTM by adapting the structure-perceptron algorithm [57]. This enables us to learn all the parameters globally in a consistent manner (i.e. at all levels of the hierarchy simultaneously). It also allows us to select different shape and appearance features from a dictionary and determine ways to optimally weight them (similar to the selection and weighting strategy used in AdaBoost [55, 56]). Structure-perceptron learning is a discriminative approach that is computationally simpler than standard methods such as maximum likelihood estimation (as used, for example for learning Conditional Random Fields [23]). Moreover, there are advantages to discriminative learning because this strategy focusses attention on estimating parameter values of the model most relevant to decision making (e.g. about segmentation or matching). We have shown the success of structure-perceptron learning in [12].

We demonstrate the success and versatility of RDTM's by applying them to a range of visual tasks. We show that they are very effective in terms of performance and speed (less than 5 seconds for a typical $300X200$ image, speed increases approximately linearly in the size of the image) when evaluated on large datasets which include horses [18] and cows [58]. In particular, to illustrate versatility, we demonstrate state-of-the-art results for different tasks of object segmentation (evaluated on the Weismann horse dataset [18]) and matching/alignemnt (evaluated on the face dataset – [19, 59]). The results on the alignment task on the face dataset are particularly interesting because we are comparing to results obtained by methods such as Active Appearance Models [60] which are specialized for faces and which have been developed over a period of many years (while we spent one week in total to run this application including the time to obtain the dataset). Overall, we demonstrate that RDTM's can perform a large range

of visual tasks while other computer vision methods typically restrict themselves to single tasks.

We perform diagnostics to quantify how different components of the system contribute to performance and at what computational cost (i.e. speed). In particular, we compare the contributions of the bottom-up processes (at all levels of the hierarchy). We hope that this analysis of the tradeoffs between speed and performance will yield general principles for optimal design of modeling and inference for computer vision systems particularly those requiring multi-level processing.

We note that certain aspects of RDTM's have similarities to the human visual system and, in particular, to biologically inspired vision models. The bottom-up process by its use of surround suppression and its transition from local to global properties is somewhat analogous to Fukushima's neocognitron [61] and more recent embodiments of this principle [62, 63].

4.2 Background

There is a vast literature on techniques for the separate tasks of object detection, segmentation, parsing, and matching/aligning. We give a brief review of the work that is the most relevant to our approach.

There has been a range of attempts to model deformable objects in order to detect, register, and recognize them. Many of them can be formulated as maximum a posteriori inference of the position states y of the object parts in terms of the data x. Formally, they seek to estimate

$$y^* = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y), \tag{4.1}$$

where $p(x|y)p(y) = p(x, y)$ is of form:

$$p(x, y) = \frac{1}{Z} \exp\{\sum_i \alpha_i f(x_i, y_i) + \sum_{i,j} \beta_{ij} g(y_i, y_j)\}. \quad (4.2)$$

where Z is the normalization constant. The unary potentials $f(x, y)$ model how well the individual features match to the positions in the image. The binary potentials $g(y_i, y_j)$ impose (probabilistic) constraints about the spatial relationships between feature points. Typically, y is defined on a flat MRF model and the number of its nodes is considerably small. See figure 4.1.

Coughlan et. al [50] provided one of the first models of this type, using a sparse representation of the boundary of a hand, and showed that dynamic programming (DP) could be used to detect the object (without need of initialization). This type of work was extended by Felzenswalb [64] and by Coughlan using pruned version of BP [52]. The main limitation of this class of model is that it only involves local pairwise interactions between points/features (see the second term in equation (4.2). This restriction is mainly due to computational reasons (i.e. the types of inference algorithms available) and not for modeling reasons. For example, the performance of BP is known to degrade for representations with many closed loops. See figure (4.1)).

Other classes of models are more suitable for matching than detection [53, 52, 65]. Some of these models [52, 65] do use longer range spatial interactions, as encoded by shape context and other features, and global transformations. But these models are typically only rigorously evaluated on matching tasks (i.e. tested on large datasets with groundtruth). They all need good initialization for position, orientation, and scale if they are required to detect objects in images with background clutter.

Recent work has introduced hierarchical models to represent the structure of objects more accurately (and enable shape regularities at multiple scales). Shape-trees were presented [66] to model shape deformations at multiple levels. Chen et. al [11]

propose an AND/OR graph representation (similar to [26, 49]), which is a multi-level mixture Markov Random Field, and provide a novel bottom-up and top-down based inference algorithm. But both of these models concentrate on modeling the shape deformation at different scales and use simple appearance models defined at leaf nodes only.

A major limitation of all these models is that they are not learnt from data. The image features were manually designed (and hence are comparatively simple), the geometry models were hand-specified, and the relative weights of appearance and shape had to be manually tuned.

Object segmentation aims at finding the boundary of the object and typically assumes that the rough location is known. It does not involve recovering the pose (i.e. position, orientation, and scale) of the object. But work on this topic has used learning and cues at multiple scales.

Borenstein and Ullman [18] provide a public horse dataset and study the problem of deformable object segmentation on this dataset. Torr and his colleagues [1] develop Object-Cut which locates the object via a pictorial model learnt from motion cues and use the min-cut algorithm to segment out the object of interest. Ren et. al [16] address the segmentation problem by combining low-, mid- and high-level cues in Conditional Random Field (CRF). Similarly, Levin and Weiss [4] utilize CRF to segment object but assuming that the position of the object is roughly given. In contrast to supervised learning, Locus [67] explores a unsupervised learning approach to learn a probabilistic object model. Recently, Cour and Shi [68] currently achieve the best performance on this horse dataset. It is important to note that none of these methods report performance on matching/alignment.

4.3 Recursive Deformable Template Model (RDTM)

This section describes the basic structure of the RDTM. Firstly we describe the graphical structure in subsection (4.3.1). Secondly we specify the state variables and the form of the probability distribution in subsection (4.3.2). Thirdly, in subsection (4.3.3), we describe the learning procedure used to determine the graph structure from one example. The inference and learning algorithms will be described in sections (4.4,4.5) respectively.

4.3.1 The Graphical Structure of the RDTM

We represent an object by a hierarchical graph defined by parent-child relationships. The top node of the hierarchy represents the position of the center of the object. The leaf nodes represent points on the object boundary and the intermediate nodes represent different subparts of the object. This is illustrated in figure (4.2). We use ν to index nodes of the hierarchy. The set of all nodes is denoted by V. The set of child nodes of ν is denoted by T_ν (i.e. T_ν specifies the vertical edges of the graph). In this chapter, the horizontal dependencies will be built out of the triples of nodes (μ, ρ, τ) in T_ν, see figure (4.3). Each node ν is also connected to image data to encode the appearance of its corresponding region. Hence the hierarchical graph is specified by $\{\nu, T_\nu, (\mu, \rho, \tau)\}$ (i.e. the nodes, the vertical edges, and the horizontal edges).

4.3.2 The state variables and the potential functions

A configuration of an RDTM is an assignment of state variables $y = \{y_\nu\}$ to all nodes $\{\nu\}$ of the hierarchy. The node variable at node ν is written as $y_\nu = (Px_\nu, Py_\nu, \theta_\nu, s_\nu)$ at each node ν, where (Px, Py), θ and s denote position, orientation, and scale respectively. It is an abstraction of the state variables of its child nodes. All these variables

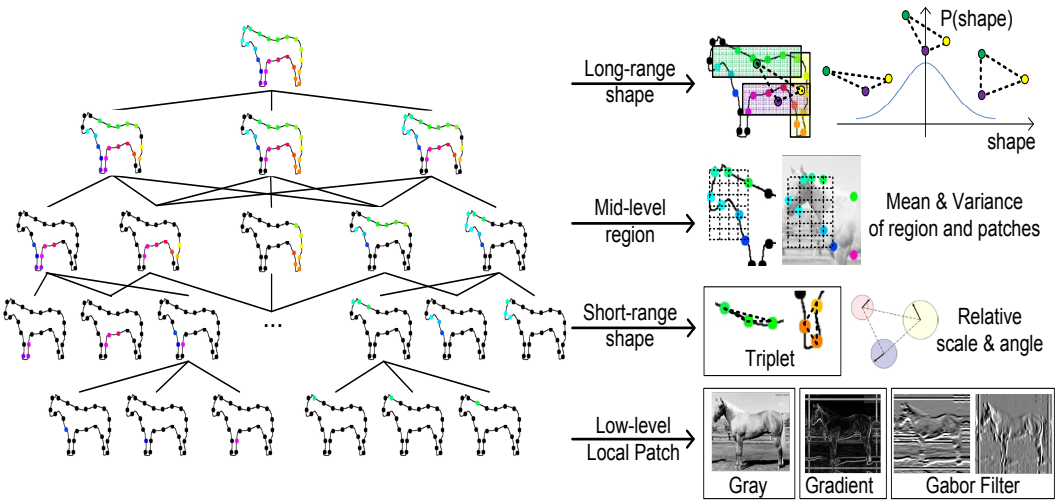


Figure 4.2: The hierarchical graph of the RDTM is constructed by a hierarchical clustering algorithm (see text for details). Black dots indicate the positions of the leaf nodes in the hierarchy. Color dots indicate the subparts which correspond to particular nodes of the hierarchy. The appearance and shape deformation are modeled at multiple levels in the hierarchy.

are unobservable. Intuitively, each node ν corresponds to a sub-region of the image determined by $(Px_\nu, Py_\nu, \theta_\nu, s_\nu)$ (rectangle region centered at (Px_ν, Py_ν) with size of s_ν). Observe that the state variables take the same form at all levels of the hierarchy (unlike standard hierarchical representations [8, 11]). This design is what we called "Recursive Composition Principle".

We define a conditional probability distribution over these state variables with potentials specified over the horizontal and vertical connections specified of the hierarchical graph structure. This enables the model to represent geometrical and data relations at different scales. This differs from standard non-hierarchical models ([4, 16]) and even some hierarchical models ([8]) which only specify data relations at the lowest levels of hierarchy (even though they do specify geometric relationships at all scales).

More precisely, the conditional distribution is specified by a log-linear model:

$$P(y|x;\alpha) = \frac{1}{Z(x;\alpha)} \exp\{\Phi(x,y) \cdot \alpha\}, \tag{4.3}$$

where x denotes the input image, y is the state of the RDTM, $\Phi(x,y)$ are potential functions, α are the parameters of the distribution (which will be learnt in section (4.5)), and $Z(x;\alpha)$ is the normalization function. The inner product is of the form:

$$\Phi(x,y) \cdot \alpha = \sum_{\nu \in V} \sum_{i} \alpha^D_{\nu,i} \Phi^D_{\nu,i}(x,y) + \sum_{\nu \in V} \sum_{(\mu,\rho,\tau) \in T_\nu} \alpha^H_{\nu,(\mu,\rho,\tau)} \Phi^H_{\nu,(\mu,\rho,\tau)}(y) + \sum_{\nu \in V} \alpha^V_\nu \Phi^V_\nu(y) \tag{4.4}$$

where the summation is calculated on potential functions defined over the hierarchy. More specifically, $\Phi(x,y)$ takes three forms: (i) the *data terms* $\Phi^D(x,y)$, (ii) the *horizontal terms* for spatial relations $\Phi^H(y)$, and (iii) the *vertical terms* $\Phi^V(y)$. These terms are defined in terms of *dictionaries* of features from which the learning algorithm selects and weights a restricted subset, see section (4.5). We now describe the three different forms of potentials.

The *data terms* $\Phi^D_{\nu,i}(x,y) = f_i(x, y_\nu)$ determine how the RDTM interacts with the image feature f_i calculated on the image region determined by $y_\nu = (Px_\nu, Py_\nu, \theta_\nu, s_\nu)$. The image features are defined for the nodes at all levels of the hierarchy (see figure 4.2). For leaf nodes, $\Phi^D(x,y)$ are specified by a dictionary of local image features computed by different operators. This leaf dictionary include the intensity, its gradient, Canny edge detectors, Difference of Offset Gaussian (DOOG) at different scales (13*13 and 22*22) and orientations $(0, \frac{1}{6}\pi, \frac{2}{6}\pi, ...)$, and so on (the bottom row of figure 4.2). There are 27 image features in total for leaf nodes. For non-leaf nodes (the second row of the right panel of figure 4.2), $\Phi^D(x,y)$ is specified by a dictionary of regional features (e.g. mean, variance, histogram of image features) defined over the sub-regions specified by the node state y_ν.

The *horizontal terms* and *vertical terms* encode the geometrical priors. The *horizontal terms* impose the horizontal connections at a range of scales (see the top and third rows in figure 4.2). It is defined over all triples μ, ρ, τ formed by the child nodes of each parent. See figure 4.3. Its form is given by $\Phi^H_{\nu,(\mu,\rho,\tau)}(y) = g(y_\mu, y_\rho, y_\tau)$, where $g(.,.,.)$ is a logarithm of Gaussian distribution defined on the *invariant shape vector* (ITV) $l(y_\mu, y_\rho, y_\tau)$ constructed from (y_μ, y_ρ, y_τ) [8]. The ITV depends only on variables of the triple, such as the internal angles, which are invariant to the translation, rotation, and scaling of the triple. This ensures that the potential is also invariant to these transformations. The parameters of the Gaussian are learnt from training data as described in section (4.5).

The *vertical terms* $\Phi^V(y)$ are used to hold the structure together by relating the state of the parent nodes to the state of their children. The state of the parent node is determined precisely by the states of the child nodes. This is defined by $\Phi^V_\nu(y) = h(y_\nu, \{y_\mu \; s.t. \mu \in T_\nu\})$, where T_ν is the set of child nodes of node ν, $h(.,.) = 0$ if the average orientations and positions of the child nodes are equal to the orientation and

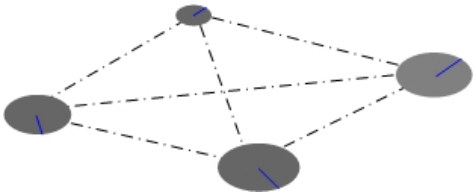


Figure 4.3: Representation based on oriented triplet. This gives the cliques for the four children of a node. In this example, four triplets are computed. Each circle corresponds to one node in the hierarchy which has a descriptor (indicated by blue line) of position, orientation and scale. The potentials of the cliques are Gaussians defined over features extracted from triple nodes, such as internal angles of the triangle and relative angles between the feature orientation and the orientations of the three edges of the triangle. These exacted features are invariant to scale and rotation.

position of the parent node. If they are not consistent, then $h(.,.) = \kappa$, where κ is a large negative number.

In summary, the hierarchical representation decomposes both the appearance and shape modeling into multiple levels. At low levels of the hierarchy (the third and fourth rows of figure 4.2), the short-range shape constraints between small parts are modeled together with the small-scale appearance cues. At higher levels (the top and second rows of figure 4.2), the long-range shape regularities between larger parts are imposed and large scale appearance cues are used.

4.3.3 Constructing the Hierarchy by One-example Learning

In this chapter, we learn the hierarchical graph from a single example of the object. We call this "one-example learning". The input is the set (P_x, P_y, θ) of points on the object boundary curve together with their orientation (i.e. the normal vector to the curve).

We automatically construct the hierarchical graph by a hierarhical aggregation al-

gorithm which is partially inspired by the "Segmentation by Weighted Aggregation (SWA) " algorithm [69]. The input is a weighted graph $G = \{V, E, W\}$ where V is the vertex set, E is the edge set, W is the weights. At the bottom level the vertices are the edge points along the entire boundary. $W_{i,j} = \exp\{-\beta_1 dist(z_i, z_j) + \beta_2 edge(z_i, z_j)\}$ where z_i is the position of point i, $dist(.,.)$ is the distance function and $edge(.,.)$ is an indicator function to measure if point i, j are neighbors or not. β_1 and β_2 are set to be 0.5 and 1 respectively (identically in all experiments).

The output is a hierarchical graph structure (the state variables of the example are thrown out), i.e. a set of nodes and there vertical and horizontal connections. We observe that the hierarchical graph gives a natural parsing of the exemplar (see figure 4.2). See the details in [69].

In one-example learning, all the parameters α are manually set to be 1 , i.e. they are equal to each other. The data terms $\Phi^D(x, y)$ are only defined at leaf nodes. The image feature is merely the intensity gradient. There is no region features for non-leaf nodes. For the horizontal terms $\Phi^H(y)$, the Gaussian distribution for $g(y_\mu, y_\rho, y_\tau)$ is defined according to the single input example. More precisely, the mean is obtained by the example and the variance parameter is set by hand. We will present how to learn the parameters α to include more image features and weigh them by structure-perceptron learning in section 4.5.

4.4 Inference: Parsing the Model

We now describe an inference algorithm suited to the hierarchical structure of the RDTM. Its goal is to obtain the best state y^* by estimating $y^* = \arg\max \Phi(x, y) \cdot \alpha$ which is defined in equation (4.4). We use a bottom-up strategy which makes proposals for the state variables from the bottom level to the top level. The bottom-up process

works by combining proposals for the states of the lower-levels nodes to form proposals for the higher-levels nodes. A snapshot of the algorithm is shown in figure (4.4). This algorithm has been developed in previous work [8, 11].

The algorithm is an example of approximate inference and, at present, no guarantees can be given for its performance and temporal efficiency. The bottom-up process can be considered as a form of dynamic programming with a specific form of pruning which exploits the hierarchical structure. In practice, the algorithm performs very well and runs in polynomial time in terms of the size of input image and the number of levels of the hierarchy. This time efficiency is required to make learning practical, see section (4.5), as well as to ensure rapid inference to detect, parse, segment, and match/align the HLLM.

The bottom-up process is designed to be computationally cheap and to have very few false negatives so that the objects are almost always found as a *variant* of one of the proposals. Cheapness is achieved by keeping the number of proposals small (avoiding the danger of combinatorial explosion due to composition) while avoiding false negatives (similar to the motivation for cascades [55, 56]). This is achieved by: (i) using thresholding to reject proposals whose probability is too small, and (ii) by surround suppression which groups proposals into clusters of similar proposals and represents them by a single proposal (the one with highest probability). These mechanisms ensures that the algorithm has linear scaling with image size, as shown in section (4.6). We will empirically quantify the performance of each component of the hierarchy in section (4.6.3).

The basic strategy of the bottom-up process is compositional. The algorithm seeks to find subparts of the object and to combine them to make bigger subparts, until eventually the whole object is detected. Each node ν^l at level l has a set of proposals $\{P^l_{\nu,a} : a = 1, ..., M^l_\nu\}$ where M^l_ν is the number of proposals for node ν^l. There are also

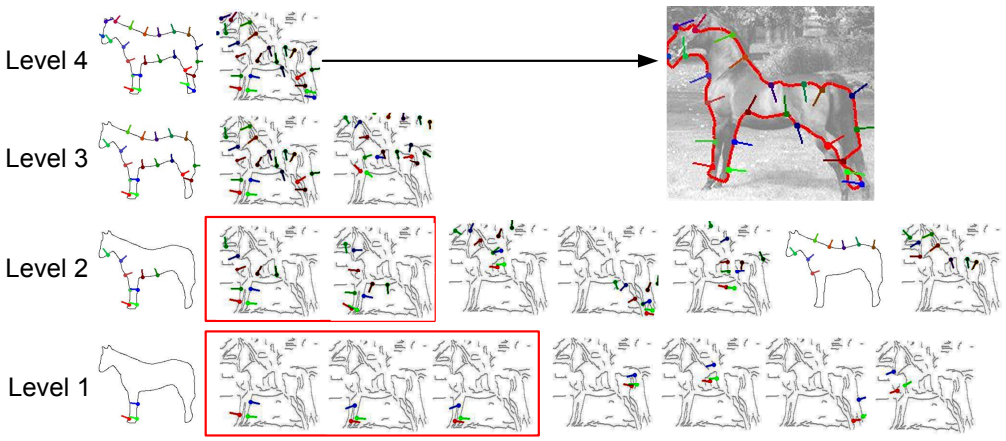


Figure 4.4: This snapshot illustrates the bottom-up inference algorithm. The bottom-up process starts from level 1 and constructs proposals from children nodes. Only proposals above threshold are kept. Similar proposals in the same local window defined over space, orientation and scale are grouped together. See the proposals inside the windows. The proposal with the highest score within this cluster is kept to propagate to upper level.

max-proposals $\{MP^l_{\nu,a}\}$, indexed by a, each associated with a local cluster $\{CL^l_{\nu,a}\}$ of proposals. Each proposal is described by a state vector $\{y^l_{\nu,a} : a = 1, ..., M^l_\nu\}$, the state vectors for it and its descendants $\{\Lambda^l_{\nu,a} : a = 1, ..., M^l_\nu\}$, and an energy function score $\{E^l_\nu(\Lambda^l_{\nu,a}) : a = 1, ..., M^l_\nu\} = \Phi(x, y) \cdot \alpha$ where the potential functions $\Phi(x, y)$ are only active for $y = \Lambda^l_{\nu,a}$. Each proposal, or max-proposal, represents a cluster $CL^l_{\nu,a}$ of related proposals – which are above threshold T_l, but which have higher energy (lower probability) than the max-proposal and so are suppressed.

We obtain the proposals by a bottom-up strategy starting at level $l = 0$ of the tree where only image cues are used, i.e. no spacial relationship is considered. Then we move to level $l = 1$ to explore all compositions of proposals from level $l = 0$. For a node ν^1 we define windows $\{W^1_{\nu,a}\}$ in space, orientation, and scale. We exhaustively search for all configurations within this window which have a score $E^1_\nu(\Lambda^1_\nu) > T_1$, where T_1 is a fixed threshold. For each window $W^1_{\nu,a}$, we select the configuration with largest score to be the maximum proposal $MP^1_{\nu,a}$ and store the remaining proposals above threshold in the associated cluster $CL^1_{\nu,a}$ (of course, many windows will contain no proposals above threshold – see section (4.6)). This window enforces surround suppression and, together with the threshold, ensures that we do not obtain too many proposals in the hierarchy.

The procedure is repeated as we go up the hierarchy. Each parent node ν^{l+1} has a set of windows $\{W^l_{\nu,a}\}$ and produces proposals $\{MP^{l+1}_{\nu,a}\}$, and associated clusters $\{CL^{l+1}_{\nu,a}\}$, by combining the proposals from its children. All proposals are required to have scores $E^{l+1}_\nu(\Lambda^{l+1}_\nu) > T_{l+1}$, where T_{l+1} is a threshold. The results are a set of proposals, and associated clusters of additional proposals, at all nodes in the hierarchical graph .

In our experiments, the thresholds T_l are set to be certain values such that the recall in the training data is 95%. In other words, for any object parts corresponding to the

Input: $\{MP^1_{\nu^1}\}$. Output:$\{MP^L_{\nu^L}\}$

- Bottom-Up(MP^1)

  Loop : $l = 1$ to L, for each node ν at level l

  1. Composition: $\{P^l_{\nu,b}\} = \oplus_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
  2. Pruning: $\{P^l_{\nu,a}\} = \{P^l_{\nu,a} | \Phi_\nu(x, \Lambda^l_{\nu,a}) \cdot \alpha > T_l\}$
  3. Local Maximum: $\{(MP^l_{\nu,a}, CL^l_{\nu,a})\} = LocalMaximum(\{P^l_{\nu,a}\}, \epsilon_W)$ where ϵ_W is the size of the window W^l_ν defined in space, orientation, and scale.

Figure 4.5: The inference algorithm. $\oplus$ denotes the operation of combining two proposals.

nodes in the hierarchy, 95% of training examples are correctly detected by using the thresholds to prune out proposals.

4.5 Structure-Perceptron Learning

We now describe our learning algorithm. This constructs the RDTM probability distribution by selecting and weighting features from the dictionaries. Recall that the graph structure of the RDTM has already been learnt from one example by the hierarchical clustering algorithm, see subsection (4.3.3). Thus the task of learning the RDTM is to estimate the weights of features specified in section (4.3.2).

4.5.1 Background on Perceptron and Structure-Perceptron Learning

Perceptron learning was developed for classification tasks but its theoretical properties, such as convergence and generalization, have only recently been justified [70]. More

recently, Collins [57] developed the structure-perceptron algorithm which applies to situations where the output is a structure (e.g. a sequence or tree of states). He obtained theoretical results for convergence, for both separable and non-separable cases, and for generalization. In addition Collins and his collaborators demonstrated many successful applications of structure-perceptron to natural language processing, including tagging [71] (an example of a sequence/chain output), and parsing [72] (an example of tree output).

Structure-perceptron learning can be applied to learning log-linear models such as RDTM. The learning proceeds in a discriminative way. By contrast to maximum likelihood learning, which requires calculating the expectation of features, structure-perceptron learning only needs to calculate the most probable configurations (parses) of the model. Therefore structure-perceptron learning is more flexible and computationally simpler (i.e. the max calculation is usually easier than the sum calculation).

To the best of our knowledge, structure-perceptron learning has never been exploited in computer vision except our previous work [12] (unlike the perceptron which has been applied to binary classification and multi-class classification tasks). Moreover, we are applying structure-perceptron to more complicated models (i.e. RDTMs) than those treated by Collins [71] (e.g. Hidden Markov Models for tagging).

4.5.2 Structure-Perceptron Learning

The goal of structure-perceptron learning is to learn a mapping from inputs $x \in X$ to output structure $y \in Y$. In our case, X is a set of images, with Y being a set of possible parse trees (i.e. configuration of RDTM's) which specify the positions, orientations, scales of objects and their subparts in hierarchical form. We use a set of training examples $\{(x_i, y_i) : i = 1...n\}$ and a dictionary of functions $\{\Phi\}$ which map each $(x, y) \in X \times Y$ to a feature vector $\Phi(x, y) \in R^d$. The task is to estimate

a parameter vector $\alpha \in R^d$ for the weights of the features. This can be interpreted as a feature selection process by giving a default value of 0 to each parameter vector, so that only features that are selected have non-zero weight. The feature vectors $\Phi(x, y)$ can include arbitrary features of parse trees, as we discussed in section 4.3.1.

The loss function used in structure-perceptron learning is of form:

$$Loss(\alpha) = \Phi(x, y) \cdot \alpha - \max_{\overline{y}} \Phi(x, \overline{y}) \cdot \alpha, \tag{4.5}$$

where y is the correct state configuration for input x, and $\overline{y}$ is a dummy variable.

The basic structure-perceptron algorithm – *Algorithm I* – is designed to minimize the loss function. Its pseudo-code is given in figure 4.6. The algorithm proceeds in a simple way (similar to the perceptron algorithm for classification). The parameters are initialized to zero and the algorithm loops over the training examples. If the highest scoring parse tree for input x is not correct, then the parameters α are updated by an additive term. The most difficult step of the method is to find $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$. But this can be performed by the inference algorithm described in section (4.5). Hence the performance and efficiency (empirically polynomial complexity) of the inference algorithm is a necessary pre-condition to using structure-perceptron learning for RDTM's.

4.5.3 Averaging Parameters

There is a simple refinement to Algorithm I, called *"the averaged parameters"* method (Algorithm II), whose pseudo-code is given in figure 4.7. The averaged parameters are defined to be $\gamma = \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha^{t,i}/NT$, where NT is the averaging window. It is straightforward to store these averaged parameters and output them. The theoretical analysis in [57] shows that Algorithm II (with averaging) gives better performance and convergence rate than Algorithm I (without averaging). We will empirically compare

Input: A set of training images with ground truth (x^i, y^i) for $i = 1..N$. Initialize parameter vector $\alpha = 0$.

Algorithm I:

For $t = 1..T, i = 1..N$

- Use bottom-up inference to find the best state of the model on the i'th training image with current parameter setting, i.e., $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$
- Update the parameters: $\alpha = \alpha + \Phi(x^i, y^i) - \Phi(x^i, y^*)$

Output: Parameters α

Figure 4.6: Algorithm I: a simple training algorithm of structure-perceptron learning

Algorithm II:

For $t = 1..T, i = 1..N$

- Parse: $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$
- Store: $\alpha^{t,i} = \alpha$
- Update: $\alpha = \alpha + \Phi(x^i, y^i) - \Phi(x^i, y^*)$

Output: Parameters $\gamma = \sum_{t,i} \alpha^{t,i}/NT$

Figure 4.7: Algorithm II: a modification of algorithm I.

these two algorithms in section (4.6).

4.5.4 Feature Selection

We emphasize that structure-perceptron learning can be considered as a procedure of feature selection (similar to AdaBoost). We specify a dictionary of features $\{\Phi\}$ and initialize their parameters $\{\alpha\}$ to be zero. As the algorithm proceeds, it assigns non-zero weights to some features thereby selecting them. This ability to perform feature selection allows us to specify a large dictionary of possible features and enable the algorithm to select those features which are most effective. This allows us to learn RDTMs for different objects *without* needing to specially design features for each object. In addition, feature selection allows us to automatically select features defined at different levels.

This ability to automatically select features from a dictionary means that our approach is more flexible than existing conditional models (e.g., CRF [16, 4, 21]) which use multi-level features but with fixed scales (i.e. not adaptive to the configuration of the hidden state). In section 4.6.5, we empirically study what features the structure-perceptron algorithm judges to be most important for a specific object like a horse. Section 4.6.6 also illustrates the advantage of feature selection by applying the same learning algorithm to the different task of face alignment without additional feature design.

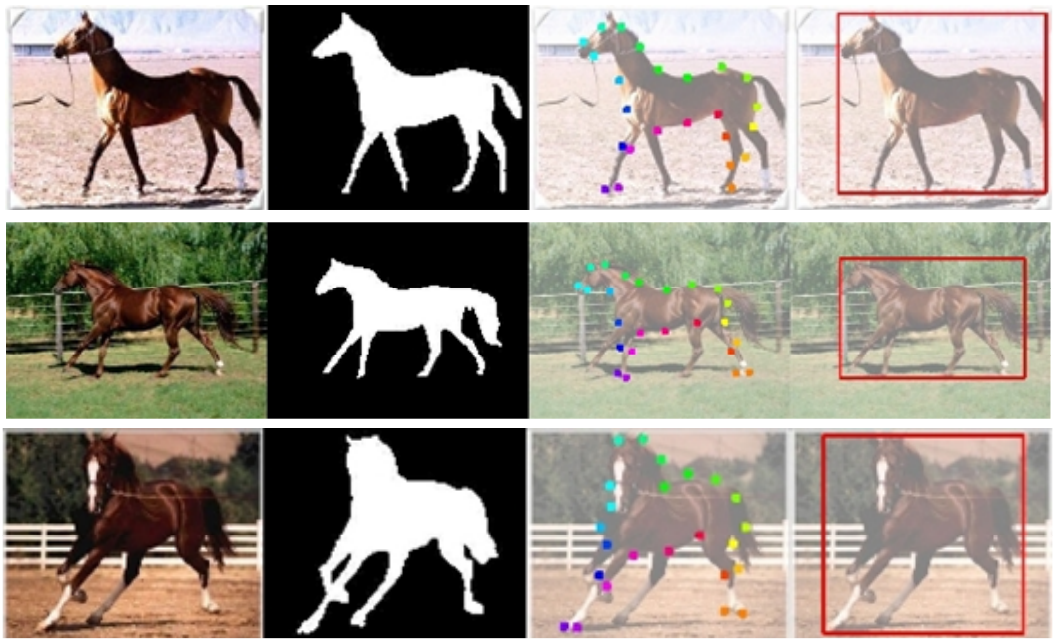


Figure 4.8: Examples of the Weizmann horse data set. This figure shows input image, ground truth of segmentation, parsing (position of leaf nodes) and detection, from left to right respectively.

4.6 Experimental Results

4.6.1 Dataset and Evaluation Criterions

We use two standard public datasets, the Weizmann Horse Dataset [18] and cows [58], to perform experimental evaluations for RDTMs. See some examples in figure 4.8. These datasets are designed to evaluate segmentation, so the groundtruth only gives the regions of the object and the background. To supplement this groundtruth, we required students to manually parse the images by locating the positions of leaf nodes of the hierarchy in the images. These parse trees are used as ground truth to evaluate the ability of the RDTM to parse the horses (i.e. to identify different parts of the horse).

To show the generality of our approach, and its ability to deal with different objects without hand-tuning the appearance features, we apply it to the task of face alignment. The dataset [19] contains ground truth of standard 65 key points which lie along the boundaries of face components with semantic meaning, i.e, eyes, nose, mouth and cheek. We use part of this dataset for training (200 images) and part for testing (80 images).

The measure for parsing/alignment. For a given image x, the parsing results are obtained by estimating the configuration y of the RDTM. To evaluate the performance of parsing (for horses) and matching/alignment (for faces) we use the **average position error** measured in terms of pixels. This quantifies the average distance between the positions of leaf nodes of the ground truth and those estimated in the parse tree.

The measure for segmentation. The RDTM does not directly output a full segmentation of the object. Instead the set of leaf nodes gives a sparse estimate for the segmentation. To enable RDTM to give full segmentation we modify it by a strategy inspired by grab-cut [73] and obj-cut [1]. We use a rough estimate of the boundary by sequentially connecting the leaf nodes of the RDTM, to initialize a grab-cut algo-

rithm (recall that standard grab-cut [73] requires human initialization, while obj-cut needs motion cues). We use **segmentation accuracy** to quantify the proportion of the correct pixel labels (object or non-object). Although segmentation accuracy is widely used as a measure for segmentation, it has the disadvantage that it depends on the relative size of the object and the background. For example, you can get 80% segmentation accuracy on the weizmann horse dataset by simply labelling every pixel as background. Therefore, to overcome the shortcoming of segmentation accuracy, we also report **precision/recall**, see [16], where $precision = \frac{P \cap TP}{P}$ and $recall = \frac{P \cap TP}{TP}$ (P is the set of pixels which are classifier as object by RDTM and TP is the set of object pixels in ground truth). We note that segmentation accuracy is commonly used in the computer vision community, while precision/recall is more standard in machine learning.

The measure for detection. We use **detection rate** to quantify the proportion of successful detections. We rate *detection* to be successful if the area of intersection of the labeled object region (obtained by graph-cut initialized by the RDTM) and the true object region is greater than half the area of the union of these regions.

The measure for performance analysis. We judge that an object(or part) is correctly parsed if each subpart (i.e. the location of each node in the hierarchy) is located close (within $k_1 \times l + k_2$ pixels where l is the level with $k_2 = 5$ and $k_1 = 2.5$) to the ground-truth. The thresholds in the distance measure vary proportionally to the height of levels so that the distance is roughly normalized according to the size of object parts. We plot the **precision-recall curve** to study the performance of the components of the whole model.

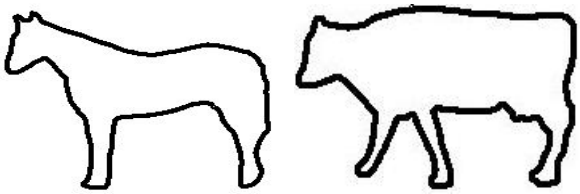


Figure 4.9: This shows the exemplars used for the horse (left) and the cow (right).

Dataset	Size	Detection	Parsing	Segmentation	Speed
Horse	328	86.0	18.7	81.3% /73.4%	3.1s
Cow	111	88.2	–	81.5% /74.3%	3.5s

Table 4.1: The performance of the RDTM with one-example learning.

4.6.2 Experiment I: One-example Learning

We first report the performance of the RDTM with one-example learning. The two exemplars needed to obtain the horse and cow hierarchies are shown in figure (4.9). We use identical parameters for each model (i.e. for the hierarchical aggregation algorithm, for the data terms, and the horizontal and vertical terms, for the proposal thresholds and window sizes).

We illustrate the segmentation and parsing results in figure (4.10). Observe that the algorithm is successful even for large changes in position, orientation and scale – and for object deformations and occlusion. The evaluation results for detection, parsing, and segmentation are shown in table (4.1). Overall, the performance is very good and the average speed is under 4 seconds for an image of 320×240.

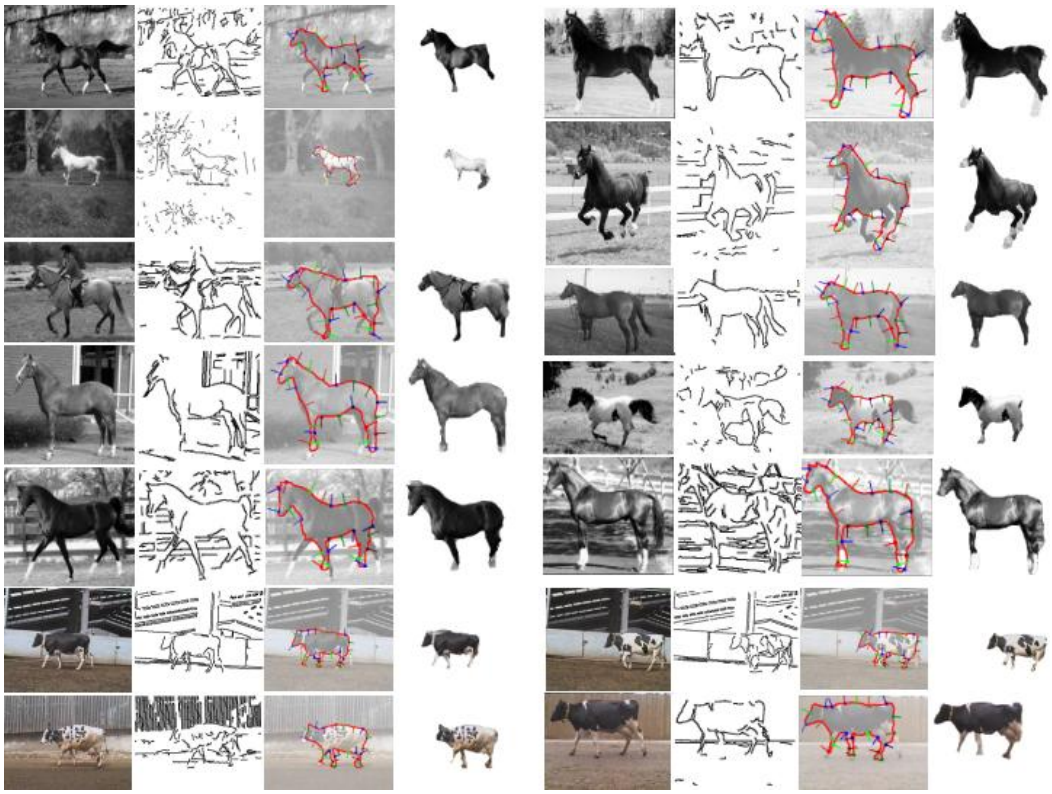


Figure 4.10: Segmentation and parsing results on the horse and cows datasets. The first column shows the raw images. The second one show the edge maps. The third one shows the parsed result. The last one shows the segmentation results.

4.6.3 Experiment II: Contributions of Object Parts: Complexity and Performance Analysis

We use the model learnt by one-example learning to analyze the effectiveness of different components of the hierarchical model in terms of performance and time-complexity. This is shown in table (4.2)and figure (4.11). We hope that this analysis of the tradeoffs between speed and performance will yield general principles for optimal design of modeling and inference for computer vision systems particularly those requiring multi-level processing.

Performance Contributions of Multi-level Object Parts. Figure (4.11) shows how different components of the hierarchy contribute to performance. It is easy to note that smaller object parts have worse performance in terms of precision-recall. More high-level knowledge including both appearance and shape prior makes object parts more distinct from background and thus improves the overall performance. One can see that there is a jump in performance when we move from level 2 to level 3, indicating that the information at level 3 is sufficient to disambiguate the object from cluttered background.

Computational Complexity Analysis. Table (4.2) shows that the number of proposals scales almost linearly with the level in the hierarchy, and the time cost for each level is roughly constant. This demonstrates that the pruning and surround suppression are important factors for making bottom-up processing effective. Overall, this helps understand the effectiveness of the bottom-up processing at different levels.

	Cluster # / Node	Prop. # / Cluster	Time / Node	Time / Image
Level 4	51	38.7	0.14s	0.14s
Level 3	77	76.3	0.29s	0.88s
Level 2	105	37.3	0.21s	1.05s
Level 1	158	9.3	0.10s	1.22s
Level 0	225	5.8	0.01s	0.18s
Hierarchy	180	10.9	0.08s	3.47s

Table 4.2: Analysis of the Bottom-Up Processing. The numbers of clusters for each node and proposals for each cluster at different levels are compared in columns 2 and 3. Time costs for each node and the whole image are listed in the last two columns. The last row shows the numbers averaged over the nodes of all the levels of the hierarchy.

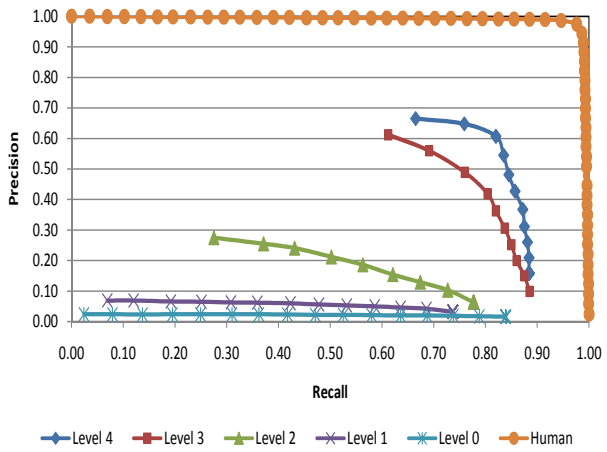


Figure 4.11: This figure shows the Precision-Recall curves for different levels. Level 4 is the top level. Level 0 is the bottom level. "Human" curve is provided as an ideal decision maker for comparison.

Table 4.3: Comparisons of one-example learning and structure-perceptron learning

Approaches	Training / Validation	Det.	Parsing	Segmentation	Speed
One-example	1 / –	86.0 %	18.7	81.3% / 73.4%	3.1s
Structure-perceptron	50 / 50	99.1%	16.04	93.6% / 85.3%	23.1s

Table 4.4: Comparisons of Segmentation Performance on Weizmann Horse Dataset

Methods	Testing	Seg. Accu.	Pre./Rec.
Our approach	228	94.7%	93.6% / 85.3%
Ren [16]	172	91.0%	86.2%/75.0%
Borenstein [74]	328	93.0%	
LOCUS [67]	200	93.1%	
Cour [68]	328	94.2%	
Levin [4]	N/A	95.0%	
OBJ CUT [1]	5	96.0%	

4.6.4 Experiment III: Evaluations of Structure-Perceptron Learning for Deformable Object Detection, Segmentation and Parsing

In this experiment, we will apply structure-perceptron learning to include all image features for the leaf nodes and non-leaf nodes, and estimate the parameters α. The hierarchical structure is obtained by one-example learning. We use the Weizeman horse dataset [18] for evaluations where a total of 328 images are divided into three subsets – 50 for training, 50 for validation, and 228 for testing. The parameters learnt from the training set, and with the best performance on validation set, are selected.

Results. The best parse tree is obtained by performing inference algorithm over RDTM learnt by structure-percepton learning. Figure 4.12 shows several parsing and segmentation results. The states of the leaf nodes of parse tree indicate the positions of the points along the boundary which are represented as colored dots. The points of same color in different images correspond to the same semantic part. One can see our model's ability to deal with shape variations, background noise, textured patterns, and changes in viewing angles. The performance of detection and parsing on this dataset is given in Table 4.3. Structure-perceptron learning which include more visual cues outperforms one-example learning in all tasks. The localization rate is around 99%. Our model performs well on the parsing task since the average position error is only 16 pixels (to give context, the radius of the color circle in figure 4.12 is 5 pixels). Note no other papers report parsing performance on this dataset since most (if not all) methods do not estimate the positions of different parts of the horse (or even represent them). The time of inference for image with typical size 320×240 is 23 seconds.

Comparisons. In table 4.4, we compare the segmentation performance of our approach with other successful methods. Note that the object cut method [1] was reported on only 5 images. Levin and Weiss [4] make the strong assumption that the position of the object is given (other methods do not make this assumption) and

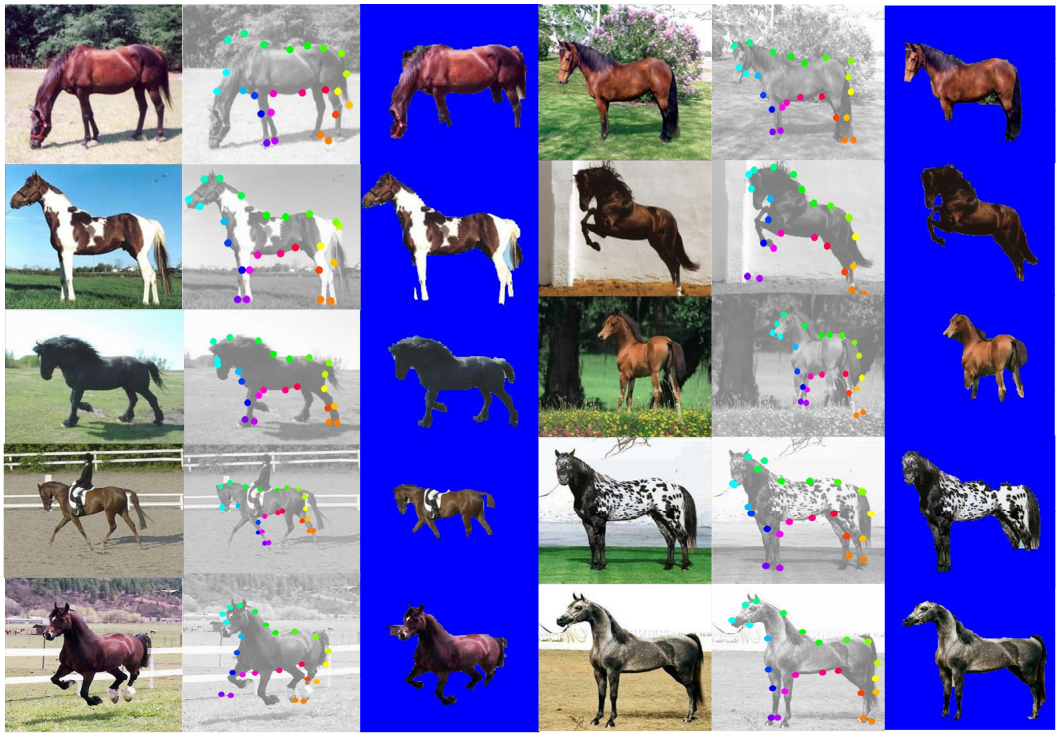


Figure 4.12: Examples of Parsing and Segmentation. Column 1 , 2 and 3 show the raw images, parsing and segmentation results respectively. Column 4 to 6 show extra examples. Parsing is illustrated by dotted points which indicate the positions of leaf nodes (object parts). Note that the points in different images with the same color correspond to the same semantical part.

not report how many images they tested on. Overall, Cour and Shi's method [68] was the best one evaluated on large dataset. But their result is obtained by manually selecting the best among top 10 results (other methods output a single result). By contrast, our approach outputs a single parse only but yields a higher pixel accuracy of 94.7%. Hence we conclude that our approach outperforms those alternatives which have been evaluated on this dataset. As described above, we prefer the precision/recall criteria [16] because the segmentation accuracy is not very distinguishable (i.e. the baseline starts at 80% accuracy, obtained by simply classifying every image pixel as

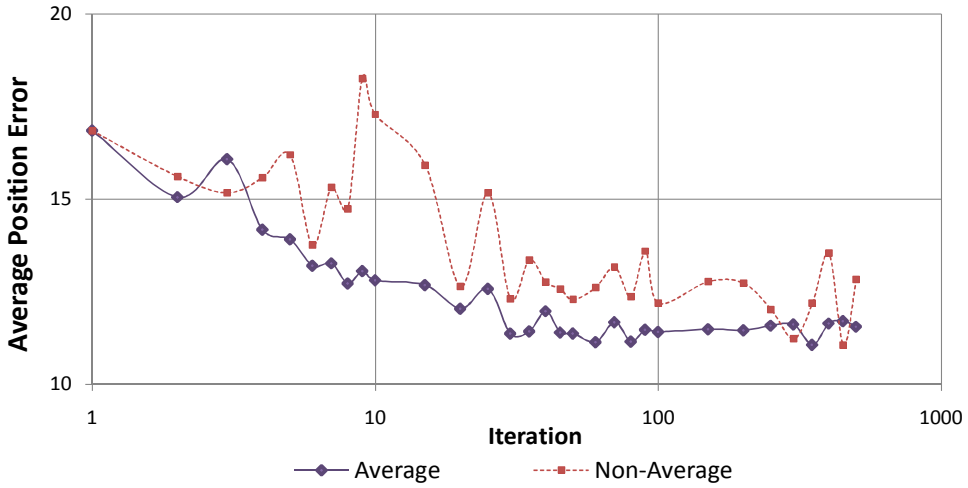


Figure 4.13: The average position errors (y-axis) across iterations (x-axis) are compared between Algorithm-II(average) and Algorithm-I (non-average).

being background). Our algorithm outperforms the only other method evaluated in this way (i.e. Ren et. al's [16]). For comparison, we translate Ren et. al's performance (86.2%/75.0%) into segmentation accuracy of 91% (note that it is impossible to translate segmentation accuracy back into precesion/recall).

4.6.5 Experiment IV: Diagnosis of structure-perceptron learning

In this section, we will conduct diagnosis experiments to study the behavior of structure-perceptron learning.

Convergence Analysis. Figure 4.13 shows the average position error on training set for both Algorithm II (averaged) and Algorithm I (non-averaged). It shows that the averaged algorithm converges much more stablely than non-averaged algorithm.

Generalization Analysis. Figure 4.14 shows average position error on training, validation and testing set over a number of training iterations. Observe that the behavior on the validation set and the testing set are quite similar. This confirms that the

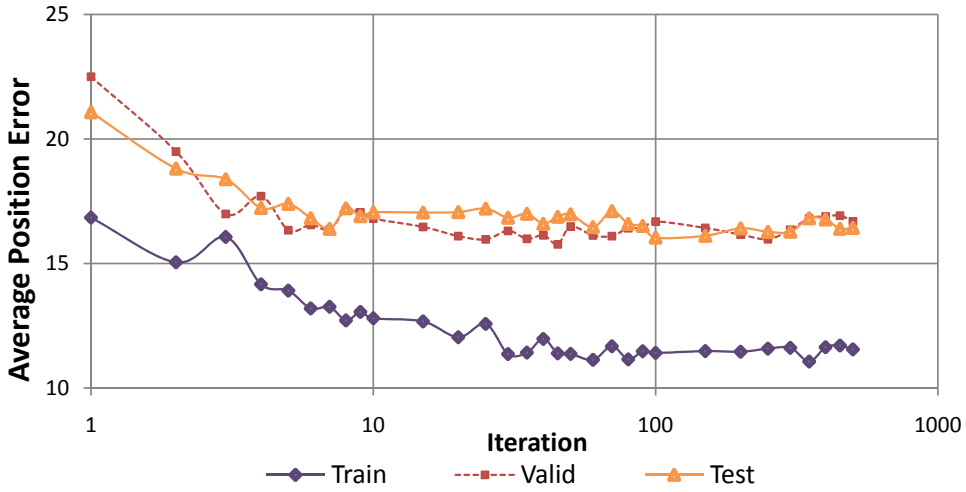


Figure 4.14: The average positions errors on training, validation and testing dataset are reported.

selection of parameters decided by the validation set is reasonable.

Feature Selection. We show the features learnt from structure-perceptron learning in figure (4.15) (features are shown at the bottom level only for reasons of space). The top 5 features, ranked according to their weights, are listed. The top left, top right and bottom left panels show the top 5 features for all leaf nodes, the node at the back of horse and the node at the neck respectively. Recall that structure-perceptron learning performs feature selection by adjusting the weights of the features.

4.6.6 Experiment V: Multi-view Face Alignment

The task of multi-view face alignment has been much more thoroughly studied than horse parsing. Our RDTM approach, using identical settings for horse parsing, *achieves an average distance error of* 6.0 *pixels, comparable with the best result* 5.7 *pixels*, obtained by [59]. Their approach is based mainly on the Active Appearance Models [60] which were motivated specifically to model faces and which assume that the shape de-

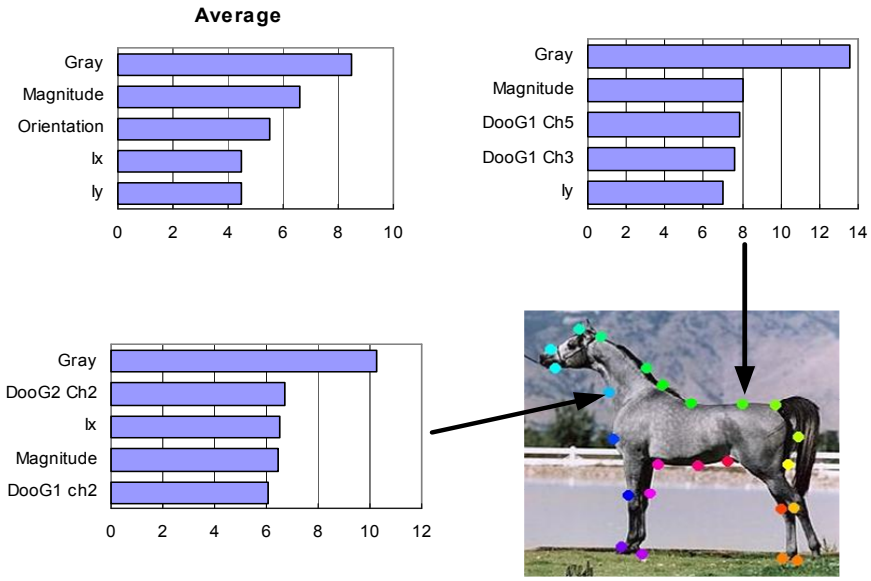


Figure 4.15: Weights of Features. The most useful features overall are gray value, magnitude and orientation of gradient, and difference of intensity along horizontal and vertical directions (Ix and Iy). DooG1 Ch5 means Difference of offset Gaussian (DooG) at scale 1 (13*13) and channel (orientation) 5 ($\frac{4}{6}\pi$).

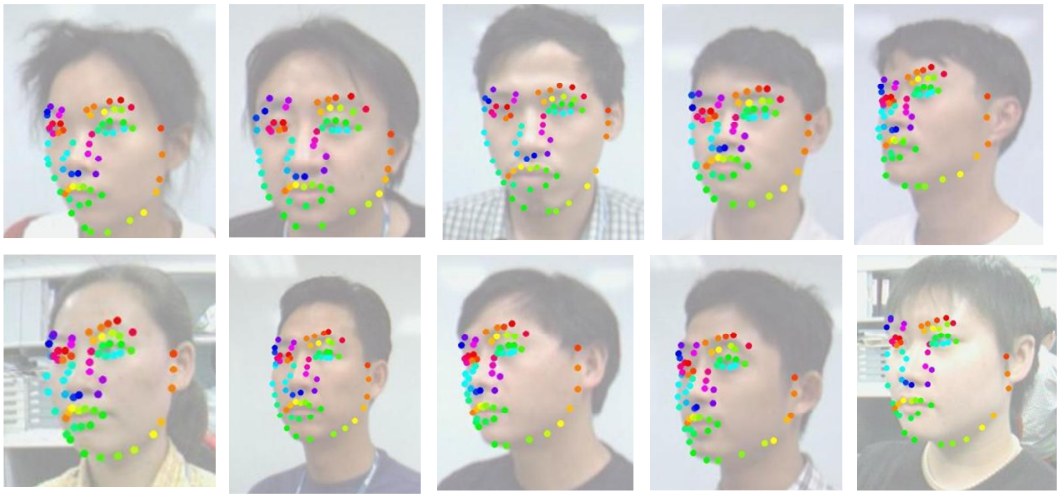


Figure 4.16: Multi-view Face Alignment.

formations are mostly rigid. By contrast, our RDTMs are suitable for both rigid and deformable objects and required no special training or tuning to apply to this problem. Figure 4.16 shows the typical parse results for face alignment.

4.7 Conclusion

We developed a Recursive Deformable Template Model (RDTM) for representing objects which can be learnt by adapting the structure-perceptron algorithm used in machine learning. Advantages of our approach include the ability to select shape and appearance features at a variety of scales in an automatic manner.

We demonstrated the effectiveness and versatility of our approach by applying it to very different problems, evaluating it on large datasets, and giving comparisons to the state of the art. Firstly, we showed that the RDTM outperformed other approaches when evaluated for segmentation on the weizmann horse dataset. It also gave good results for parsing horses (where we supplied the groundtruth), though there are no other parsing results reported for this dataset. Secondly, we applied RDTMs to the

completely different task of multi-view face alignment (without any parameter tuning or selection of features) and obtained results very close to the state of the art.

However, the structure of the RDTM has to require the object template to be input by human. In the next chapter, we will discuss how RDTM can be learnt in an unsupervised way.

CHAPTER 5

Unsupervised Learning: Recursive Composition, Suspicious Coincidence and Competitive Exclusion

In chapter 4, we have discussed how to learn a RDTM in a supervised manner (i.e. structure-perceptron). In this chapter, we will address the issues of unsupervised learning. We describe a new method for unsupervised structure learning of a *Recursive Deformable Template* model (RDTM) for deformable objects. The learning is unsupervised in the sense that we are given a training dataset of images containing the object in cluttered backgrounds but we do not know the position or boundary of the object. The structure learning is performed by a bottom-up and top-down process. The bottom-up process is a novel form of hierarchical clustering which recursively composes proposals for simple structures to generate proposals for more complex structures. We combine standard clustering with the *suspicious coincidence principle* and the *competitive exclusion principle* to prune the number of proposals to a practical number and avoid an exponential explosion of possible structures. The hierarchical clustering stops automatically, when it fails to generate new proposals, and outputs a proposal for the object model. The top-down process validates the proposals and fills in missing elements. We tested our approach by using it to learn a hierarchical compositional model for parsing and segmenting horses on Weizmann dataset. We show that the resulting model is comparable with (or better than) alternative methods. The versatility of our approach is demonstrated by learning models for other objects (e.g.,

faces, pianos, butterflies, monitors, etc.). It is worth noting that the low-levels of the object hierarchies automatically learn generic image features while the higher levels learn object specific features.

5.1 Introduction

The goal of this chapter is to learn a hierarchical model for deformable objects. The learning is unsupervised in the sense that we are given a training dataset of images containing the object in cluttered backgrounds but we do not know the position or boundary of the object. Unsupervised learning is desirable since it avoids the need for time consuming hand labeling and prevents implicit biases. We apply the model to tasks such as object segmentation and parsing (matching of object parts).

Learning a hierarchical compositional model is very challenging since it requires us to learn the structure of the model (e.g. the relationships between the variables, and the existence of hidden variables) in addition to the parameters of the model. The difficulties are visually illustrated in figure 5.1: (i) the objects are deformable so there is ambiguity in the structure. (ii) there is cluttered background noise. (iii) parts of the object may be missing, (iv) the input is simple oriented edge features, which is a simple and highly ambiguous representation.

We now discuss some critical computational issues for unsupervised learning which motivate our hierarchical approach and contrast it to other unsupervised approaches for learning object models (e.g. [32], [9]). An abstraction of the structure learning problem is that we have a dataset of images each of which contain approximately M object features and N total features. In this chapter, we are considering the case of $M = 100$ and $N = 5000$. Learning an object model requires solving a complicated correspondence problem to determine which features are object, which are background, and the

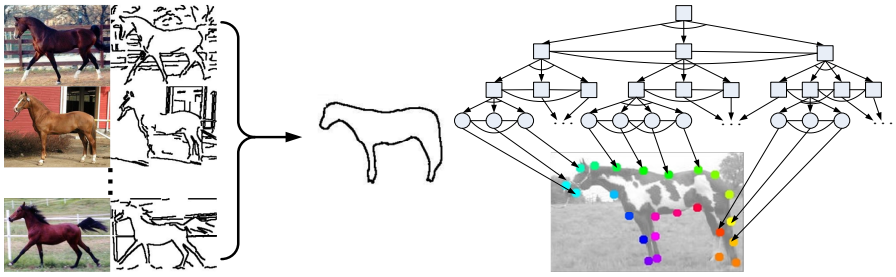


Figure 5.1: Left:The learning is unsupervised in the sense that we are given a training dataset of images containing the object in cluttered backgrounds but we do not know the position or boundary of the object. Right: The hierarchical representation of the object. The boxes represent non-leaf nodes. The circles denote leaf nodes that directly relate to properties of the input image. The topology of the structure is not given, but learnt in an unsupervised way.

spatial relationships between the object features. There are several strategies for addressing this correspondence problem. The first naive strategy is brute force enumeration which involves testing all N^M possibilities. This is only practical if M and N are small and the appearances of the features are sufficiently distinctive to enable many possible correspondences to be rejected. The second strategy is to learn the model in a greedy (non-hierarchical) way by sequentially growing subparts one by one [32, 9]. This is practical in cases where brute force fails, but it still makes two assumptions: 1) sparseness assumption which requires M and N to be fairly small, e.g., $M = 6$ and $N = 100$ in [32] and 2) small ambiguity assumption which requires the appearances of the features to be somewhat distinctive. Neither of these two strategies are applicable if the features are edgelets, because M and N are both large and all edgelets have the same appearance which leads to big ambiguity. (One strategy is to use more powerful features, but we argue that this merely postpones the complexity problem). The purpose of this chapter is to develop a more general unsupervised learning method without making strong assumptions of sparseness and low ambiguity. In other words, we try

to provide a unified learning scheme for both generic features and object structures. Hence, we are driven to a third strategy, named as Recursive Composition, that creates a model by combining elementary structures to build a hierarchy.

Our strategy is based on a novel form of bottom-up recursive clustering. We assume that the object can be expressed in terms of recursive compositions of elementary structures (see the right panel of figure 5.1). We learn the structure by repeatedly combining proposals for substructures to make proposals for more complex structures. This process stops when we cannot generate any more proposals. A huge number of proposals are examined and stored at every level to avoid ambiguity since small substructures are not necessarily distinct enough between object and background. However, combining proposals in this way risks a combinatorial explosion (imagine that the number of combinations will grow exponentially as we go up to the upper levels). We avoid this by the use of two principles: (i) suspicious coincidences, and (ii) competitive exclusion. Suspicious coincidences eliminates proposals which occur infrequently in the image dataset (in less than 90 percent of the images). The competitive exclusion principle is adapted from ecology community where it prevents two animals from sharing the same environmental niche. In this chapter, competitive exclusion eliminates proposals which seek to explain overlapping parts of the image.

The bottom-up clustering is followed by a top-down stage which refines and fills in gaps in the hierarchy. These gaps can occur for two reasons. Firstly, the competitive exclusion principle sometimes eliminates subparts of the hierarchy because of small overlaps. Secondly, gaps may occur at low-levels of the hierarchy, for example at the neck of a horse, because there are considerable local shape variations and so suspicious coincidences are not found. The top-down process can remove these gaps automatically by relaxing the two principles. For example, at higher levels the system discovers more regular relationships between the head and torso of the horse which

provides context enabling the algorithm to fill in the gaps.

In summary, hierarchical bottom-up and top-down procedure allows us to grow the structure exponentially (the height of a hierarchy is $\log(M)$) and end up with a nearly complete structure (the resulting representation is dense and the size M could be big). We tested our approach by using it to learn a hierarchical compositional model for parsing and segmenting horses on Weizmann dataset [18]. We show that the resulting model is comparable with (or better than) alternative methods. The versatility of our approach is demonstrated by learning models for other objects (e.g., faces, pianos, butterflies, monitors, etc.).

5.2 Background: Hierarchical Structure Learning

Hierarchical design dates back to Fukushima's Neocognitron [61]. Recently, there have been many new developments including new representations and learning algorithms. For example, Geman et al. [49] propose a hierarchical object model designed by a compositional principle using an AND/OR graph. Inference algorithms have been invented for this type of model [11], which we will adapt and use to perform inference in this chapter (Note [11] does not address any learning algorithm). Deep networks [75], another type of multi-layer system, has recently been proposed by Hinton et al. [75]. Ullman et al. [76] learns a hierarchical feature representation in a supervised manner. Poggio and his colleagues'[63] build a hierarchical structure for rapid object recognition motivated by mimicing the architecture of the visual cortex of the human brain. Ahuja and Todorovic's hierarchical representation [77] is based on segmented image regions. Fleuret and Geman [78] provide a coarse-to-fine strategy which starts from edgelets. Fidler and Leonardis[79]'s unsupervised learning approach is most related to our work (the main difference is that they treat rigid objects, have a less dense representation, and do not have a top-down stage). In summary, in several cases these

models[49, 11, 63] are not learnt but are specified by the user. Some unsupervised methods [76] assume that background is clean and the object is roughly aligned. Unsupervised learning (in the sense of this chapter) has been performed [32, 11] (without a hierarchy) but for object models with limited number of object features (as discussed earlier). Most models [32, 63, 76, 49, 9] focus on recognition or categorization, but not parsing.

5.3 The Recursive Deformable Template Model (RDTM) and the Inference algorithm

5.3.1 The Recursive Deformable Template Model (RDTM)

In this subsection, we introduce the Recursive Deformable Template Model which is slightly different from the one in chapter 4. Recall that RDTM represents objects in a hierarchical form. The object template consists of a small number of small sub-templates which is composed by smaller subsub-templates, and so on. But in this chapter, RDTM has no interactions between non-leaf nodes and image features. The graph structure for the hierarchical model is depicted in the right panel of figure 5.1 and defined as follows. We let V_r be the set of nodes of the graph with root node r. Each node $\nu \in V_r$ has a set of child nodes T_ν (a node is constrained to have a single parent), V_r^{LEAF} are the leaf nodes (the only ones which interact with the image). For any node $\nu \in V_r$, we can define a graph V_ν with root node ν containing the descendants of ν. The edges for the graph are of three types: (i) vertical *data edges* which relate the leaf nodes to the image, (ii) horizontal edges relating the children of each node to each other (described below), and (iii) vertical edges relating parents to children (specified by $\{T_\nu\}$). In this chapter, all vertical edges are directed. The horizontal edges only connect the child nodes with the same parent. Moreover, we restrict the number of

child nodes to be less than six, i.e., $|T_\nu| \leq 6$. This ensures that the size of the biggest clique of the hierarchy is small. We use the notation Ω_r to represent the graph (i.e. the set of nodes V_r and the set of edges).

A configuration of the graph is an assignment of state variables $z = \{z_\nu\}$ to all $\nu \in V_r$. In this chapter, we set $z_\nu = (x_\nu, y_\nu, \theta_\nu, s_\nu)$, where (x, y), θ and s denote image position, orientation, and scale respectively (the scale is the area of the image that the node represents). We use the notation $Z_\nu = \{z_\mu : \mu \in V_\nu\}$ to denote the state of node ν and all its descendent nodes.

We define a Gibbs distribution $P(z, d; \omega, \Omega)$ for the probability of the graph as follows:

$$P(z, d; \omega, \Omega) = \frac{1}{Z(\omega, \Omega)} \exp\{-E(z, d; \omega, \Omega)\}. \tag{5.1}$$

where $Z(\omega, \Omega)$ is the partition function, d is the image, ω denotes the parameters of the distribution and Ω denotes the graph structure (i.e. the nodes and the edges). The energy function $E(z, d; \omega, \Omega)$ is the sum of three terms corresponding to the three types of edges in the graph:

$$E_d(d, Z_r; \omega, \Omega) + E_h(Z_r; \omega, \Omega) + E_v(Z_r; \omega, \Omega), \tag{5.2}$$

where E_d, E_h, E_v are energy terms defined at the data, horizontal, and vertical edges. We now describe the specific choices used in this chapter.

The *data term* E_d is given by:

$$E_d(d, Z_r; \omega, \Omega) = \sum_{\nu \in V_r^{LEAF}} f(d_\nu, z_\nu), \tag{5.3}$$

where V_r^{LEAF} is the set of the leaf nodes and $f(.,.)$ is the (negative) logarithm of a Gaussian defined over the grey-scale intensity gradient (i.e. $d_\nu = \vec{\nabla} I_\nu$). It biases the leaf nodes to be located at image locations where the image gradient is large, and for

their orientations to be perpendicular to the image gradient. This term is fixed and not learnt.

The *horizontal term* E_h is given by:

$$E_h(Z_r; \omega, \Omega) = \sum_{\nu \in V_r / V_r^{LEAF}} \sum_{(\mu,\rho,\tau):\mu,\rho,\tau \in T_\nu} g(z_\mu, z_\rho, z_\tau; \omega), \tag{5.4}$$

where $g(z_\mu, z_\rho, z_\tau; \omega)$ is the (negative) logarithm of Gaussian distribution defined on the *invariant triplet vector* (ITV) $l(z_\mu, z_\rho, z_\tau)$ constructed from (z_μ, z_ρ, z_τ) [9]. (The ITV is invariant to the translation, rotation, and scaling of the triple, which ensures that the full probability distribution is also invariant to these transformations). The summation is over all triples formed by the child nodes of each parent, see the right panel of figure (5.1). The parameters of the Gaussian are indicated by $\omega = (\mu, \sigma)$.

The *vertical term* E_v is used to hold the structure together by relating the state of the parent nodes to the state of their children. It is defined by:

$$E_v(Z_r; \omega, \Omega) = \sum_{\nu \in V_r / V_r^{LEAF}} h(z_\nu; \{z_\mu \; s.t. \mu \in T_\nu\}), \tag{5.5}$$

where $h(.,.) = 0$ provided the average orientations and positions of the child nodes are equal to the orientation and position of the parent node – formally $(x_\nu, y_\nu, \theta_\nu) = (1/|T_\nu|) \sum_{\mu \in T_\nu} (x_\mu, y_\mu, \theta_\mu)$. If this equality does not hold, then $h(.,.) = \kappa$, where κ is a large positive number. The scale variable $s_\nu = \sum_{\mu \in T_\nu} s_\mu$, hence the parent node represents the sum of the regions in the image represented by its children.

Observe that the nodes of the graph have the same variables at all levels (i.e. (x, y, θ, s)). This makes use of hierarchical independence assumption–the higher level interactions depend only on the summarization of all child nodes at the lower levels. More precisely, the child nodes only propagate a limited summary (center position, total size and orientation) of their state information to their parents. The choice of the vertical term E_v means that this information is simply the average of their node variables (except for size).

We stress that we can use equations (5.2,5.3,5.4,5.5) to calculate the energy for any subgraph by specifying the root node of the subgraph. This gives an effective way for computing the full energy, by combining the energy of the subgraphs, and is exploited during inference and learning. We can also exploit this hierarchy in order to compute other important properties, such as the partition function $Z(\omega, \Omega)$. Note, in figure 5.1, the vertical edges are directed and horizontal edges only connect the child nodes (less than six) with the same parent. Therefore, the partition function can be factorized into several components (corresponding to the cliques) whose sizes are small.

5.3.2 The Inference Algorithm for parsing when RDTM is known

The inference algorithm for the RDTM has been introduced in chapter 4 (see section 4.4). This algorithm will be used both when we are learning the model, see section (5.4), and when we are applying the model to new images to perform detection, segmentation, and parsing.

5.4 Unsupervised Structure Learning

We now address the critical issue of how to learn the structure of the hierarchical model from a set of images which contain an example of the object with varying background. Formally, this requires us to estimate the structure parameters Ω and the distribution parameters ω. The task of the unsupervised learning is defined as :

$$(\omega^*, \Omega^*) = \arg\max P(\omega, \Omega | d) \tag{5.6}$$

$$= \arg\max P(d|\omega, \Omega) P(\omega, \Omega) \tag{5.7}$$

$$= \arg\max \sum_z P(d|z, \omega) P(z|\omega, \Omega) P(\omega, \Omega) \tag{5.8}$$

where $P(d|z,\omega)P(z|\omega,\Omega) = P(d,z|\omega,\Omega)$ is defined in equation (5.1). $P(\omega,\Omega)$ accounts for the prior distribution of the structure which plays a similar role of structure regularization. $P(\omega,\Omega)$ is factorized into $P(\omega)$ and $P(\Omega)$. The parameters indicated by ω are μ and σ in the gaussian functions. The prior on σ is of the form $\mathcal{N}(0,\beta I)$. The prior on μ^l at certain level l puts hard constraints on the range of the distance allowed between any two substructures. The prior of Ω is uniform distribution. The summation $\sum_z P(d|z,\omega)P(z|\omega,\Omega)P(\omega,\Omega)$ is used as a *score* function to measure the goodness of the fit of the structure. Intuitively, the score function tells us how frequently a structure encoded by (Ω,ω) appears in the training set. Our approach is based on a bottom-up process to propose a set of structures (Ω,ω) followed by a top-down process which refines the result by adjusting (Ω,ω).

Our algorithm makes several approximations to simplify the learning problem. Firstly, we will exploit the hierarchical nature of the model to estimate the parameters ω locally. In principle, we could use these local estimates to initialize an EM algorithm to determine the parameters globally. Secondly, while our algorithm proposes a structure Ω we cannot guarantee that it is the globally optimal structure. But our experimental results, see section (5.5), provide empirical evidence that these approximations are reasonable.

5.4.1 The Bottom-Up Process

The bottom-up process constructs hierarchical object models by composing them from more elementary components. We use two principles to prevent a combinatorial explosion of compositions and to ensure that our compositions result in desirable object models. The two principles are: (i) *suspicious coincidences* where we keep compositions which occur frequently in the images and reject compositions which do not, and (ii) *competitive exclusion* where we remove compositions which match to regions of

the image which overlap with other compositions. The relative importance of these different principles is shown empirically in the results section, see table 5.3 and figure 5.6. For example, observe how competitive exclusion play a small role when the compositions are small but is of major importance as the compositions get large.

Our strategy proceeds by creating vocabularies of concepts at different levels in the hierarchy, where each concept is a hierarchical compositional model. See figure 5.2 for visual (symbolic) illustration. The concepts are generated by composing concepts at lower levels. The basic ideas are to detect all instances of the concepts at level l in the images (using the inference algorithm for each concept). We *form compositions* of these concepts by identifying sets of these concepts that appear in sub-regions of the images. We *cluster* these compositions based on the spatial relationships between their elements (the instances of the concepts at level $l-1$) to get a set of concepts at level l whose distribution parameters ω are estimated from the clusters. We run the inference algorithm on the images to find the instances of all the concepts. Then we use the *suspicious coincidence principle* to remove concepts which occur infrequently (i.e. have few instances). Next we use the *competitive exclusion principle* to remove concepts whose instances overlap with those of other concepts (with better scores). The remaining concepts form the vocabulary of concepts at level $l+1$. The process repeats until no new concepts are composed. For our applications (e.g. images containing a large object with variable background), it will terminate automatically in a small number of proposals when the hierarchy has reached a maximum size (i.e. there is no larger structure to be found).

The full procedure is described more formally in the next few paragraphs and is summarized by the pseudo-code in figure (5.3) and illustrated visually in figure 5.2. First we introduce some notation (see table 5.1)). A *concept* at level l is notated by $P^l_{\omega,\nu}$. It is a RDTM with root node ν and ω represents the parameters of this model. A

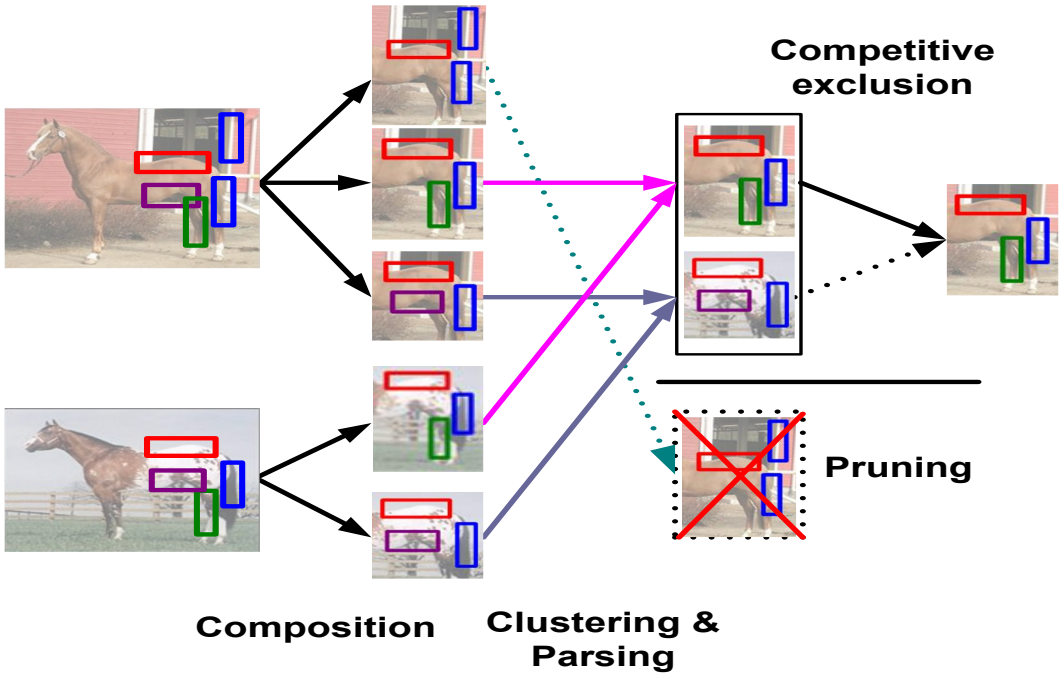


Figure 5.2: This figure illustrates the procedure of the unsupervised learning. The first column shows the responses of the features from the "vocabulary" at level l. the second column shows the compositions at step 1). the third column shows the clustering at step 2. The noise(non-regular) pattern bounded by dotted line is pruned out by step 4. The last column plots the results after step 5. At step 5, the compositions, which are constructed by different components, but parse the same areas in the image, are grouped together (the maximum is kept

concept	$P^l_{\omega,\nu}$
max concept	$MP^l_{\omega,\nu}$
instance	P^l_{ν}
max instance	$MP^l_{\nu,a}$

Table 5.1: The table shows the notation for the learning algorithm.

vocabulary of concepts at level l is the set $\{P^l_{\omega,\nu}\}$. We can parse the training dataset using the vocabulary of concepts at level l to get a set of *instances* of each concept $\{MP^l_{\nu,a}\}$, where $a = 1, ..., N^l_\nu$ indexes the set of instances. Each instance is represented by its state variables Z_ν.

We define the bottom-up process as follows. At each level $l-1$ we have a vocabulary of concepts $\{P^{l-1}_{\omega,\nu}\}$ and a set of instances of the concepts in the images $\{MP^{l-1}_{\nu,a}\}$. We then repeat the following steps.

1. Composition. We search through the images to find instances of triplets of concepts $MP^{l-1}_{\mu,\omega}, MP^{l-1}_{\rho,\omega}, MP^{l-1}_{\tau,\omega}$ within subregions of size S_{l-1} (for all μ, ρ, τ in the vocabulary). From each triplet instance we construct an instance P^l_ν where node ν has children $\mu, \rho, \tau \in T_\nu$.

2. Clustering. We cluster the instances P^l_ν by the second order moments of the shape descriptor (triplet vector) of the positions, scales and orientation $(x_\rho, y_\rho, \theta_\rho, s_\rho)$ of the children of ν (this clustering is done separately for all triples in the concept vocabulary at level $l-1$). The clustering outputs a set of *concepts* at level l notated by $\{P^l_{\omega,\nu}\}$. Some of the parameters $\omega = (\mu, \sigma)$ of each concept are estimated from the second order moments (see above) and the remainder are inherited from the underlying concepts at level $l-1$.

3. Parsing. We parse the image dataset using the inference algorithm (see section (5.3.2)) for the set of concepts output by the clustering $\{P^l_{\omega,\nu}\}$. This gives a set of instances of the concepts in the dataset $\{MP^l_{\nu,a}\}$.

4. Pruning by Suspicious Coincidences. We remove concepts from $\{P^l_{\omega,\nu}\}$ if they do not have sufficient number of instances in the dataset (i.e. if there are instances of the concept in less than ninety percent of the images).

5. Competitive exclusion. We remove concepts from $\{P^l_{\omega,\nu}\}$ if their instances have significant overlap with instances of other concepts (and the other concepts have better

scores).

The remaining concepts form the concept vocabulary $\{P^l_\omega\}$ at level l. Their instances in the dataset $\{MP^l\}$ have already been calculated (in the parsing step above). The process repeats itself until no new concepts are generated. In practice, this happens within 4-5 levels (see experimental section). The process is initialized for the leaf nodes. The vocabulary of concepts at level 1 $\{P^1_{\omega,\nu} : \nu = 1, ..., 4\}$ consists of four types of edgelets characterized by their orientations $\theta = m\pi/4$ for $m = 0, 1, 2, 3$. The size s is set to a default value s_1. We parse the training set of images to detect the instances of all the concepts (more precisely, we compute the set of points (x, y) : $E_d(\vec{\nabla} I(x, y), (x, y, m\pi/4, s_1)) < T_1$, where T_1 is a threshold, for $m = 0, 1, 2, 3$). This gives the set of instances of each concept $\{P^1_{\omega,\nu}\}$.

5.4.2 The Top-Down Process

The top-down process fills in the missing parts of the hierarchy and also adds a dense representation at the lowest level (to enable segmentation of the image). Missing parts of the hierarchy can occur for two reasons: (i) competitive exclusion may be too strict at eliminating proposals which only slightly overlap, (ii) shape variations at certain parts of the object may be big at small scales (hence rejected by suspicious coincidences) but are more regular at larger scales. For example, there are big variations locally where the legs of a horse join the torso, which make it hard to detect small scale regularities by clustering. But there are large scale regularities between the legs and the torso which are found at higher levels in the bottom-up process. The top-down process can fill in the gaps at lower level by using the higher levels connections as "context" which allow the pruning criteria to be relaxed.

A greedy strategy is used to examine every node in the hierarchy. For each node ν^l at level l, we seek to add a substructure from the dictionary (the set of proposals

Input: $\{MP^1\}$. Output: $\{{MP_\omega}^L\}$. $\oplus$ denotes the operation of combining two proposals.

- Bottom-Up-Structure-Construction(MP^1)

  Loop : $l = 1$ to L

  1. Composition: $\{P_\nu^l\} = \oplus(MP_{\mu,a}^{l-1}, MP_{\rho,b}^{l-1}, MP_{\tau,c}^{l-1})$ and $T_\nu^l = \{\mu^{l-1}, \rho^{l-1}, \tau^{l-1}\}$
  2. Clustering: $\{{P_{\omega\nu}}^l\} = Clustering(\{P_\nu^l\})$
  3. Parsing: for each ${P_{\omega\nu}}^l$, $\{MP_{\nu,a}^l\} = Parser({P_{\omega\nu}}^l, \{MP_\mu^{l-1}, MP_\rho^{l-1}, MP_\tau^{l-1}\})$
  4. Suspicious Coincidence (Pruning) : $\{{P_{\omega\nu}}^l\}\{P\omega_\nu^l | Score(P\omega_\nu^l) > T_{pruning}\}$
  5. Competitive Exclusion: $\{({MP_{\omega\nu}}^l, {CL_{\omega\nu}}^l)\} = CompetitiveExclusion(\{{P_{\omega\nu}}^l\}, \epsilon_{region})$ where ϵ_{region} is the size of the window W_{image} defined in regions which are covered by a set of images.

- $\tilde{P}_\omega = \arg\max_\nu Score({MP_{\omega\nu}}^L)$

- Top-Down-Structure-Extension

  Loop: $l = L$ to 2, for each node ν at level l within $\tilde{P}_\omega$

  – repeat

  1. $P_{\omega\rho}^* = \arg\max_{{P_{\omega\rho}}^{l-1}} Score(\tilde{P}_\omega \oplus {P_{\omega\rho}}^{l-1})$
  2. $\Delta = Score(\tilde{P}_\omega \oplus P_{\omega\rho}^*) - Score(\tilde{P}_\omega)$
  3. $\tilde{P}_\omega = \tilde{P}_\omega \oplus P_{\omega\rho}^*; T_\nu^l = T_\nu^l \cup \rho^{l-1}$

  until $\Delta < T_{extension}$

Figure 5.3: Bottom-up and Top-down Learning.

obtained in the bottom-up processing) at the lower level $l-1$ as the new child of ν^l so that the extended structure fits the data best (locally). This extension operation for the node ν^l is repeated until the gain is less than some threshold. The extensions correspond to adding more energy terms defined in equation (5.4) and (5.5). In figure (5.5), one can observe that substructures are added into the hierarchy to capture the details of the head and hind leg.

5.5 Experimental Results

Performance Comparisons for segmentation and parsing. We applied our approach to learn hierarchical composition models for a number of different objects. We will first concentrate on the hierarchical model for the horse, learnt from data from the Weizmann database [18] which is divided to training (12 images without labeling) and testing (316 images with groundtruth) sets. These images cover many poses (standing, running, drinking, etc.), changes in viewing angles, different scales, textured body patterns and cluttered background. Our strategy to obtain segmentation, which is inspired by Grab-Cut [73] , is to obtain the parse by the inference algorithm on the RDTMs and then segment object by graph-cut using the feature statistics inside the boundary as initializations (note that, unlike us, Grab-Cut requires initialization by a human). The comparisons using segmentation accuracy are shown in table 5.2. The methods in [16, 4] are based on supervised learning. Only two methods [1, 4] reported higher accuracy than ours: but (i) Obj Cut[1] is tested on only five images while our method is tested on more than 300 images; (ii) [4] assumes the position of object is given for both learning and testing (Note our method does not make this assumption). [11] manually designed the model (no learning) whose performance, i.e., about 91%, is similar to [16] and inferior to ours. [68] is not listed because they manually select the best among top results (all other methods output single result). In conclusion, our method achieves

Method	Train	Test	Segmentation	Speed
Our method	12	316	93.3	16.9s
Ren [16]	172	172	91.0	–
Borenstein [74]	64	328	93.0	–
LOCUS [67]	20	200	93.1	–
OBJ CUT [1]	N/A	5	96.0	–
Levin [4]	N/A	N/A	95.0	–

Table 5.2: Results on horse dataset. Columns 2 and 3 give the size of images taken for training and testing. Column 4 quantifies the segmentation accuracy. The last column shows the average time taken for one image. Note that [1] is tested on only 5 images and [4] assumes the position of the object is roughly given.

the comparable (if not the best) performance even though we use far less information (we only know that the object is present somewhere in the image, while the supervised methods know the precise location of the boundaries). In addition, our model simultaneously performs other tasks such as detection and parsing (labeling different parts of the horse). See figure 5.4 for the typical segmentation and parsing results (we also cropped the results from [1] for comparison). Note that the color points indicate identities of object parts. Our method obtains more complete and stable boundaries than [1] which learns object models from video where extra motion cues are used to make learning easier. The average time of our inference algorithm including both parsing and segmentation is 17 seconds.

Study the Hierarchy. The final hierarchy for horse is depicted in the left panel of figure 5.5. The mean position and orientation of edgelets of leaf nodes are depicted to sketch the model learnt by the unsupervised method. The colored rectangles in the dotted box highlight the parts filled in at the top-down stage. Note that the final

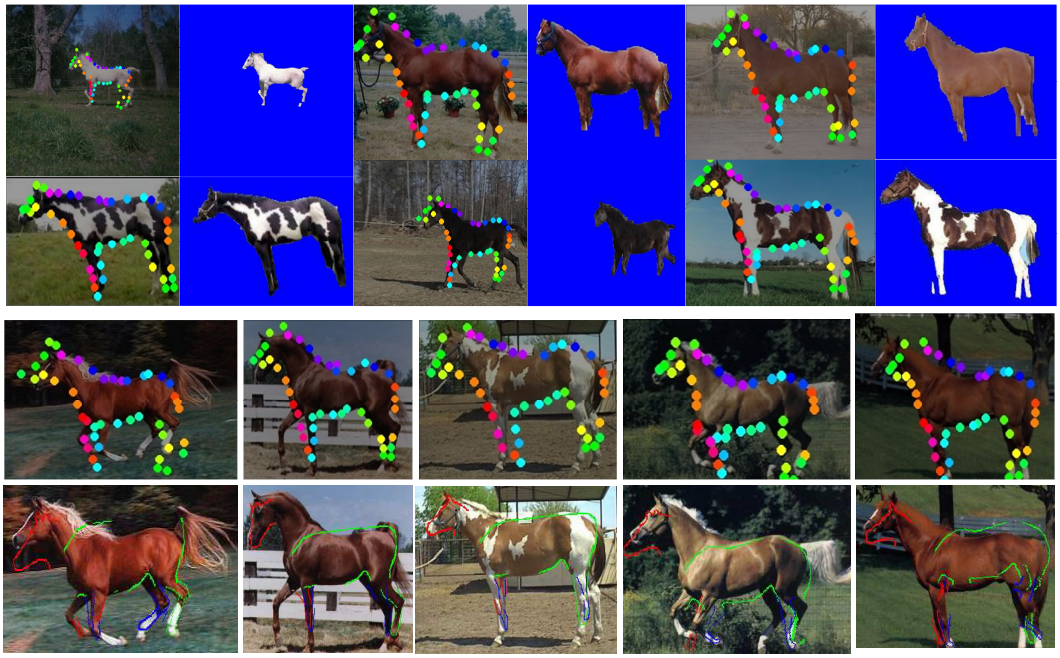


Figure 5.4: In the top two rows, Columns 1 ,3 and 5 show parse results of our method followed by segmentation results. The parse result is illustrated by the colored dots which correspond to the leaf nodes of the object. The correspondences are consistent for different images. Rows 3 and 4 show the parse results of our method and OBJ CUT (cropped from [1]) on 5 images used in [1] respectively. Note our method obtains more complete boundary.

skeleton of horse is nearly complete while the sketches obtained by the bottom-up processing miss several parts (head, leg and back). The responses of subparts of the hierarchy is depicted in the right panel of figure 5.5. One can see that the numbers of proposals decrease from low level to high level.

Complexity Analysis. Our experiments also show the relative effectiveness of our two principles as we form compositions at different levels of the hierarchy, see table (5.3). The reduction in compositions by *clustering* is fairly large (a reduction factor of 10) at level 1 but becomes negligible at higher levels as the compositions get more distinct. *Suspicious coincidences* causes major reductions in compositions with reduction factors ranging from 60 to 2000. This is because the vast number of compositions occur infrequently. *Competitive exclusion* has little effect at level 1 (a reduction factor of 5) but increases rapidly at higher levels to a reduction factor of 15 at level 4. This is because larger compositions are more likely to overlap and compete for the same niches in the images. Competitive exclusion principle is the main factor that causes the bottom-up process to stop at a specific level.

Analyze the Hierarchical Dictionary. It seems plausible that our algorithm will learn generic features (e.g., oriented straight lines, single curves, double edges, Y-junctions, T-junctions, etc.) at low-level of the hierarchy and more specific object features at the higher-levels. This is confirmed by observing the vocabularies that are extracted at different levels, see the left panel of figure (5.6). Recall that all these features are encoded by the same hierarchical composition principle. In the right panel of figure (5.6), we also plot the distribution of the concepts at the different levels of the dictionary. The peaks of levels 4, 3, 2 and 1 appear from right to left. Note that the concepts at level 1 have only one instance per image.

More objects. To demonstrate the generality of our approach, we apply it to learn models for a range of other objects collected from Caltech 101 [17], MIT LabelMe [80]

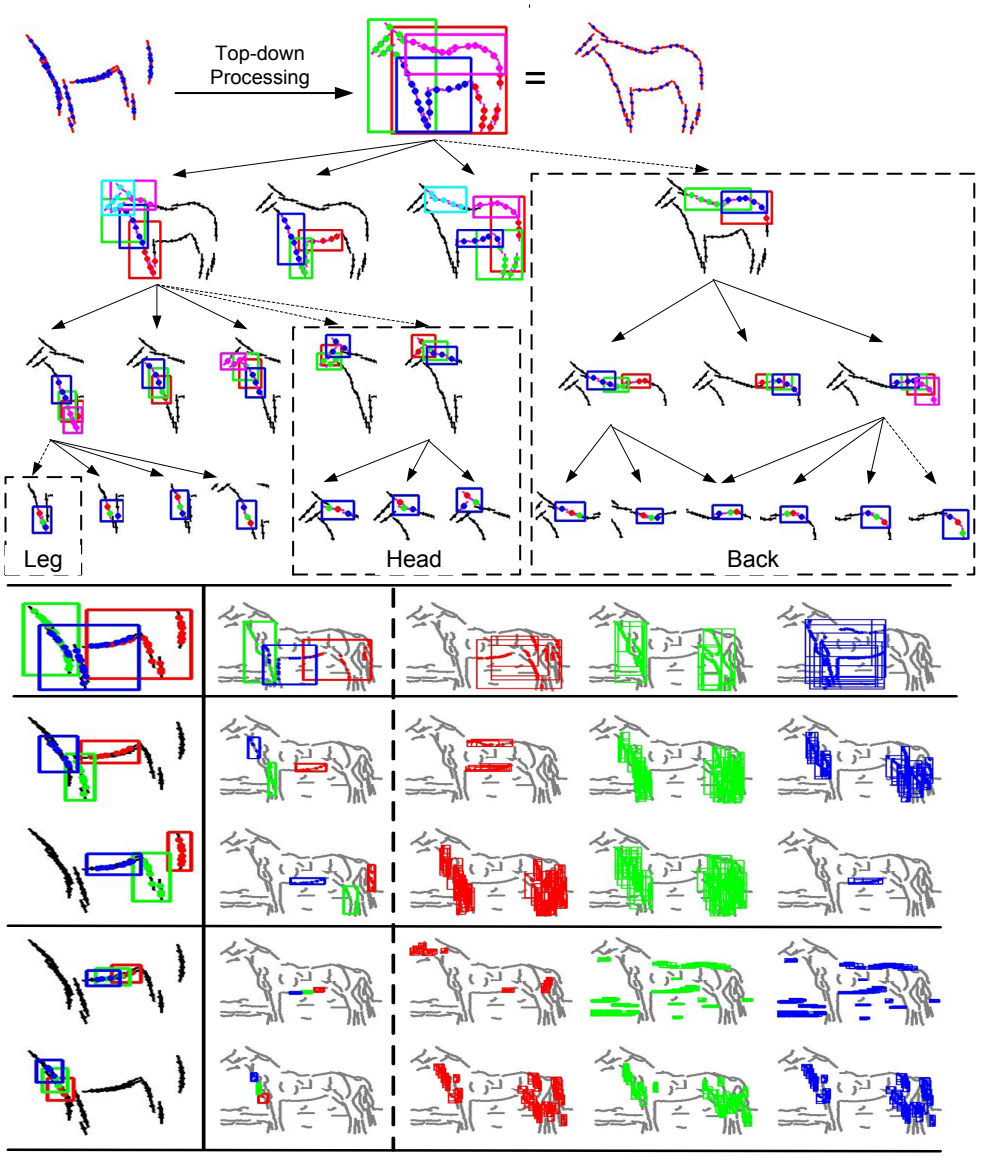


Figure 5.5: Top: The hierarchy shows the learnt hierarchical model. The colored rectangles highlight the identities of the structures. All parts are obtained at the bottom-up stage except that the rectangles in the dotted boxes show the parts of the model that were learnt in the top-down stage. Bottom: The rows illustrate structures at levels 4,3,3,2,2 (i.e. top row is level 4, next row is level 3,...). The first column gives the structure (with three children colour coded). The second column shows the structure detected on a specific image. The third, fourth, and fifth columns show the proposals for the sub-structure – colour coded.

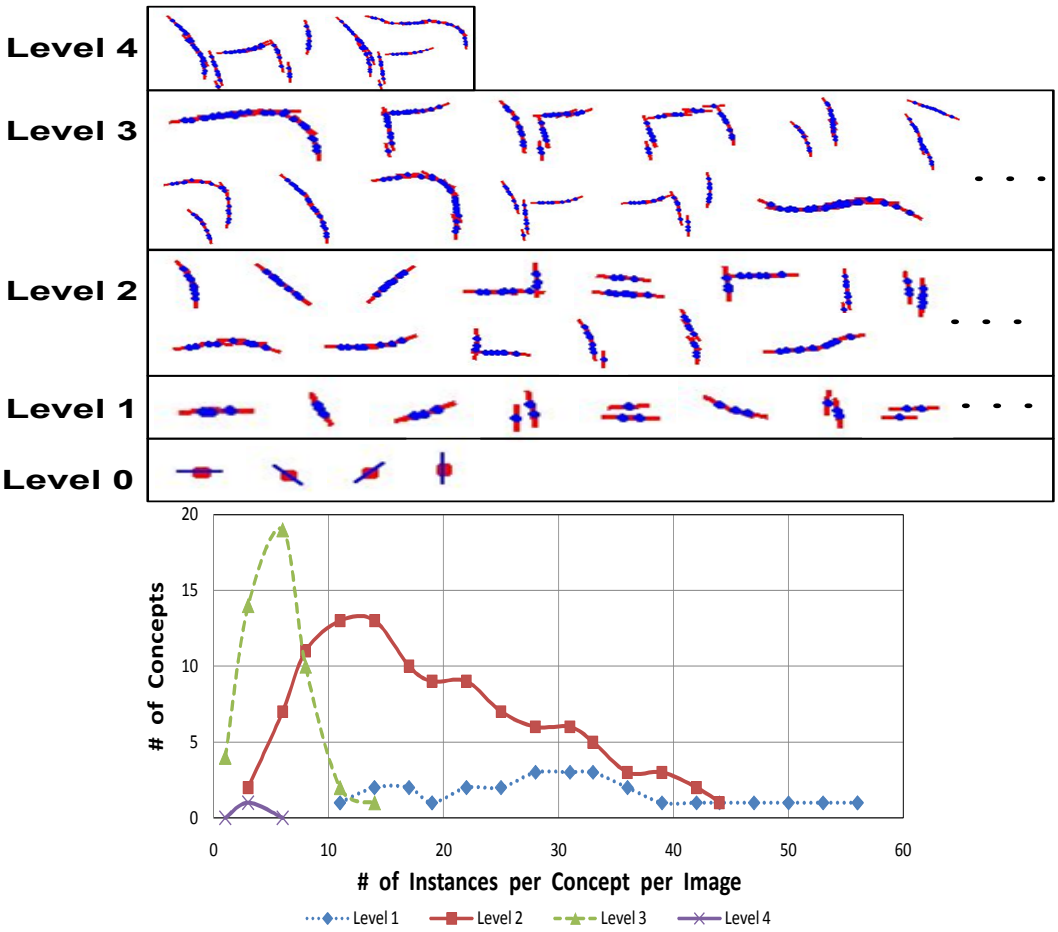


Figure 5.6: Top: This figure shows elements of the vocabulary for horses at different levels (mean values only). Observe how the vocabulary contains "generic" shapes at low levels, but finds horse specific parts at the higher levels. Bottom: This figure shows the histogram of concepts at the different levels of the dictionary. A point in a curve quantifies the number of concepts which have a certain number of instances.

L	Composition	Clusters	Suspic. Coincid.	Compet. Exclus.	Time
0				4	1s
1	167431	14684	262	48	117s
2	2034851	741662	995	116	254s
3	2135467	1012777	305	53	99s
4	236955	72620	30	2	9s

Table 5.3: Columns from 2 to 5 show the numbers of proposals after composition (step 1), clusters after clustering (step 2), after pruning (step 4) and after competitive exclusion (step 5) respectively. The next column shows the time (seconds) taken for each level. Level 0 shows the results of the grouping of the edge points(360 degrees are divided into 4 angle ranges). all numbers are calculated over 12 real images in the training dataset.

and internet, see figure (5.7). These images cover different types of shapes (man-made structure and animal), cluttered background (monitor, face and deer) and rotation (violin). These models were learnt using a small amount of training data (12 images per object). The dictionaries learnt from different objects have similar elementary structures at low levels. They can be applied to parse images using the inference algorithm. This experiment shows the versatility of our approach while modeling different types of shapes of objects.

5.6 Conclusion

We described a new method for unsupervised structure learning of a recursively compositional model for deformable objects. The structure learning was performed by a bottom-up and top-down process. We tested our approach by using it to learn hierar-

Figure 5.7: This figure illustrates the generality of our approach. We show some typical training images which contain cluttered background, different shapes and deformations. The red-sketch images show the learnt models.

chical models for horses and other objects (e.g. watches, purses, faces, grand pianos, violins, revolvers, butterflies). We evaluated the resulting models by comparing their performance to alternative methods.

CHAPTER 6

AND/OR Graph Learning

In this chapter we formulate a novel AND/OR graph representation for parsing articulated objects into parts and recovering their poses. The AND/OR graph extends the recursive deformable template model to to handle an enormous variety of articulated poses with a compact graphical model. We also extend inference algorithm, compositional inference, that uses a bottom-up compositional process for proposing configurations for the object. The strategy of surround suppression is applied to ensure that the inference time is polynomial in the size of input data. We present a novel structure-learning method, Max Margin AND/OR Graph (MM-AOG), to learn the parameters of the AND/OR graph model discriminatively. Max-margin learning is a generalization of the training algorithm for support vector machines (SVMs). The parameters are optimized globally, i.e. the weights of the appearance model for individual nodes and the relative importance of spatial relationships between nodes are learnt simultaneously. The kernel trick can be used to handle high dimensional features and to enable complex similarity measures to discriminate between object configurations. We applied our approach – the AND/OR graph representation, compositional inference and max-margin learning – to the tasks of detecting, segmenting and parsing horses and human body. We demonstrate that the inference algorithm is fast and analyze its computational complexity empirically. To evaluate max margin learning, we perform comparison experiments on the horse and human baseball datasets, showing significant improvements over state of the art methods on benchmarked datasets.

6.1 Introduction

In this chapter, we address the problem of detecting, segmenting and parsing articulated deformable objects, such as horses and human body, in cluttered backgrounds. Parsing articulated object like human body (i.e. pose estimation of body parts) in static image has recently received a lot of attention. Such problems arise in many applications including human action analysis, human body tracking, and video analysis. But the major difficulties of parsing the human body, which arise from the *large appearance variations* (e.g. different clothes) and *enormous number of poses*, have not been fully solved. There are three aspects to addressing these problems. Firstly, what representation is capable of modeling the large variation of both shape and appearance? Secondly, how can we learn a probabilistic model defined on this representation? Thirdly, if we have a probabilistic model, how can we perform inference efficiently (i.e. rapidly search over all the possible configurations of the object in order to estimate poses for novel images). These three aspects are clearly related to each other. Intuitively, the greater the representational power, the bigger the computational complexity of learning and inference. Most works in the literature, e.g. [2, 20, 11], focus on only one or two aspects, and not on all of them (see section (6.2.2) for a review of the literature). In particular, the representations used have been comparatively simple. Moreover, those attempts which do use complex representations tend to specify their parameters by hand and do not learn them from training data. The recursive deformable template model described in chapters 4 and 5 is also limited to handle a large number of articulated poses. In this chapter, we adapt the AND/OR graphs [26, 49] representation for articulated objects. AND/OR graphs offer a far richer more deeply structured representation for objects and scenes but are only useful provided efficient inference and learning algorithms can be found.

In this chapter, we *represent* the different poses of the human body and horse by the

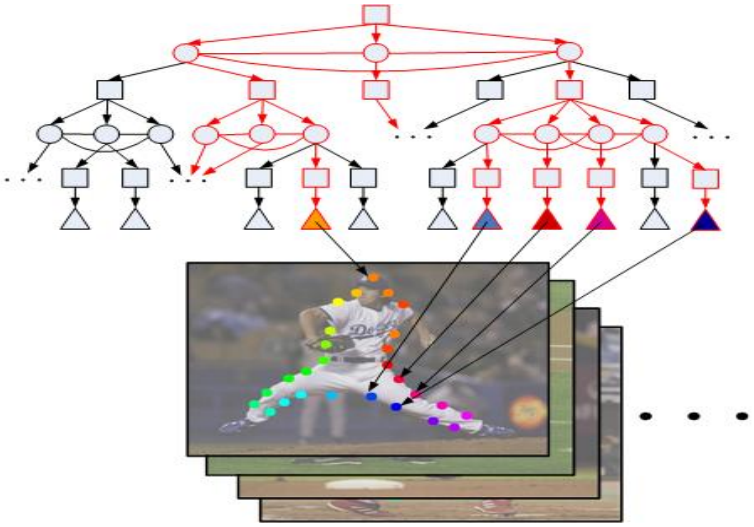


Figure 6.1: The AND/OR representation allows us to model enormous poses of the object. A parse tree which is an instantiation of the AND/OR graph represents a specific pose of the human body. The nodes and edges in red indicate one parse tree. In this chapter, there are 98 poses which can be modeled by the parse trees of the whole AND/OR graph.

form of AND/OR graphs proposed for modeling deformable articulated objects [11], see figure (6.1). The design of this graph uses the principle of *recursive composition*, so that lower level nodes in the graph only pass on summary statistics (as an abstraction) to the higher level nodes. More precisely, the nodes of the AND/OR graph specify the position, orientation and scale of sub-configurations of the object (together with an index variable which specifies which sub-configurations of the object are present). The probability distribution defined on this representation is built using local potentials and hence obeys the Markov condition. It is designed to be invariant to the position, pose, and size of the object. The advantages of this AND/OR graph (see figure (6.1)) is that it can represent an enormous number of different poses (98 for human body and 40 for horse in this chapter) in a compact form (i.e. only a small number of nodes are required), enforce (probabilistic) spatial relations on the configuration, and use many

image features as input (to address the large appearance variations).

We next describe an algorithm for performing inference over this representation. This is a challenging task since the space of possible configurations is enormous. But the use of the summarization principle in our design of the AND/OR graph enables us to use (pruned) dynamic programming for inference. This is a modification of the inference algorithm described in chapter 4.

Finally, we *learn* the model parameters, which specify the geometry and the appearance, by a novel extension of the max-margin algorithm for structure learning [82, 83, 84]. This learning is global in the sense that we learn all the parameters simultaneously (by an algorithm that is guaranteed to find the global minimum) rather than learning local subsets of the parameters independently. Max-margin learning has been shown to be more effective than standard maximum likelihood estimation when the overall goal is classification (e.g. into different poses). It also has some technical advantages such as: (i) avoiding the computation of the partition function of the distribution, and (ii) the use of the kernel trick to extend the class of features.

In summary, our method makes contributions to both machine learning and computer vision. The contribution to machine learning is to extend max-margin learning to AND/OR graphs (max-margin has previously been applied to simpler models, see section (6.2)). The contribution to computer vision is the combination of the AND/OR *representation*, the max-margin *learning*, and compositional *inference* [11] to model articulated object (horse and human body) parsing. Moreover, our results, see section (6.6), show that our approach significantly outperforms the state of the art on benchmarked datasets.

6.2 Background

6.2.1 Object Representation

Detection, segmentation and parsing are all challenging problems. Most computer vision systems only address one of these tasks. There has been influential work on detection [52] and on the related problem of registration [53],[54]. Work on segmentation includes [1], [58], [74], [68], [4], [67], and [16]. Much of this work is formulated, or can be reformulated, in terms of probabilistic inference. But the representations are fixed graph structures defined at a single scale. This restricted choice of representation enables the use of standard inference algorithms (e.g. the hungarian algorithm, belief propagation) but it puts limitations on the types of tasks that can be addressed (e.g. it makes parsing impossible), the number of different object configurations that can be addressed, and on the overall performance of the systems.

In the broader context of machine learning, there has been a growing use of probabilistic models defined over variable graph structures. Important examples include stochastic grammars which are particularly effective for natural language processing [29]. In particular, vision researchers have advocated the use of probability models defined over AND/OR graphs [26],[49] where the OR nodes enable the graph to have multiple topological structures. Similar AND/OR graphs have been used in other machine learning problems [25].

But the representational power of AND/OR graphs comes at the price of increased computational demands for performing inference (and learning). For one dimensional problems, such as natural language processing, this can be handled by dynamic programming. But computation becomes considerably harder for vision problems and it is not clear how to efficiently search over the large number of configurations of an AND/OR graph. The inference problem simplifies significantly if the OR nodes are

restricted to lie at certain levels of the graph (e.g. [37], [8]), but these simplifications are not suited to the problem we are addressing.

6.2.2 Human Body Parsing

There has been considerable recent interest in human body parsing. Sigal and Black [85] address the occlusion problem by enhancing appearance models. Triggs and his colleagues [86] learn more complex models for individual parts by SVM and combine them by an extra classifier. Mori [20] use super-pixels to reduce the search space and thus speed up the inference. Ren et al. [87] present a framework to integrate multiple pairwise constraints between parts, but their models of body parts are independently trained. Ramanan [88] proposes a tree structured CRF to learn a model for parsing human body. Lee and Cohen [89] and Zhang et al. [90] use MCMC for inference. In summary, these methods involve representations of limited complexity (i.e. with less varieties of pose than AND/OR graphs). If learning is involved, it is local but not global (i.e. the parameters are not learnt simulateously) [87, 85, 86, 20]. Moreover, the performance evaluation is performed by the bullseye criterion: outputting a list of poses and taking credit if the groundtruth result is in this list [20, 90, 2].

The most related work is by Srinivasan and Shi [2] who introduced a grammar for dealing with the large number of different poses. Their model was manually defined, but they also introduced some learning in a more recent paper [91]. Their results are the state of the art, so we make comparisons to them in section (6.6).

By contrast, our model uses the AND/OR graph in the form of Chen et al. [11] which combines a grammatical component (for generating multiple poses) with a markov random field (MRF) component which represents spatial relationships between components of the model (see [49, 26] for different types of AND/OR graph models). We perform global learning of the model parameters (both geometric and

appearance) by max-margin learning. Finally, our inference algorithm outputs a only a single pose estimate which, as we show in section (6.6), is better than any of the results in the list output by Srinivasan and Shi [2] (and their output list is better than that provided by other algorithms [20]).

6.2.3 Max Margin Structure Learning

The first example of max-margin structure learning was proposed by Altun et al. [82] to learn Hidden Markov Models (HMMs) discriminatively. This extended the max margin criterion, used in binary classification [92] and multi-class classification [93], to learning structures where the output can be a sequence of binary vectors (hence an extension of multi-class classification to cases where the number of classes is 2^n, where n is the length of the sequence). We note that there have been highly successful examples in computer vision of max-margin applied to binary classification, see SVM-based face detection [94].

Taskar et al. [83] generalized max margin structure learning to general markov random fields (MRF's), referred to a max margin markov network (M^3). Taskar et al. [84] also extended this approach to probabilistic context-free grammar (PCFG) for language parsing. But max-margin learning has not, until now, been extended to learning AND/OR graph models which can be thought of as combinations of PCFG's with MRF's.

This literature on max-margin structure learning shows that it is highly competitive with conventional maximum likelihood learning methods as used, for example, to learn conditional random fields (CRF's) [23]. In particular, max-margin structure learning avoids the need to estimate the partition function of the probability distribution (which is major technical difficulty of maximum likelihood estimation). Max-margin structure learning essentially learns the parameters of the model so that the groundtruth

states are those with least energy (or highest probability) and states which are close to groundtruth also have low energy (or high probability). See section (6.5) for details.

6.3 The AND/OR Graph Representation

6.3.1 The Topological Structure of the AND/OR Graph

The structure of an AND/OR graph is represented by a graph $G = (V, E)$ where V and E denote the set of vertices and edges respectively. The vertex set V contains three types of nodes,"OR","AND" and "LEAF" nodes which are depicted in figure (6.1) by circles, rectangles and triangles respectively. See two examples in figures (6.2) and (6.3). These nodes have attributes including position, scale, and orientation. The edge set E contains vertical edges defining the topological structure and horizontal edges defining spatial constraints on the node attributes. For each node $\nu \in V$, the set of its child nodes is defined by T_ν. Hence $\{T_\nu\}$ denotes all possible vertical edges of the AND/OR graph (the presence of OR nodes means that not all child nodes will appear in a parse, see next subsection). The horizontal edges are defined on triplets (μ, ρ, τ) of the children of AND nodes. The structure of the AND/OR graph is represented by the set of nodes and the edge set $\{(\nu, T_\nu, (\mu, \rho, \tau))\}$.

The directed (vertical) edges connect nodes at successive levels of the tree. They connect: (a) the AND nodes to the OR nodes, (b) the OR nodes to the AND nodes, and (c) the AND nodes to the LEAF nodes. The LEAF nodes correspond directly to points in the image. Connection types (a) and (c) have fixed parent-child relationships, but type (b) has switchable parent-child relationship (i.e. the parent is connected to only one of its children, and this connection can switch). The horizontal edges only appear relating the children of the AND nodes. They correspond to Markov Random Fields (MRF's) and define spatial constraints on the node attributes (implemented by

potentials). These constraints are defined to be invariant to translation, rotation, and scaling of the attributes of the children.

The AND/OR graph we use in this chapter to represent the poses of human body and horse is shown in figures (6.2) and (6.3). In figure (6.2), the top node shows all the 98 possible configurations (i.e. parse trees of the human body). These configurations are obtained by AND-ing sub-configurations such as the torso, the left leg, and the right leg of the body (see circular nodes in the second row). Each of these sub-configurations has different *aspects* as illustrated by the AND nodes (rectangles in the third row). These sub-configurations, in turn, are composed by AND-ing more elementary configurations (see fourth row) which can have different aspects (see fifth row). The overall structure of this representation was hand-specified by the authors. Future work will attempt to learn it from examples.

6.3.2 The Representational Power of the AND/OR Graph Representation

The representational power of AND/OR graph is given by the number of topological configurations of the graph which we call parse trees and which correspond to different poses. Each parse tree corresponds to a specification of which AND nodes are selected by the OR nodes (i.e. each OR node is required to select a unique child). Hence the number of different parse trees is bounded above by W^{C^h}, where C is the maximum number of children of AND nodes (in this chapter we restrict $C \leq 4$), W denotes the maximum number of possible children of OR nodes, and h is the number of levels containing OR nodes with more than one child node. The total number of parameters associated with the potential functions, which are defined on the edges of an AND/OR graph, is bounded above by MW^C where M is the number of AND nodes connecting to OR nodes. Hence the AND/OR graph can represent an exponentially large number of articulated poses, corresponding to different topologies, but with a compact form of

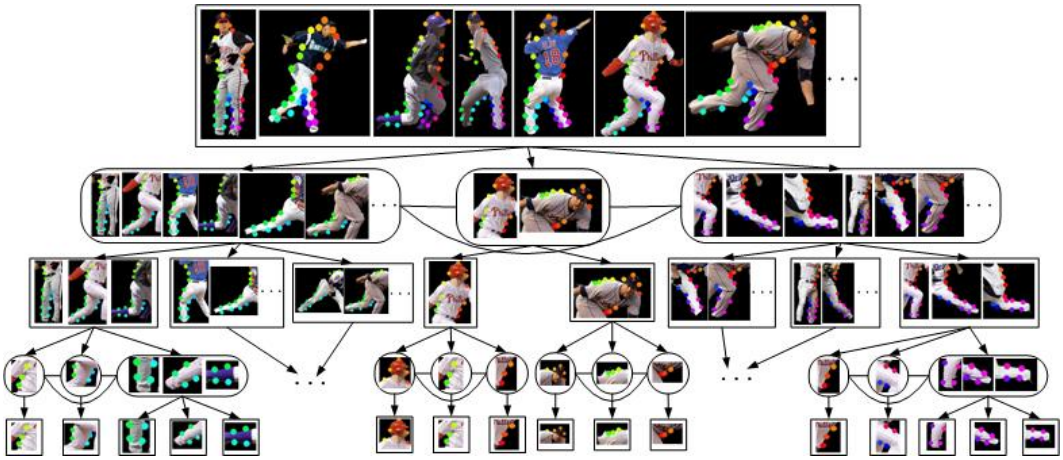


Figure 6.2: The AND/OR graph is an efficient way to represent different appearances of an object. The graph was hand specified.. The bottom level of the graph indicates points along the boundary of human body. The higher levels indicate combinations of elementary configurations. The graph that we used contains eight levels (three lower levels are not depicted here due to lack of space). Color points distinguish different body parts. The arms are not modeled in this chapter (or in related work in the literature).

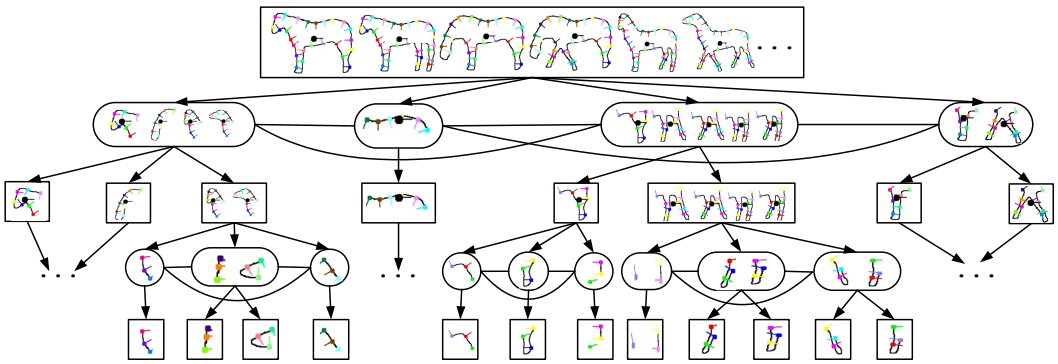


Figure 6.3: The AND/OR graph for horses. There are 40 poses allowed in this chapter. The first (typical) pose in the top node will be used for single hierarchy by fixing the child of all OR nodes.

polynomial size. This property of the AND/OR graph representation is very desirable for learning because it requires few training images to achieve good generalization. In the experiments reported in this chapter we have $M = 35, C = 4, W = 3, h = 4$. There are 98 poses modeled by the AND/OR graph.

6.3.3 The State Variables

A configuration (parse tree) of the AND/OR graph is an assignment of state variables $y = \{z_\nu, t_\nu\}$ with the state variable $z_\nu = (z^x_\nu, z^y_\nu, z^\theta_\nu, z^s_\nu)$ to each node ν, where (z^x, z^y), z^θ and z^s denote image position, orientation, and scale respectively. The $t = \{t_\nu\}$ variable defines the specific topology of the parse tree, where t_ν denotes the children of node ν. More precisely, t_ν defines the vertical parent-child relations by indexing the children of node ν. t_ν is fixed and $t_\nu = T_\nu$ if ν is an AND node (because the node is always connected to all its children – recall that T_ν is the set of child nodes of ν), but t_ν is a variable for an OR node ν (to enable sub-configurations to switch their appearances and shapes), see figure (6.2). We use the notation Λ_ν to denote the state y_ν at node ν, together with the states of all the descendent nodes of ν (i.e. the children of ν, their children, and so on). The input to the graph is the image $x = \{x_\nu\}$ defined on the image lattice (at the lowest level of the graph).

We define $V^{LEAF}(t)$, $V^{AND}(t)$,$V^{OR}(t)$ to be the set of LEAF, AND, and OR nodes which are active for a specific choice of the topology t of a parse tree. These sets can be computed recursively from the root node, see figure (6.2). The AND nodes in the second row (i.e. the second highest level of the graph) are always activated and so are the OR nodes in the third row. The AND nodes activated in the fourth row, and their OR node children in the fifth row, are determined by the t variables assigned to their parent OR nodes. This process repeats till we reach the lowest level of the graph.

A novel feature of this AND/OR representation is that the node variables are the

same at all levels of the hierarchy. We call this the *summarization principle* which make use of the compositionality. It means that the state of an AND node will be a simple deterministic function of the state variables of the children (see section (6.3.4)). This differs from other AND/OR graphs [26],[49] where the node variables at different levels of the graph are typically at different levels of abstraction. The use of the summarization principle helps us to define a successful inference algorithm.

6.3.4 The Potential Functions for the AND/OR Graph

The conditional distribution on the states and the data is given by:

$$P(y|x;\mathtt{w}) = \frac{1}{Z(x;\mathtt{w})} \exp\langle \mathtt{w}, \Psi(x,y)\rangle \,. \tag{6.1}$$

where x is the input image, y is the parse tree, and $Z(x,\mathtt{w})$ is the partition function. $P(y|x;\mathtt{w})$ is a (conditional) exponential model which is defined by an inner product $\langle \mathtt{w}, \Psi(x,y)\rangle$ between features $\Psi(x,y)$ and model parameters $\mathtt{w}$ (to be learnt). The features $\Psi(x,y)$ are of three types: (i) appearance features $\Psi^D(x,y)$, (ii) horizontal spatial relationship features $\Psi^H(y)$, and (iii) vertical relationship features $\Psi^V(y)$. Note that only the appearance features depend on the data x (the other features are like prior distributions). The inner product $\langle \mathtt{w}, \Psi(x,y)\rangle$ can be decomposed into three energy terms:

$$\langle \mathtt{w}, \Psi(x,y)\rangle = -E^D(x,y) - E^H(y) - E^V(y) \tag{6.2}$$

The data term $E^D(x,y)$ is given by:

$$E^D(x,y) = \sum_{\nu \in V^{LEAF}(t)} \mathtt{w}^D_\nu \Psi^D_\nu(x,y) \tag{6.3}$$

where the appearance features $\Psi^D_\nu(x,y), \forall \nu \in V^{LEAF}(t)$ are data dependent and model the appearance of the object. They relate the appearance of the active leaf

nodes to properties of the local image. More formally, y in $\Psi^D_\nu(x,y)$ refers to $z_\nu = (z^x, z^y, z^s, z^\theta)$ for the active nodes $\nu \in V^{LEAF}$. $\Psi^D_\nu(x, z_\nu)$ represent the local image features including the grey intensity, the gradient, canny edge map, the responses of Gabor filters at different scales and orientations, and related features. We use a total of 101 features of this type (i.e. the vector $\Psi^D(x,y)$ has 101 dimensions). But not all these features will be used (the max-margin learning will typically set some of the parameters $\mathtt{w}^D$ to be zero).

The next two terms in r.h.s of equation (6.2) make use of the hierarchical structure. The horizontal component $E^H(y)$ of the hierarchical shape prior is used to impose the horizontal connections at a range of scales and defined by

$$E^H(y) = \sum_{\nu \in V^{AND}(t)} \sum_{(\mu,\rho,\tau) \in t_\nu} \mathtt{w}^H_{\nu,\mu,\rho,\tau} \Psi^H_{\nu,\mu,\rho,\tau}(y) \tag{6.4}$$

where the horizontal spatial relationship features $\Psi^H(y)$ specify the horizontal relationships (which correspond to geometric constraints at a range of scales). They are defined by $\Psi^H_{\nu,\mu,\rho,\tau}(y) = g(z_\mu, z_\rho, z_\tau), \forall \nu \in V^{AND}(t)$ where $g(.,.,.)$ is a logarithm of Gaussian distribution defined on the *invariant shape vector* $l(z_\mu, z_\rho, z_\tau)$ (see section 3.3.2) constructed from triple child nodes (z_μ, z_ρ, z_τ) of node ν. This shape vector models the shape deformation and depends only on variables of the triple, such as the internal angles, which are invariant to the translation, rotation, and scaling of the triple. This type of feature is defined over all triples formed by the child nodes of each parent. The parameters of the Gaussians are estimated from the labeled training data (this is local learning, but max-margin will learn their parameters $\mathtt{w}^H$ globally).

The vertical component $E^V(y)$ is used to hold the structure together by relating the state of the parent nodes to the state of its children. $E^V(y)$ is divided into three vertical energy terms denoted by $E^{V,A}(y)$, $E^{V,B}(z)$ and $E^{V,C}(z)$ which correspond to type(A),

type(B) and type(C) vertical connections respectively. Hence we have

$$E^V(y) = E^{V,A}(y) + E^{V,B}(y) + E^{V,C}(y) \tag{6.5}$$

$E^{V,A}(y)$ specifies the coupling from the AND node to the OR node. This coupling is deterministic – the state of the parent node is determined precisely by the states of the child nodes. This is defined by:

$$E^{V,A}(y) = \sum_{\nu \in V^{AND}(t)} \mathtt{w}_\nu^{V,A} \Psi_\nu^{V,A}(y) \tag{6.6}$$

where $\Psi_\nu^{V,A}(y) = h(z_\nu, \{z_\mu \; s.t. \mu \in t_\nu\}), \forall \nu \in V^{AND}(t)$, where $h(.,.) = 0$ if the average orientations and positions of the child nodes are equal to the orientation and position of the parent node (i.e. the vertical constraints are "hard"). If they are not consistent, then $h(.,.) = \kappa$, where κ is a large negative number.

$E^{V,B}(y)$ accounts for the probability of the assignments of the connections from OR nodes to AND nodes. We define:

$$E^{V,B}(y) = \sum_{\nu \in V^{OR}(t)} \mathtt{w}_\nu^{V,B} \Psi_\nu^{V,B}(y) \tag{6.7}$$

where $\Psi_\nu^{V,B}(y)$ is an indicator function which equals one while the node ν is active and zero otherwise. $\mathtt{w}_\nu^{V,B}$ encodes the weights of the assignments determined by t_ν.

The energy term $E^{V,C}(y)$ defines the connection from the lowest AND nodes to the LEAF nodes. This is similar to the definition of $E^{V,A}(y)$, and $E^{V,C}(y)$ is given by

$$E^{V,C}(y) = \sum_{t_\nu \in V^{LEAF}(t)} \mathtt{w}_\nu^{V,C} \Psi_\nu^{V,C}(y) \tag{6.8}$$

where $\Psi_\nu^{V,C}(y) = h(z_\nu; z_{t_\nu})$ where $h(.,.) = 0$ if the orientation and position of the child (LEAF) node is equal to the orientation and position of the parent (AND) node. If they are not consistent, then $h(.,.) = \kappa$.

Finally, we can compute the energy of the sub-tree for a particular node ν as root node. The sub-tree energy is useful when performing inference, see section (6.4). This is computed by summing over all the potential functions associating to the node ν and its descendants. This energy is defined by:

$$E_\nu(\Lambda_\nu) = E^D(x, y) + E^H(y) + E^V(y). \tag{6.9}$$

where $y \in \Lambda_\nu$ and $V^{LEAF}(t), V^{AND}(t), V^{OR}(t)$ in the summation of each term are defined in the set of the node ν and its descendants.

6.4 The Inference/Parsing Algorithm

We use the inference algorithm first reported in [11] (an modification of the algorithm described in section 4.4) to obtain the best parse tree y^* of an image x by computing $y^* = \arg\max_y \langle \mathtt{w}, \Psi(x, y) \rangle$ where the inner product is defined in equation (6.2). This algorithm runs in polynomial time in terms of the size of input image and the number of levels of the AND/OR graph (no other algorithm has this level of inference performance on AND/OR graphs). This rapid inference is necessary to make max margin learning practical.

The algorithm has a bottom-up stage which makes proposals for the configuration of the AND/OR graph. This proceeds by combining proposals for sub-configurations to build proposals for larger configuration. For AND nodes, we combine proposals for the child nodes to form a proposal for the parent node. For OR nodes, we enumerate all proposals from all branches without composition. To prevent a combinatorial explosion we prune out weak proposals which have low fitness score ($\langle \mathtt{w}, \Psi(x, y) \rangle$ evaluated for the configuration) and use clustering which selects a small set of *max-proposals* (each representing a cluster).

The pseudo-code for the algorithm is shown in figure 6.4. We use the follow-

ing notation. Each node ν^l at level l has a set of *proposals* $\{P^l_{\nu,a}\}$ where a indexes the proposals (see table (6.2) for the typical number of proposals). There are also max-proposals $\{MP^l_{\nu,a}\}$, indexed by a, each associated with a local cluster $\{CL^l_{\nu,a}\}$ of proposals (see table (6.2) for the typical number of max-proposals). Let S denote the number of clusters. Each proposal, or max-proposal, is described by a state vector $\{y^l_{\nu,a} : a = 1, ..., M^l_\nu\}$, the state vectors for it and its descendants $\{\Lambda^l_{\nu,a} : a = 1, ..., M^l_\nu\}$, and an energy function score $\{E^l_\nu(\Lambda^l_{\nu,a}) = \langle \mathtt{w}, \Psi(x, y)\rangle : a = 1, ..., M^l_\nu\}$.

We obtain the proposals by a bottom-up strategy starting at level $l = 2$ (AND node) of the tree. For a node ν^2 we define windows $\{W^2_{\nu,a}\}$ in space, orientation, and scale. We exhaustively search for all configurations within this window which have a score (goodness of fit criterion) $E^2_\nu(\Lambda^2_{\nu,a}) < K_2$, where K_2 is a fixed threshold. For each window $W^2_{\nu,a}$, we select the configuration with largest score to be the proposal $MP^2_{\nu,a}$ and store the remaining proposals below threshold in the associated cluster $CL^2_{\nu,a}$. This window enforces *surround suppression* which performs clustering to keep the proposal with the maximum score in any local window. Surround suppression grantees the number of the remaining proposals at each level is proportional to the size of image (input data). Note that the potential functions associated with the nodes at level l only rely on the position, orientation and scale at level l, but not on the states of its descendants at level $l-1, l-2, .., 1$. Therefore, different detailed configurations of subparts with the same global pose will have identical energies for the higher levels. This strategy essentially is (pruned) dynamic programming and ensures that we do not obtain too many proposals in the hierarchy and avoid a combinatorial explosion of proposals. We will analyze this property empirically in section 6.6. The procedure is repeated as we go up the hierarchy. Each parent node ν^{l+1} produces proposals $\{P^{l+1}_{\nu,a}\}$, and associated clusters $\{CL^{l+1}_{\nu,a}\}$, by combining the proposals from its children. All proposals are required to have scores $E^{l+1}_\nu(\Lambda^{l+1}_\nu) < K_{l+1}$, where K_l is a threshold.

Input: $\{MP^1_{\nu^1}\}$. Output:$\{MP^L_{\nu^L}\}$. $\oplus$ denotes the operation of combining two proposals.

Loop : $l = 1$ to L, for each node ν at level l

- IF ν is an OR node
  1. Union: $\{MP^l_{\nu,b}\} = \bigcup_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
- IF ν is an AND node
  1. Composition: $\{P^l_{\nu,b}\} = \oplus_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
  2. Pruning: $\{P^l_{\nu,a}\} = \{P^l_{\nu,a} | E(\Lambda^l_{\nu,a}) > K_l\}$
  3. Local Maximum: $\{(MP^l_{\nu,a}, CL^l_{\nu,a})\} = LocalMaximum(\{P^l_{\nu,a}\}, \epsilon_W)$ where ϵ_W is the size of the window W^l_ν defined in space, orientation, and scale.

Figure 6.4: The inference algorithm. The operation LocalMaximum implements surround suppression.

Recall that the cluster is defined over image position, orientation and scale. Thus, the number S of proposals of each node at different levels is linearly proportional to the size of the image. It is straight forward to conclude that the complexity of our algorithm is bounded above by $M \times W^C \times S^C$. Recall that C is the maximum number of children of AND nodes (in this chapter we restrict $C \leq 4$), W denotes the maximum number of possible children of OR nodes and M is the number of AND nodes connecting to OR nodes. This shows that the algorithm speed is polynomial in W and S (and hence in the image size). The complexity for our experiments is reported in section (6.6).

In practice, the thresholds K_l are not explicitly defined. Instead, we keep top K proposals for each node whose energy scores are greater than those of any other proposals. K is empirically set to be 300 in our experiments. In very rare situations we may find no proposals for the state of one node of a triplet. In this case, we use the states of the other two nodes together with the horizontal potentials (geometrical relationship) to propose states for the node. A similar technique was used in [8].

6.5 Max Margin AND/OR Graph Learning

6.5.1 Primal and Dual Problems

The task of AND/OR graph learning is to estimate the parameters $\mathtt{w}$ from a set of training samples (x_1, y_1),...,$(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ drawn from some fixed, but unknown probability distribution. In this chapter, x is the image and y is the configurations of the AND/OR graph.

We formulate this learning task in terms of the max-margin criterion which is designed to learn the parameters which are best for classification (i.e. to estimate y) rather than use the standard maximum likelihood criterion (see [92] for a justification

for this strategy). But observe that the classification is over the set of values $\mathcal{Y}$, which is exponentially large, and hence differs greatly from simple binary classification. Effectively max-margin learning seeks to find values of the parameters $\mathtt{w}$ which ensure that the energies $\langle \Psi(x, y), \mathtt{w} \rangle$ are smallest for the ground-truth states y and for states close to the ground-truth. A practical advantages of max-margin learning is that it gives a computationally tractable learning algorithm (which avoids the need to compute the partition function of the distribution).

The main idea of the max margin approach is to forego the probabilistic interpretation of equation (6.1). Instead we concentrate on the discriminative function:

$$F(x, y, \mathtt{w}) = \langle \Psi(x, y), \mathtt{w} \rangle . \tag{6.10}$$

We define the *margin* γ of the parameter $\mathtt{w}$ on example i as the difference between the true parse y_i and the best parse y^*:

$$\gamma_i = F(x_i, y_i, \mathtt{w}) - \max_{y \neq y_i} F(x_i, y, \mathtt{w}) \tag{6.11}$$

$$= \langle \mathtt{w}, \Psi_{i,y_i} - \Psi_{i,y^*} \rangle \tag{6.12}$$

where $\Psi_{i,y_i} = \Psi(x_i, y_i)$ and $\Psi_{i,y} = \Psi(x_i, y)$.

Intuitively, the size of margin quantifies the confidence in rejecting the incorrect parse y using the function $F(x, y, \mathtt{w})$. Larger margins [92] leads to better generalization and prevents over-fitting.

The *goal* of max margin AND/OR graph learning is to maximize the margin:

$$\max_{\gamma} \gamma \tag{6.13}$$

$$s.t. \quad \langle \mathtt{w}, \Psi_{i,y_i} - \Psi_{i,y} \rangle \geq \gamma L_{i,y}, \forall y; \quad \|\mathtt{w}\|^2 \leq 1; \tag{6.14}$$

where $L_{i,y} = L(y_i, y)$ is a loss function (note there are an exponential number $|\mathcal{Y}|$ of constraints in equation (6.14). The purpose of the loss function is to give partial

credit to states which differ from the groundtruth by only small amounts (i.e. it will encourage the energy to be small for states near the groundtruth).

The loss function is defined as follows:

$$L(y_i, y) = \sum_{\nu \in V^{AND}} \triangle(z^i_\nu, z_\nu) + \sum_{\nu \in V^{LEAF}} \triangle(z^i_\nu, z_\nu) \tag{6.15}$$

where $\triangle(z^i_\nu, z_\nu) = 1$ if $dist(z^i_\nu, z_\nu) \geq \sigma$. Otherwise, we have $\triangle(z^i_\nu, z_\nu) = 0$. $dist(.,.)$ is a measure of the spacial distance between two image points and σ is a threshold. Note that the summations are defined over the active nodes. This loss function, which measures the distance/cost between two parse trees, is calculated by summing over individual parts. This ensures that the computational complexity of the loss function is linear in the size of the LEAF and AND nodes of the hierarchy.

By standard manipulation, the optimization can be reformulated as minimizing the constrained quadratic cost function of the weights:

$$\min_{\mathtt{w}} \frac{1}{2}\|\mathtt{w}\|^2 + C\sum_i \xi_i \tag{6.16}$$

$$s.t. \quad \langle \mathtt{w}, \Psi_{i,y_i} - \Psi_{i,y} \rangle \geq L_{i,y} - \xi_i, \forall y; \tag{6.17}$$

where C is a fixed penalty parameter which balances the trade-off between margin size and outliers. Outliers are training samples which are only correctly classified after using a slack variable ξ_i to "move them" to the correct side of the margin. The constraints are imposed by introducing Lagrange parameters $\alpha_{i,y}$ (one α for each constraint).

The solution to this minimization can be found by differentiation and expressed in form:

$$\mathtt{w}^* = C\sum_{i,y} \alpha^*_{i,y} \left(\Psi_{i,y_i} - \Psi_{i,y}\right), \tag{6.18}$$

where the α^* are obtained by maximizing the dual function:

$$\max_{\alpha} \sum_{i,y} \alpha_{i,y} L_{i,y} -$$
$$\frac{1}{2} C \sum_{i,j} \sum_{y,z} \alpha_{i,y} \alpha_{j,z} \left\langle \Psi_{i,y_i} - \Psi_{i,y}, \Psi_{j,y_j} - \Psi_{j,z} \right\rangle \tag{6.19}$$
$$s.t. \quad \sum_{y} \alpha_{i,y} = 1, \forall i; \quad \alpha_{i,y} \geq 0, \forall i, y; \tag{6.20}$$

Observe that the solution will only depend on the training samples (x_i, y_i) for which $\alpha_{i,y} \neq 0$. These are the so-called *support vectors*. They correspond to training samples that either lie directly on the margin or are outliers (that need to use slack variables). The concept of support vectors is important for the optimization algorithm that we will use to estimate the α^* (see next subsection).

It follows from equations (6.18,6.19), that the solution only depends on the data by means of the inner product $\Psi \cdot \Psi'$ of the potentials. This enables us to use the kernel trick [95] which replaces the inner product by a kernel $K(,)$ (interpreted as using features in higher dimensional spaces). In this chapter, the kernels $K(,)$ take two forms, the linear kernel, $K(\Psi, \Psi') = \Psi \cdot \Psi'$ for image features Ψ^D and the radial basis function (RBF) kernel, $K(\Psi, \Psi') = \exp(-r\|\Psi - \Psi'\|^2)$ for shape features Ψ^H where r is a parameter of RBF.

6.5.2 Optimization of the Dual

The main diificulty with optimizing the dual, see equation (6.19), is the exponential number of constraints (and hence the exponential number of $\{\alpha_{i,y}\}$ to solve for). We risk having to enumerate all the parse trees $y \in \mathcal{Y}$ which is almost impractical for an AND/OR graph. Fortunately, in practice only a small number of support vectors will be needed (equivalently, only a small number of the $\{\alpha_{i,y}\}$ will be non-zero). This motivates the working set algorithm [82, 96] to optimize the objective function in equation

(6.19). The algorithm aims at finding a small set of *active constraints* that ensure a sufficiently accurate solution. More precisely, it sequentially creates a nested working set of successively tighter relaxations using a cutting plane method. It is shown [82, 96] that the remaining (exponentially many) constraints are guaranteed to be violated by no more than ϵ, without needing to explicitly add them to the optimization problem. The pseudocode of the algorithm is given in figure (6.5). Note that the inference algorithm is performed at the first step of each loop. Therefore, the efficiency of the training algorithm highly depends on the computational complexity of the inference algorithm (recall that we show in section (6.4) that the complexity of the inference algorithm is polynomial in the size of the AND/OR graph and the size of the input image). Thus, the efficiency of inference makes the learning practical. The second step is to create the working set sequentially and then estimate the parameter α on the working set. The optimization over the working set is performed by Sequential Minimal Optimization (SMO) [97]. This involves incrementally satisfying the Karush-Kuhn-Tucker (KKT) conditions which are used to enforce the constraints. The pseudo-code of the SMO algorithm is depicted in figure (6.6). This procedure consists of two step. The first step selects a pair of data points not satisfying the KKT conditions. The pseuod-code of pair selection is shown in figure (6.7). Two KKT conditions are defined by:

$$\alpha_{i,y} = 0 \Rightarrow H(x_i, y) \leq H(x_i, y^*) + \epsilon; \quad (KKT1)$$

$$\alpha_{i,y} > 0 \Rightarrow H(x_i, y) \geq H(x_i, y^*) - \epsilon; \quad (KKT2)$$

where $H(x_i, y) = \langle \mathtt{w}, \Psi_{i,y} \rangle + L(y_i, y)$, $y^* = \arg\max_y H(x_i, y)$ and ϵ is a tolerance parameter.

The second step is a local ascent step which attempts to update the parameters

Loop over i

1. $y^* = \arg\max_y H(x_i, y)$ where $H(x_i, y) = \langle \mathtt{w}, \Psi_{i,y} \rangle + L(y_i, y)$.
2. if $H(x_i, y^*; \alpha) - \max_{y \in S_i} H(x_i, y; \alpha) > \varepsilon$
   $S_i \leftarrow S_i \bigcup y^*$
   $\alpha_s \leftarrow$optimize dual over S, $S = S \bigcup S_i$

Figure 6.5: Working Set Optimization

given the selected pair. The updating equations are defined as:

$$\alpha_{x_i,y'}^{new} = \alpha_{x_i,y'} + \delta$$
$$\alpha_{x_i,y''}^{new} = \alpha_{x_i,y''} - \delta \tag{6.21}$$

The dual optimization problem in equation (6.19) becomes a simple problem on δ:

$$\max_{\delta} [H(x_i, y') - H(x_i, y'')]\delta - \frac{1}{2} C \|\Psi_{i,y'} - \Psi_{i,y''}\|^2 \delta^2 \tag{6.22}$$
$$s.t. \quad \alpha_{x_i,y'} + \delta \geq 0, \alpha_{x_i,y''} - \delta \geq 0 \tag{6.23}$$

Equivalently we have :

$$\max_{\delta} a\delta - \frac{b}{2}\delta^2 \tag{6.24}$$
$$s.t. \quad c \leq \delta \leq d \tag{6.25}$$

where $a = H(x_i, y') - H(x_i, y'')$, $b = C\|\Psi_{i,y'} - \Psi_{i,y''}\|^2$, $c = -\alpha_{x_i,y'}$, $d = \alpha_{x_i,y''}$.

Hence, the analytical solution for two data points can be easily obtained by

$$\delta^* = \max(c, \min(d, a/b)). \tag{6.26}$$

Up to now, we have the solutions for the updating equations in (6.21). More details can be found in [97] and [83].

Given a training set S and parameter α

Repeat

1. select a pair of data points (y', y'') not satisfying the KKT conditions.
2. solve optimization problem on (y', y'')

Until all pairs satisfy the KKT conditions.

Figure 6.6: Sequential Minimal Optimization (SMO)

1. $Violation = False$
2. For each $x^i, y', y'' \in S_i$
   (a) If $H(x^i, y') > H(x^i, y'') + \epsilon$ and $\alpha_{x_i,y'} = 0$ (KKT 1)
   $Violation = TRUE$; Goto step 3.
   (b) If $H(x^i, y') < H(x^i, y'') - \epsilon$ and $\alpha_{x_i,y'} > 0$ (KKT 2)
   $Violation = TRUE$; Goto step 3.
3. Return y', y'', v

Figure 6.7: Pair Selection in SMO

6.6 Experiments

In this section, we first study the performance of the AND/OR graph with max-margin learning on the horse dataset [18] and analyze the computational complexity of the inference. Next we apply the AND/OR graph for parsing the human body which has more poses and compare its performance with alternative methods.

6.6.1 Datasets and Implementation Details.

Datasets. We performed the experimental evaluations on two datasets, i.e. the Weizmann Horse Dataset [18] and Mori's Human Baseball dataset [20]. There are many results reported for comparisons ([16, 68, 4, 1, 67] for horse segmentation and [2, 98, 20] for human body parsing). The Weizmann horse dataset is designed to evaluate segmentation, so the groundtruth only gives the regions of the object and the background. To supplement this groundtruth, we required students to manually parse the images by locating the positions of active leaf nodes (about 24 to 36 nodes) of the AND/OR graph in the images. These parse trees are used as ground truth to evaluate the ability of the AND/OR graph to parse the horses. In the experiment of human body parsing, Srinivasan and Shi [2] only used 5 joint nodes (head-torso, torso-left thigh, torso-right thigh, left thigh-left lower leg, right thigh-right lower leg) per image. In our case, there are 27 nodes along the boundary of human body per image used to give more detailed parsing than those [2, 98, 20]. Therefore, we also asked students to label the parts of the human body as ground truth (i.e. to identify different parts of the human). There are 328 horse images in [18] of which 100 images are used for testing. The AND/OR model has 40 possible configurations to cover horse poses. For human body parsing, we used 48 human baseball images in Mori's dataset [20] as the testing set. Some examples of the dataset are shown in figures (6.8) and (6.9) (The parsing and segmen-

tations results are obtained by our method). The AND/OR graph for human body is capable of modeling 98 poses. In figure (6.9), observe that the dataset contains a large variance of poses of human body and the appearance of clothes changes a lot from image to image. We created a training dataset by collecting 156 human baseball images from the internet and got students to manually label the parse tree for each image.

Parameter Settings. The AND/OR graph learnt by max-margin was used to obtain the parse y (i.e. to locate the body parts). We used max-margin on the training dataset to learn the parameters of the max-margin model. During learning, we set $C = 0.1$ in equation (6.19), used the radial basis function kernel with $r = 0.1$, set the parameter in the loss function equation (6.15) to be $\sigma = 12$, and set $\epsilon = 0.01$ in figure (6.5). Our strategy to obtain segmentation, which is inspired by Grab-Cut [73], is to obtain the parse by the inference algorithm on the AND/OR graph and then segment object by graph-cut using the feature statistics inside the boundary as initializations (note that, unlike our approach, Grab-Cut requires initialization by a human).

The Criterion for Parsing. The *average position error* [2] is used as the measure of the quality of parsing. The position error means the distance at pixel level between the positions of groundtruth and the parsing result. The smaller the position error, the better the quality of the parsing. For horse, there are 24 to 36 leaf nodes used to cover the boundary of a horse. In the experiment of human body parsing, Srinivasan and Shi [2] only used 5 joint nodes (head-torso, torso-left thigh, torso-right thigh, left thigh-left lower leg, right thigh-right lower leg) per image. In our case, there are 27 nodes along the boundary of human body per image used to give more detailed parsing.

The Criterions for Segmentation. Two evaluation criterions are used to measure the performances of segmentation. We use *segmentation accuracy* to quantify the proportion of the correct pixel labels (object or non-object). For performance comparisons of human body parsing, we use the segmentation measure, "*overlap score*" named by

[2], to quantify the performance of segmentation of human body. The overlap score is defined by $\frac{area(P \cap G)}{area(P \cup G)}$, where P is the area which the algorithm outputs as the segmentation and G is the area of ground-truth. The bigger the overlap score, the better the segmentation.

The Criterion for Detection. We rate detection as a success if the area of intersection of the detected object region and the true object region is greater than half the area of the union of these regions.

6.6.2 Performance of the AND/OR graph on the Horse dataset

Results. In table (6.1) we compare the performances between the AND/OR graph with 40 configurations and a simple hierarchical model with a fixed configuration (i.e. we fix the states of the OR nodes). This configuration (the first one in the top node in figure (6.3)) was chosen to be the typical pose that most frequently occurred. Column 3 gives the parsing accuracy – the average position error of leaf node of the AND/OR graph is 10 pixels. Column 4 quantifies the segmentation accuracy. Column 5 quantifies the detection rate. Column 6 lists the training time. The last column shows the average time of inference taken for one image. A computer with 4 GB memory and 2.4 GHz CPU was used for training and testing. The time costs are 150 minutes for learning a hierarchy and 180 minutes for AND/OR graph. For a new image, the testing (inference) time is 20 seconds for the hierarchy model and 27 second for the AND/OR model. The AND/OR graph outperforms the simple hierarchical model in the tasks of parsing, detection and segmentation with only 30% more computational cost. In figure (6.8), we show the parse and segmentation results obtained by the single hierarchy model and the AND/OR graph model. The states of the leaf nodes of parse tree indicate the positions of the points along the boundary which are represented as colored dots. In different images, the same color corresponds to the same object parts. Both

models learnt by max-margin learning are able to deal with large shape deformation and appearance variations. See the top four examples which contains the white, black and textured body with cluttered background. The hierarchical model was only capable of reliably locating the main body but the AND/OR graph is able to reliably capture more details such as the legs and heads (despite their variability under different poses). See the last four examples in figure (6.8) where the legs and heads appear at different poses. The hierarchical model succeeds in segmentation in most cases even though its parse results are not accurate. In the last example, the incorrect parsing, where the hierarchical model locates the head at a wrong position with similar appearance, results in wrong segmentation. In this case, the AND/OR graph model performs well on both parsing and segmentation tasks.

Comparisons. In table (6.1), we compare the segmentation performance of our approach with other successful methods. Note that the object cut method [1] was reported on only 5 images. Levin and Weiss [4] make the strong assumption that the position of the object is given (other methods do not make this assumption) and not report how many images they tested on. Overall, Cour and Shi's method [68] was the best one evaluated on large dataset. But their result is obtained by manually selecting the best among top 10 results (other methods output a single result). By contrast, our approach outputs a single parse only but yields a higher pixel accuracy of 95.2%. Hence we conclude that our approach outperforms those alternatives which have been evaluated on this dataset. Note no other papers report parsing performance on this dataset since most (if not all) methods do not estimate the positions of different parts of the horse (or even represent them).

Figure 6.8: This figure is best viewed in color. Columns from (a) to (d) show the parsing and segmentation results obtained by hierarchy and AND/OR graph models respectively. The colored dots correspond to the leaf nodes of the object.

Table 6.1: Performance for parsing, segmentation and detection. The table compares the results for the hierarchial model (without OR nodes), AND/OR graph and other alternative methods.

Models	Testing Size	Parsing	Seg.	Det.	Training	Testing
Hierarchical Model	100	15.6	94.5%	99%	150 m	20s
AND/OR Graph	50	9.7	95.2%	100%	180 m	27s
Ren et. al[16]	172	–	91%	–	–	–
Borenstein [74]	328	–	93.0%	–	–	–
LOCUS [67]	200	–	93.1%	–	–	–
Cour [68]	328	–	94.2%	–	–	–
Levin [4]	N/A	–	95.0%	–	–	–
OBJ CUT [1]	5	–	96.0%	–	–	–

6.6.3 Computational Complexity Analysis

Table (6.2) shows the complexity properties of the algorithm. We described the AND levels only (the model has 8 levels). The computation for the OR-nodes is almost instantaneous (you just need to list the proposals from all its children AND nodes) so we do not include it. Column 2 gives the number of nodes at each level. Column 3 states the average number of *aspects* [1] of the AND nodes at each level. Column 4 states the average number of max-proposals for each node. Column 5 gives the average number of proposals. Column 6 gives the time. Observe that the number of proposals increases by an order of magnitude from level 6 to level 8. This is mostly due to the similar increase in the number of aspects (the more the number of aspects, the more

[1]Here is the definition of *aspects*. Let AND node ν have children OR nodes $\{\rho_i : i \in t_\nu\}$. This gives a set of grandchildren AND nodes $\bigcup_{i \in t_\nu} t_{\rho_i}$. The aspect of ν is $\prod_{i \in t_\nu} |t_{\rho_i}|$. The aspect of an AND node is an important concept because when passing proposals up to an AND node we must take into account the number of aspects of this node. We can, in theory, have proposals for all possible aspects.

Table 6.2: Complexity Analysis. This table shows the numbers of proposals and time costs at different levels.

L	Nodes	Aspects	Max-Proposals	Proposals	Time
8	1	12	11.1	2058.8	1.206s
6	8	1.5	30.6	268.9	1.338s
4	27	1	285.1	1541.5	1.631s
2	68	1	172.2	1180.7	0.351s

the number of proposals needed to cover them). But surround suppression is capable of reducing the number of proposals greatly (compare the numbers of Max-proposals and proposals in Table (6.2)).

6.6.4 Human Body Parsing

Parsing Results. We illustrate our parsing and segmentation results of human body in figure (6.9). The dotted points indicate the positions of the leaf nodes of parse tree which lie along the boundary of human body. The same parts in different images share the same color. For example, yellow and red points correspond to the left and right shoulder respectively. Light blue and dark blue points correspond to the left and right legs respectively. Observe that the variation of poses are extremely large, but our AND/OR graph is capable of covering the articulated poses of body parts and segmenting the body nicely. The time cost of training the AND/OR graph is 20 hours. The inference takes 2 minutes for image with size 640×480.

Performance Comparisons. We compare the performance obtained by our approach to those reported by Srinivasan and Shi [2], which are the best results achieved so far on this dataset (e.g. better than Mori et al. 's [20]). Firstly, we compare the

Figure 6.9: The first column shows the parse results of the human body. Color points indicate the positions of body parts. The same color points in different images correspond to the same parts. The second column show the segmentations of human body. The next four columns show extra examples.

average position errors in figure (6.10). Observe that our best parse gives performance slightly better than the best (manually selected) of the top 10 parses output by [2] and significantly better than the best (manually selected) of their top three parses. Secondly, we compare the average overlap scores in figure (6.10). The difference of performance measured by overlap score is more significant. Observe that our result is significantly better than the best (manually selected) of their top 10 parses.

Convergence Analysis. We study the convergence behavior of max-margin AND/OR graph learning in figure (6.11). The left figure shows the convergence curve in terms of objective function defined in equation (6.19). There is a big jump before iteration 200. The right figure plots the convergence curves of the average position error on the training and testing data. One can see that the trends of two curves are very similar.

6.7 Discussion

We formulated a novel AND/OR graph representation capable of describing the different configurations of deformable articulated objects. The representation makes use of the summarization principle which distinguishes it from other type of AND/OR graph [26]. We developed a novel compositional inference algorithm for proposing configurations. Surround suppression ensures that the inference time is polynomial in the size of image. We demonstrated that the algorithm was fast and effective as evaluated by performance measures on two public datasets. We learn the parameters of the AND/OR graph in a globally optimal way by extending max-margin structure learning technique developed in machine learning. Advantages of our approach include (i) the ability to model the enormous number of poses that occur for articulated objects such as humans, (ii) the discriminative power provided by max-margin learning (by contrast to MLE), and (iii) the use of the kernel trick to make use of high-dimensional features. We gave detailed experiments on the Weizmann horse and human baseball datasets,

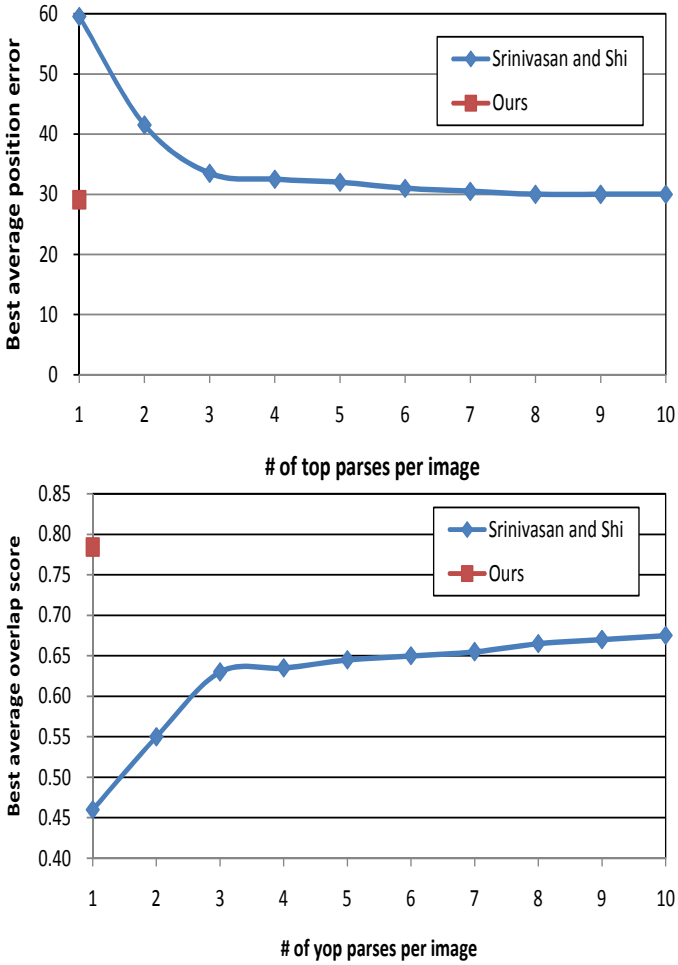


Figure 6.10: We compare our results with that of Srinivasan and Shi [2]. The performance of parsing (position error) and segmentation (overlap score) are shown in the top and bottom figures respectively. Note that [2] select the best one (manually)of the top parses.

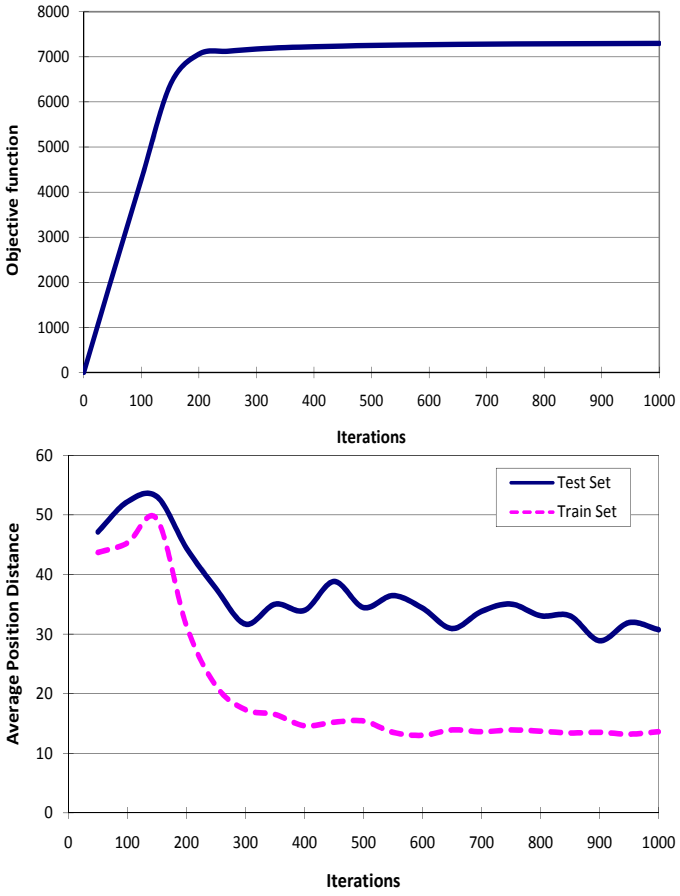


Figure 6.11: Convergence Analysis. We study the behavior of max margin training. The first panel shows the convergence curve in terms of the objective function defined in equation (6.19). The second panel shows the converge curves of the average position error evaluated on training and testing set.

showing significant improvements over the state-of-the-art methods. We are currently working on improving the inference speed of our algorithm by using a cascade strategy. We are also extending the model to represent humans in more details.

Part IV

Image Parsing by Recursive Segmentation - Recognition Template

CHAPTER 7

Image Understanding by Recursive Segmentation and Recognition Template

In this chapter, we will move from object modeling to image modeling and understanding, and apply the same recursive composition principle to the design of image parser. Particularly, we are interested in the task of image segmentation and scene labeling. Our solutions are partially influenced by natural language processing.

Language and image understanding are two major goals of artificial intelligence which can both be conceptually formulated in terms of parsing the input signal into a hierarchical representation. Natural language researchers have made great progress by exploiting the 1D structure of language to design efficient polynomial-time parsing algorithms. By contrast, the two-dimensional nature of images makes it much harder to design efficient image parsers and the form of the hierarchical representations is also unclear. Attempts to adapt representations and algorithms from natural language have only been partially successful.

In this chapter, we propose a Hierarchical Image Model (HIM) for 2D image parsing which outputs image segmentation and object recognition. This HIM has multiple layers and has advantages for representation, inference, and learning. Firstly, the HIM has a coarse-to-fine representation which is capable of capturing long-range dependency and exploiting different levels of contextual information. Secondly, the structure of the HIM allows us to design a rapid inference algorithm, based on dy-

namic programming, which enables us to parse the image rapidly in polynomial time. Thirdly, we can learn the HIM efficiently in a discriminative manner from a labeled dataset. We demonstrate that HIM outperforms other state-of-the-art methods by evaluation on the challenging public MSRC image dataset. Finally, we sketch how the HIM architecture can be extended to model more complex image phenomena.

7.1 Introduction

Language and image understanding are two major tasks in artificial intelligence. Natural language researchers have formalized this task in terms of parsing an input signal into a hierarchical representation. They have made great progress in both representation and inference (i.e. parsing). Firstly, they have developed probabilistic grammars (e.g. stochastic context free grammar (SCFG) [99] and beyond [100]) which are capable of representing complex syntactic and semantic language phenomena. For example, speech contains elementary constituents, such as nouns and verbs, that can be recursively composed into a hierarchy of (e.g. noun phrase or verb phrase) of increasing complexity. Secondly, they have exploited the one-dimensional structure of language to obtain efficient polynomial-time parsing algorithms (e.g. the inside-outside algorithm [101]).

By contrast, the nature of images makes it much harder to design efficient image parsers which are capable of simultaneously performing segmentation (parsing an image into regions) and recognition (labeling the regions). Firstly, it is unclear what hierarchical representations should be used to model images and there are no direct analogies to the syntactic categories and phrase structures that occur in speech. Secondly, the inference problem is formidable due to the well-known complexity and ambiguity of segmentation and recognition. Unlike most languages (Chinese is an exception), whose constituents are well-separated words, the boundaries between different image

regions are usually highly unclear. Exploring all the different image partitions results in combinatorial explosions because of the two-dimensional nature of images (which makes it impossible to order these partitions to enable dynamic programming). Overall it has been hard to adapt methods from natural language parsing and apply them to vision despite the high-level conceptual similarities (except for restricted problems such as text [102]).

Attempts at image parsing must make trade-offs between the complexity of the models and the complexity of the computation (for inference and learning). Broadly speaking, recent attempts can be divided into two different styles. The first style emphasizes the modeling problem and develops stochastic grammars [103, 30] capable of representing a rich class of visual relationships and conceptual knowledge about objects, scenes, and images. This style of research pays less attention to the complexity of computation. Learning is usually performed, if at all, only for individual components of the models. Parsing is performed by MCMC sampling and is only efficient provided effective proposal probabilities can be designed [30]. The second style builds on the success of conditional random fields (CRF's)[23] and emphasizes efficient computation. This yields simpler (discriminative) models which are less capable of representing complex image structures and long range interactions. Efficient inference (e.g. belief propagation and graph-cuts) and learning (e.g. AdaBoost, MLE) are available for basic CRF's and make these methods attractive. But these inference algorithms become less effective, and can fail, if we attempt to make the CRF models more powerful. For example, TextonBoost [21] requires the parameters of the CRF to be tuned manually. Overall, it seems hard to extend the CRF style methods to include long-range relationships and contextual knowledge without significantly altering the models and the algorithms.

In this chapter, we introduce Hierarchical Image Models (HIM)'s for image pars-

ing. HIM's balance the trade-off between model and inference complexity by introducing a hierarchy of hidden states. In particular, we introduce *segmentation templates* which represent complex image knowledge and serve as elementary constituents analogous to those used in speech. As in speech, we can recursively compose these constituents at lower levels to form more complex constituents at higher level. Each node of the hierarchy corresponds to an image region (whose size depends on the level in the hierarchy). The state of each node represents both the partitioning of the corresponding region into segments and the labeling of these segments (i.e. in terms of objects). Segmentations at the top levels of the hierarchy give coarse descriptions of the image which are refined by the segmentations at the lower levels. Learning and inference (parsing) are made efficient by exploiting the hierarchical structure (and the absence of loops). In short, this novel architecture offers two advantages: (I) Representation – the hierarchical model using segmentation templates is able to capture long-range dependency and exploiting different levels of contextual information, (II) Computation – the hierarchical tree structure enables rapid inference (polynomial time) and learning by variants of dynamic programming (with pruning) and the use of machine learning (e.g. structured perceptrons[57]).

To illustrate the HIM we implement it for parsing images and we evaluate it on the public MSRC image dataset [21]. Our results show that the HIM outperforms the other state-of-the-art approaches. We discuss ways that HIM's can be extended naturally to model more complex image phenomena.

7.2 Hierarchical Image Model

7.2.1 The Model

We represent an image by a hierarchical graph defined by parent-child relationships. See figure 7.1. The hierarchy corresponds to the image pyramid (with 5 layers in this chapter). The top node of the hierarchy represents the whole image. The intermediate nodes represent different sub-regions of the image. The leaf nodes represent local image patches (27×27 in this chapter). We use a to index nodes of the hierarchy. A node a has only one parent node denoted by $Pa(a)$ and four child nodes denoted by $Ch(a)$. Thus, the hierarchy is a quad tree and $Ch(a)$ encodes all its vertical edges. The image region represented by node a is denoted by $R(a)$. A pixel in $R(a)$, indexed by r, corresponds to an image pixel. The set of pairs of neighbor pixels in $R(a)$ is denoted by $E(a)$.

A configuration of the hierarchy is an assignment of state variables $y = \{y_a\}$ with $y_a = (s_a, c_a)$ at each node a, where s and c denote region partition and object labeling, respectively and (s, c) is called the "Segmentation and Recognition" pair, which we call an ***S-R pair***. All state variables are unobservable. More precisely, each region $R(a)$ is described by a *segmentation templates* which is selected from a dictionary D_S. Each segmentation template consists of a partition of the region into K non-overlapping sub-parts, see figure 7.1. In this chapter $K \leq 3$, $|D_s| = 30$, and the segmentation templates are designed by hand to cover the taxonomy of shape segmentations that happen in images, such as T-junctions, Y-junctions, and so on. The variable s refers to the indexes of the segmentation templates in the dictionary, i.e., $s_a \in \{1..|D_s|\}$. c gives the object labels of K sub-parts (i.e. labels one sub-part as "horse" another as "dog" and another as "grass"). Hence c_a is a K-dimension vector whose components take values $1, ..., M$ where M is the number of object classes. The labeling of a pixel r in

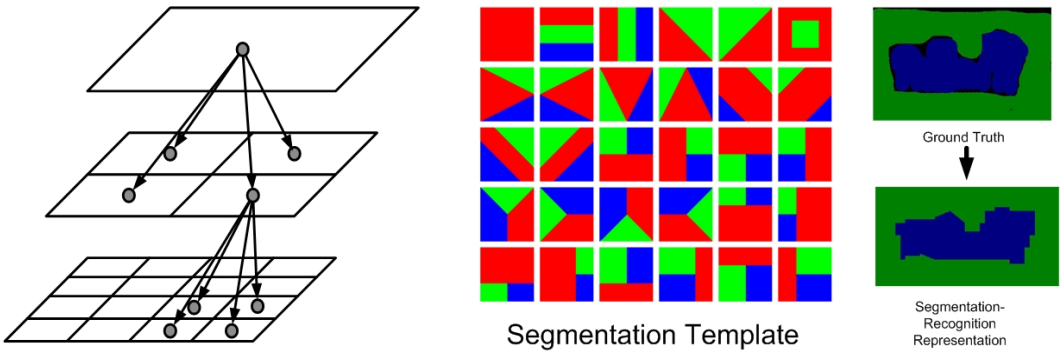


Figure 7.1: The left panel shows the structure of the Hierarchical Image Model. The grey circles are the nodes of the hierarchy. All nodes, except the top node, have only one parent nodes. All nodes except the leafs are connected to four child nodes. The middle panel shows a dictionary of 30 segmentation templates. The color of the sub--parts of each template indicates the object class. Different sub-parts may share the same label. For example, three sub-parts may have only two distinct labels. The last panel shows that the ground truth pixel labels (upper right panel) can be well approximated by composing a set of labeled segmentation templates (bottom right panel).

region $R(a)$ is denoted by $o_a^r \in \{1..M\}$ and is directly obtained from s_a, c_a. Any two pixels belonging to the same sub-part share the same label. The labeling o_a^r is defined at the level of node a. In other words, each level of the hierarchy has a separate labeling field. We will show how our model encourages the labelings o_a^r at different levels to be consistent.

A novel feature of this hierarchical representation is the multi-level *S-R pairs* which explicitly model both the segmentation and labeling of its corresponding region, while traditional vision approaches [21, 104, 105] use labeling only. The S-R pairs defined in a hierarchical form provide a coarse-to-fine representation which captures the "gist" (semantical meaning) of image regions. As one can see in figure 7.2, the global S-R pair gives a coarse description (the identities of objects and their spatial layout) of the whole image which is accurate enough to encode high level image properties in a compact form. The mid-level one represents the leg of a horse roughly. The four templates at the lower level further refine the interpretations. We will show this approximation quality empirically in section 7.3.

The conditional distribution over all the states is given by:

$$p(y|x;\alpha) = \frac{1}{Z(x;\alpha)} \exp\{-E_1(x,s,c;\alpha_1) - E_2(x,s,c;\alpha_2) - E_3(s,c;\alpha_3) \quad (7.1)$$
$$-E_4(c;\alpha_4) - E_5(s;\alpha_5) - E_6(s,c;\alpha_6)\}$$

where x refers to the input image, y is the parse tree, α are the parameters to be estimated, $Z(x;\alpha)$ is the partition function and $E_i(x,y)$ are energy terms. Equivalently, the conditional distribution can be reformulated in a log-linear form:

$$\log p(y|x;\alpha) = \psi(x,y) \cdot \alpha - \log Z(x;\alpha) \quad (7.2)$$

Each energy term is of linear form, $E_i(x,y) = -\psi_i(x,y) \cdot \alpha_i$, where the inner product is calculated on potential functions defined over the hierarchical structure. There are six types of energy terms defined as follows.

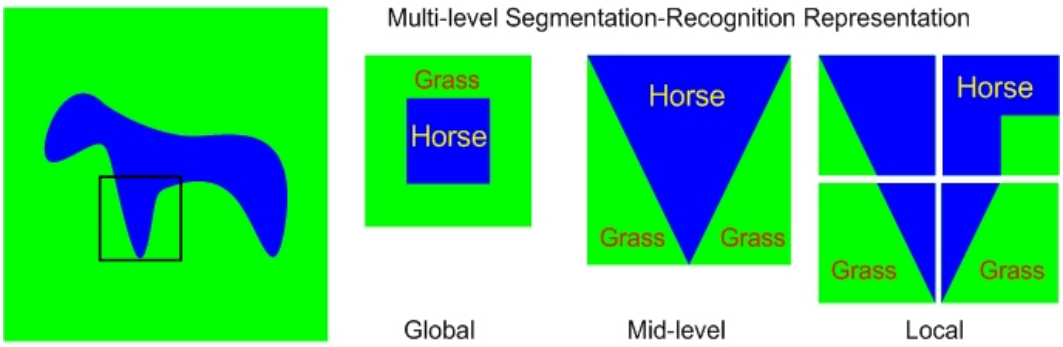


Figure 7.2: This figure illustrates how the segmentation templates and object labels (S-R pair) represent image regions in a coarse-to-fine way. The left figure is the input image which is followed by global, mid-level and local S-R pairs. The global S-R pair gives a coarse description of the object identity (horse), its background (grass), and its position in the image (central). The mid-level S-R pair corresponds to the region bounded by the black box in the input image. It represents (roughly) the shape of the horse's leg. The four S-R pairs at the lower level combine to represent the same leg more accurately.

The first term $E_1(x, s, c)$ is a object specific data term which represents image features of regions. We set $E_1(x, s, c) = -\sum_a \alpha_1 \psi_1(x, s_a, c_a)$ where $\sum_a$ is the summation over all nodes at different levels of the hierarchy, and $\psi_1(x, s_a, c_a)$ is of the form:

$$\psi_1(x, s_a, c_a) = \frac{1}{|R(a)|} \sum_{r \in R(a)} \log p(o_a^r | x) \tag{7.3}$$

where $p(o_a^r|x) = \frac{\exp\{F(x^r, o_a^r)\}}{\sum_{o'} \exp\{F(x^r, o')\}}$, x^r is a local image region centered at the location of r, and $F(\cdot, \cdot)$ is a strong classifier output by multi-class boosting [106]. The image features used by the classifier (47 in total) are the greyscale intensity, the color (R,G, B channels), the intensity gradient, the Canny edge, the response of DOG (difference of Gaussians) and DOOG (Difference of Offset Gaussian) filters at different scales (13*13 and 22*22) and orientations (0,30,60,...), and so on. We use 55 types of shape (spatial) filters [21] to calculate the responses of 47 image features. There are $2585 = 47 * 55$ features in total.

The second term (segmentation specific) $E_2(x, s, c) = -\sum_a \alpha_2 \psi_2(x, s_a, c_a)$ is designed to favor the segmentation templates in which the pixels belonging to the same partitions (i.e., having the same labels) have similar appearance. We define:

$$\psi_2(x, s_a, c_a) = \frac{1}{|E(a)|} \sum_{(q,r) \in E(a)} \phi(x^r, x^q | o_a^r, o_a^q) \tag{7.4}$$

where $E(a)$ are the set of edges connecting pixels q, r in a neighborhood and $\phi(x^r, x^q|o_a^r, o_a^q)$ has the form of $\phi(x^r, x^q|o_a^r, o_a^q) = \begin{cases} \gamma(r,q) & if\ o_a^r = o_a^q \\ 0 & if\ o_a^r \neq o_a^q \end{cases}$, where $\gamma(r, q) = \lambda \exp\{-\frac{g^2(r,q)}{2\gamma^2}\}$ $\frac{1}{dist(r,q)}$, $g(.,.)$ is a distance measure on the colors x^r, x^q and $dist(r, q)$ measures the spatial distance between r and q. $\phi(x^r, x^q|o_a^r, o_a^q)$ is so called the contrast sensitive Potts model which is widely used in graph-cut algorithms [107] as edge potentials (only in one level) to favors pixels with similar colour having the same labels.

The third term, defined as $E_3(s,c) = -\sum_{a,b=Pa(a)} \alpha_3 \psi_3(s_a, c_a, s_b, c_b)$ (i.e. the nodes a at all levels are considered and b is the parent of a) is proposed to encourage the consistency between the configurations of every pair of parent-child nodes in two consecutive layers. $\psi_3(s_a, c_a, s_b, c_b)$ is defined by the Hamming distance:

$$\psi_3(s_a, c_a, s_b, c_b) = \frac{1}{|R(a)|} \sum_{r \in R(a)} \delta(o_a^r, o_b^r) \tag{7.5}$$

where $\delta(o_a^r, o_b^r)$ is the Kronecker delta, which equals one whenever $o_a^r = o_b^r$ and zero otherwise. The hamming function ensures to glue the segmentation templates (and their labels) at different levels together in a consistent hierarchical form. This energy term is a generalization of the interaction energy in the Potts model. However, $E_3(s,c)$ has a hierarchical form which allows multi-level interactions.

The fourth term $E_4(c)$ is designed to model the co-occurrence of two object classes (e.g., a cow is unlikely to appear next to an aeroplane):

$$E_4(c) = -\sum_{a} \sum_{i,j=1..M} \alpha_4(i,j)\psi_4(i,j,c_a,c_a) - \sum_{a,b=Pa(a)} \sum_{i,j=1..M} \alpha_4(i,j)\psi_4(i,j,c_a,c_b) \tag{7.6}$$

where $\psi_4(i,j,c_a,c_b)$ is an indicator function which equals one while $i \equiv c_a$ and $j \equiv c_b$ ($i \equiv c_a$ means i is a component of c_a) hold true and zero otherwise. α_4 is a matrix where each entry $\alpha_4(i,j)$ encodes the compatibility between two classes i and j. The first term on the r.h.s encodes the classes in a single template while the second term encodes the classes in two templates of the parent-child nodes. It is worth noting that class dependency is encoded at all levels to capture both short-range and long-range interactions.

The fifth term $E_5(s) = -\sum_a \alpha_5 \psi_5(s_a)$, where $\psi_5(s_a) = \log p(s_a)$ encode the generic prior of the segmentation template. Similarly the sixth term $E_6(s,c) = -\sum_a \sum_{j \equiv c_a} \alpha_6 \psi_6(s_a, j)$, where $\psi_6(s_a, j) = \log p(s_a, j)$, models the co-occurrence of the

segmentation templates and the object classes. $\psi_5(s_a)$ and $\psi_6(s_a, j)$ are directly obtained from training data by label counting. The parameters α_5 and α_6 are both scalars.

Justifications. The HIM has several partial similarities with other work. HIM is a coarse-to-fine representation which captures the "gist" of image regions by using the S-R pairs at multiple levels. But the traditional concept of "gist" [108] relies only on image features and does not include segmentation templates. HIM also looks slightly like a hierarchical version of the jigsaw puzzle approach [109] which does not have segmentation templates (or a hierarchy). Levin and Weiss [4] use a segmentation mask which is more object-specific than our segmentation templates (and they do not have a hierarchy). It is worth nothing that, in contrast to TextonBoost [21], we do not use "location features" in order to avoid the dangers of overfitting to a restricted set of scene layouts. Our approach has some similarities to some hierarchical models (which have two-layers only) [104],[105] – but these models also lack segmentation templates. The hierarchial model proposed by [110] is an interesting alternative but which does not perform explicit segmentation.

7.2.2 Parsing by Dynamic Programming

Parsing an image is performed as inference of the HIM. More precisely, the task of parsing is to obtain the maximum a posterior (MAP):

$$y^* = \arg\max_y p(y|x;\alpha) = \arg\max_y \psi(x,y)\cdot\alpha \tag{7.7}$$

The size of the states of each node is $O(M^K|D_s|)$ where $K = 3, M = 21, |D_s| = 30$ in our case. Since the form of y is a tree, Dynamic Programming (DP) can be applied to calculate the best parse tree y^* according to equation 7.7. Note that the pixel label o_a is determined by (s, c), so we only need consider a subset of pixel labelings. It is unlike flat MRF representation where we need to do exhaustive search over all

pixel labels o (which would be impractical for DP). The final output of the model for segmentation is the pixel labeling determined by the (s, c) of the lowest level.

It is straight forward to see that the computational complexity of DP is $O(M^{2K}|D_s|^2 H)$ where H is the number of edges of the hierarchy. Although DP can be performed in polynomial time, the huge number of states make exact DP still impractical. Therefore, we resort to a pruned version of DP similar to the method described in section 4.4 in chapter 4. Pruned DP has also been applied to language parsing [111] as a standard way to deal with large state spaces. For brevity we omit the details.

7.2.3 Learning the Model

Since HIM is a conditional model, in principle, estimation of its parameters can be achieved by any discriminative learning approach, such as maximum likelihood learning as used in Conditional Random Field (CRF)[23], max-margin learning [84], and structure-perceptron learning[57]. In this chapter, we adopt the structure-perceptron learning which has applied for learning the recursive deformable template (see chapter 4). Note that structure-perceptron learning is simple to implement and only needs to calculate the most probable configurations (parses) of the model. By contrast, maximum likelihood learning requires calculating the expectation of features which is difficult due to the large states of HIM. Therefore, structure-perceptron learning is more flexible and computationally simpler. Moreover, Collins [57] proved theoretical results for convergence properties, for both separable and non-separable cases, and for generalization.

The structure-perceptron learning will not compute the partition function $Z(x; \alpha)$. Therefore we do not have a formal probabilistic interpretation. The goal of structure-perceptron learning is to learn a mapping from inputs $x \in X$ to output structure $y \in Y$. In our case, X is a set of images, with Y being a set of possible parse trees which

specify the labels of image regions in a hierarchical form. It seems that the ground truth of parsing trees needs all labels of both segmentation template and pixel labelings. In our experiment, we will show that how to obtain the ground truth directly from the segmentation labels without extra human labeling. We use a set of training examples $\{(x_i, y_i) : i = 1...n\}$ and a set of functions ψ which map each $(x, y) \in X \times Y$ to a feature vector $\psi(x, y) \in R^d$. The task is to estimate a parameter vector $\alpha \in R^d$ for the weights of the features. The feature vectors $\psi(x, y)$ can include arbitrary features of parse trees, as we discussed in section 7.2.1. The loss function used in structure-perceptron learning is usually of form:

$$Loss(\alpha) = \psi(x, y) \cdot \alpha - \max_{\overline{y}} \psi(x, \overline{y}) \cdot \alpha, \tag{7.8}$$

where y is the correct structure for input x, and $\overline{y}$ is a dummy variable.

The basic structure-perceptron algorithm is designed to minimize the loss function. We adapt *"the averaged parameters"* version whose pseudo-code is given in figure 7.3. The algorithm proceeds in a simple way (similar to the perceptron algorithm for classification). The parameters are initialized to zero and the algorithm loops over the training examples. If the highest scoring parse tree for input x is not correct, then the parameters α are updated by an additive term. The most difficult step of the method is finding $y^* = \arg\max_y \psi(x^i, y) \cdot \alpha$. This is precisely the parsing (inference) problem. Hence the practicality of structure-perceptron learning, and its computational efficiency, depends on the inference algorithm. As discussed earlier, see section 7.2.2, the inference algorithm has polynomial computational complexity for an HIM which makes structure-perceptron learning practical for HIM. The averaged parameters are defined to be $\gamma = \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha^{t,i}/NT$, where T is the number of epochs, NT is the total number of iterations. It is straightforward to store these averaged parameters and output them as the final estimates.

Input: A set of training images with ground truth (x^i, y^i) for $i = 1..N$. Initialize parameter vector $\alpha = 0$.

For $t = 1..T, i = 1..N$

- find the best state of the model on the i'th training image with current parameter setting, i.e., $y^* = \arg\max_y \psi(x^i, y) \cdot \alpha$
- Update the parameters: $\alpha = \alpha + \psi(x^i, y^i) - \psi(x^i, y^*)$
- Store: $\alpha^{t,i} = \alpha$

Output: Parameters $\gamma = \sum_{t,i} \alpha^{t,i}/NT$

Figure 7.3: Structure-perceptron learning

7.3 Experimental Results

Dataset. We use a standard public dataset, the MSRC 21-class Image Dataset [21], to perform experimental evaluations for the HIM. This dataset is designed to evaluate scene labeling including both image segmentation and multi-class object recognition. The ground truth only gives the labeling of the image pixels. To supplement this ground truth (to enable learning), we estimate the true labels (states of the S-R pair) of the nodes in the five-layer hierarchy of HIM by selecting the S-R pairs which have maximum overlap with the labels of the image pixels. This approximation only results in 2% error in labeling image pixels. There are a total of 591 images. We use the identical splitting as [3], i.e., 45% for training, 10% for validation, and 45% for testing. The parameters learnt from the training set, with the best performance on validation set, are selected.

Implementation Details. For a given image x, the parsing result is obtained by estimating the best configuration y^* of the HIM. To evaluate the performance of parsing

we use the **global accuracy** measured in terms of all pixels and the **average accuracy** over the 21 object classes (global accuracy pays most attention to frequently occurring objects and penalizes infrequent objects). A computer with 8 GB memory and 2.4 GHz CPU was used for training and testing. For each class, there are around $4,500$ weak classifiers selected by multi-class boosting. The boosting learning takes about 35 hours of which 27 hours are spent on I/O processing and 8 hours on computing. The structure-perceptron learning takes about 20 hours to converge in $5520(T = 20, N = 276)$ iterations. In the testing stage, it takes 30 seconds to parse an image with size of 320×200 (6s for extracting image features, 9s for computing the strong classifier of boosting and 15s for parsing the HIM).

Results. Figure 7.4 (best viewed in color) shows several parsing results obtained by the HIM and by the classifier by itself (i.e. $p(o_a^r|x)$ learnt by boosting). One can see that the HIM is able to roughly capture different shaped segmentation boundaries (see the legs of the cow and sheep in rows 1 and 3, and the boundary curve between sky and building in row 4). Table 7.1 shows that HIM improves the results obtained by the classifier by 7.3% for average accuracy and 5.5% for global accuracy. In particular, in rows 6 and 7 in figure 7.4, one can observe that boosting gives many incorrect labels. It is impossible to correct such large mislabeled regions without the long-range interactions in the HIM, which improves the results by 20% and 32%.

Comparisons. In table 7.1, we compare the performance of our approach with other successful methods [3, 112, 113]. Our approach outperforms those alternatives by 10% in average accuracy and 6% in global accuracy. Note that no other method perform consistently in both measures. Our boosting results are better than Textonboost [3] because of image features. Would we get better results if we use a CRF with our boosting instead of a hierarchy? We argue that we wouldn't because the CRF only improves TextonBoost's performance by 3 percent [3], while we gain 6 percent

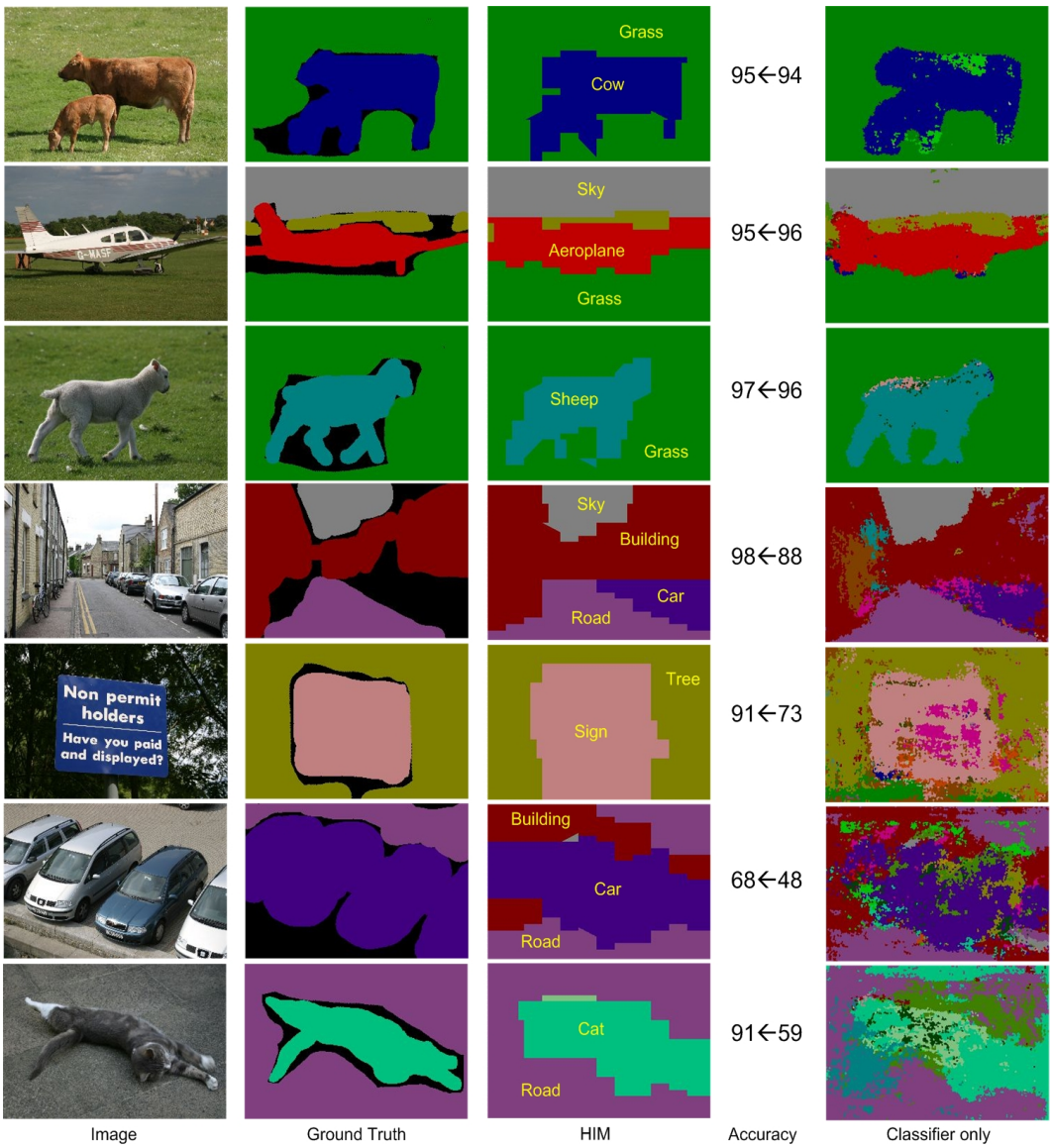


Figure 7.4: This figure is best viewed in color. The colors indicate the labels of 21 object classes as in [3]. The columns (except the fourth "accuracy" column) show the input images, ground truth, the labels obtained by HIM and the boosting classifier respectively. The "accuracy" column shows the global accuracy obtained by HIM (left) and the boosting classifier (right). In these 7 examples, HIM improves boosting by 1%, -1% (an outlier!), 1%, 10%, 18%, 20% and 32% in terms of global accuracy.

	Textonboost[3]	PLSA-MRF [112]	Mspatch [113]	Classifier	HIM
Average	57.7	64.0	61.8	67.2	74.5
Global	72.2	73.5	75.1	75.9	81.4

Table 7.1: Performance Comparisons for average accuracy and global accuracy. "Classifier only" are the results where the pixel labels are predicted by the classifier obtained by boosting only.

by using the hierarchy (and we start with a higher baseline). Some other methods [114, 105, 104], which are worse than [112, 113] and evaluated on simpler datasets [104, 105] (less than 10 classes), are not listed here due to lack of space. In summary, our results are significantly better than the state-of-the-art methods.

Diagnosis on the function of S-R Pair. Figure 7.5 shows how the S-R pairs (which include the segmentation templates) can be used to (partially) parse an object into its constituent parts, by the correspondence between S-R pairs and specific parts of objects. We plot the states of a subset of S-R pairs for some images. For example, the S-R pair consisting of two horizontal bars labeled "cow" and "grass" respectively indicates the cow's stomach consistently across different images. Similarly, the cow's tail can be located according to the configuration of another S-R pair with vertical bars. In principle, the whole object can be parsed into its constituent parts which are aligned consistently. Developing this idea further is an exciting aspect of our current research.

7.4 Conclusion

This chapter describes a novel hierarchical image model (HIM) for 2D image parsing. The hierarchical nature of the model, and the use of segmentation templates, enables the HIM to represent complex image structures in a coarse-to-fine manner. We can

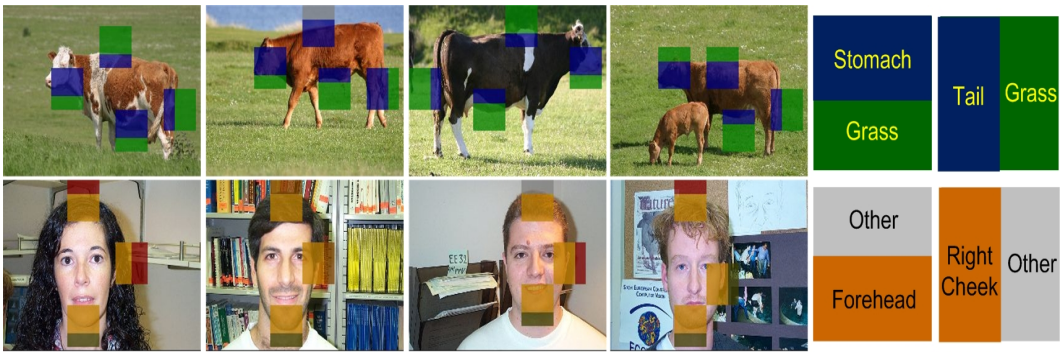


Figure 7.5: The S-R pairs can be used to parse the object into parts. The colors indicate the identities of objects. The shapes (spacial layout) of the segmentation templates distinguish the constituent parts of the object. Observe that the same S-R pairs (e.g. stomach above grass, and tail to the left of grass) correspond to the same object part in different images.

perform inference (parsing) rapidly in polynomial time by exploiting the hierarchical structure. Moreover, we can learn the HIM probability distribution from labeled training data by adapting the structure-perceptron algorithm. We demonstrated the effectiveness of HIM's by applying them to the challenging task of segmentation and labeling of the public MSRC image database. Our results show that we outperform other state-of-the-art approaches.

The design of the HIM was motivated by drawing parallels between language and vision processing. We have attempted to capture the underlying spirit of the successful language processing approaches – the hierarchical representations based on the recursive composition of constituents and efficient inference and learning algorithms. Our current work attempts to extend the HIM's to improve their representational power while maintaining computational efficiency.

Part V

Concluding Remarks

CHAPTER 8

Conclusions and Future Directions

In this thesis, we have shown how the recursion and composition principle can be applied to the deformable object modeling and image understanding. For deformable object modeling, we proposed the recursive deformable template model, and presented both supervised and unsupervised methods for learning the model. We also extended this model to an AND/OR graph for articulated object parsing. For image understanding, we proposed the recursive segmentation and recognition template model. We demonstrated the effectiveness and versatility of our framework by applying it to very different problems (deformable object detection, segmentation, parsing, and image segmentation and scene labeling), evaluating it on large datasets, and giving comparisons to the state of the art.

The key idea behind our approaches is the the recursion and composition which results in a hierarchical representation. The hierarchies are capable of modeling both short- and long- range visual relationship. The hierarchical design also allows us to apply dynamic programming for polynomial-time inference and supervised machine learning. Recursive Composition combined with suspicious coincidence and competitive exclusion leads to efficient supervised learning.

In the future work, we are interested to extend our recursive and compositional system to learn deep-structured visual vocabulary for multiple objects simultaneously, and design a new image parser based on the unified visual representation.

REFERENCES

[1] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Obj cut," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 18–25.

[2] P. Srinivasan and J. Shi, "Bottom-up recognition and parsing of the human body," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[3] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context." in *International Journal of Computer Vision*, 2007.

[4] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 581–594.

[5] J. Tenenbaum and A. Yuille, *IPAM Summer School: The Mathematics of the Mind.* IPAM, UCLA., 2008.

[6] L. Valiant, *Circuits of the Mind.* Oxford University Press, 1994.

[7] L. Zhu and A. L. Yuille, "A hierarchical compositional system for rapid object detection," in *Advances in Neural Information Processing Systems*, 2005.

[8] L. Zhu, Y. Chen, and A. L. Yuille, "Unsupervised learning of a probabilistic grammar for object detection and parsing," in *Advances in Neural Information Processing Systems*, 2006, pp. 1617–1624.

[9] ——, "Unsupervised learning of probabilistic grammar-markov models for object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[10] Y. Chen, L. Zhu, A. L. Yuille, and H. Zhang, "Unsupervised learning of probabilistic object models for object classification, segmentation and recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[11] Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang, "Rapid inference on a novel and/or graph for object detection, segmentation and parsing," in *Advances in Neural Information Processing Systems*, 2007.

[12] L. Zhu, Y. Chen, X. Ye, and A. L. Yuille, "Structure-perceptron learning of a hierarchical log-linear model," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[13] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. L. Yuille, "Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion," in *Proceedings of The 10th European Conference on Computer Vision*, 2008.

[14] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille, "Max margin and/or graph learning for parsing the human body," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[15] L. Zhu, Y. Chen, Y. Lin, and A. L. Yuille, "A hierarchical image model for polynomial-time 2d parsing," in *Advances in Neural Information Processing System*, 2008.

[16] X. Ren, C. Fowlkes, and J. Malik, "Cue integration for figure/ground labeling," in *Advances in Neural Information Processing Systems*, 2005.

[17] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Proceedings of Conference on Computer Vision and Image Understanding*, vol. 106, pp. 59–70, 2007.

[18] E. Borenstein and S. Ullman, "Class-specific top-down segmentation," in *Proceedings of European Conference on Computer Vision*, 2002, pp. 109–124.

[19] S. Z. Li, H. Zhang, S. Yan, and Q. Cheng, "Multi-view face alignment using direct appearance models," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, 2002, pp. 324–329.

[20] G. Mori, "Guiding model search using segmentation," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 1417–1423.

[21] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 1–15.

[22] D. A. McAllester, M. Collins, and F. Pereira, "Case-factor diagrams for structured probabilistic modeling," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2004, pp. 382–391.

[23] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of International Conference on Machine Learning*, 2001, pp. 282–289.

[24] D. Klein and C. D. Manning, "Natural language grammar induction using a constituent-context model," in *Advances in Neural Information Processing Systems*, 2001, pp. 35–42.

[25] R. Dechter and R. Mateescu, "And/or search spaces for graphical models," *Artificial Intelligence*, vol. 171, no. 2-3, pp. 73–106, 2007.

[26] H. Chen, Z. Xu, Z. Liu, and S. C. Zhu, "Composite templates for cloth modeling and sketching," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 943–950.

[27] L. S. Zettlemoyer and M. Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 658–666.

[28] B. D. Ripley, *Pattern Recognition and Neural Networks*. New York, NY, USA: Cambridge University Press, 1996.

[29] C. Manning and H. Schuetze, *Foundations of statistical natural language processing*. Cambridge, Mass, USA: MIT Press, 1999.

[30] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," in *Proceedings of IEEE International Conference on Computer Vision*, 2003, pp. 18–25.

[31] H. Barlow, "Unsupervised learning," *Neural Computation*, vol. 1, pp. 295–311, 1989.

[32] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 264–271.

[33] ——, "A sparse object category model for efficient learning and exhaustive recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 380–387.

[34] D. J. Crandall and D. P. Huttenlocher, "Weakly supervised learning of part-based spatial models for visual object recognition," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 16–29.

[35] D. J. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher, "Spatial priors for part-based recognition using statistical models," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 10–17.

[36] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka, "Visual categorization with bags of keypoints," in *Proceedings of European Conference on Computer Vision International Workshop on Statistical Learning in Computer Vision*, 2004.

[37] M. Meila and M. I. Jordan, "Learning with mixtures of trees," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, 2000.

[38] S. D. Pietra, V. J. D. Pietra, and J. D. Lafferty, "Inducing features of random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.

[39] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, no. 8, pp. 1627–1660, 1997.

[40] A. McCallum, "Efficiently inducing features of conditional random fields," in *Conference on Uncertainty in Artificial Intelligence*, 2003, pp. 403–410.

[41] N. Friedman, "The bayesian structural em algorithm," in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 129–138.

[42] L. Shams and C. von der Malsburg, "Are object shape primitives learnable?" *Neurocomputing*, vol. 26-27, pp. 855–863, 1999.

[43] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman, "Dataset issues in object recognition," in *Toward Category-Level Object Recognition*, 2006, pp. 29–48.

[44] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *Computational Statistics Quaterly*, vol. 4, pp. 269–282, 1990.

[45] T. Kadir and M. Brady, "Saliency, scale and image description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[46] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[47] S. Lazebnik, C. Schmid, and J. Ponce, "Semi-local affine parts for object recognition," in *Proceedings of IEEE International Workshop on Biologically Motivated Computer Vision*, 2004.

[48] Y. Amit and D. Geman, "A computational model for visual selection," *Neural Computation*, vol. 11, no. 7, pp. 1691–1715, 1999.

[49] Y. Jin and S. Geman, "Context and hierarchy in a probabilistic image model," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2145–2152.

[50] J. M. Coughlan, A. L. Yuille, C. English, and D. Snow, "Efficient deformable template detection and localization without user initialization," *Proceedings of Conference on Computer Vision and Image Understanding*, pp. 303–319, 2000.

[51] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," pp. 355–368, 1999.

[52] J. M. Coughlan and S. J. Ferreira, "Finding deformable shapes using loopy belief propagation," in *Proceedings of European Conference on Computer Vision*, 2002, pp. 453–468.

[53] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000, pp. 2044–2051.

[54] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4.

[55] P. A. Viola and M. J. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Advances in Neural Information Processing Systems*, 2001, pp. 1311–1318.

[56] ——, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[57] M. Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *Proceedings of Annual Meeting on Association for Computational Linguistics conference on Empirical methods in natural language processing*, 2002, pp. 1–8.

[58] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Proceedings of European Conference on Computer Vision'04 Workshop on Statistical Learning in Computer Vision*, 2004, pp. 17–32.

[59] H. Li, S.-C. Yan, and L.-Z. Peng, "Robust non-frontal face alignment with edge based texture," *Journal of Computer Science and Technology*, vol. 20, no. 6, pp. 849–854, 2005.

[60] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *Proceedings of European Conference on Computer Vision*, 1998, pp. 484–498.

[61] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[62] Y. Amit, D. Geman, and X. Fan, "A coarse-to-fine strategy for multiclass shape detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 12, pp. 1606–1621, 2004.

[63] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 994–1000.

[64] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[65] Z. Tu and A. L. Yuille, "Shape matching and recognition - using generative models and informative features," in *Proceedings of European Conference on Computer Vision*, 2004, pp. 195–209.

[66] P. F. Felzenszwalb and J. D. Schwartz, "Hierarchical matching of deformable shapes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[67] J. M. Winn and N. Jojic, "Locus: Learning object classes with unsupervised segmentation," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 756–763.

[68] T. Cour and J. Shi, "Recognizing objects by piecing together the segmentation puzzle," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[69] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000, pp. 1070–1077.

[70] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.

[71] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron," in *Proceedings of Annual Meeting on Association for Computational Linguistics*, 2001, pp. 263–270.

[72] M. Collins and B. Roark, "Incremental parsing with the perceptron algorithm," in *Proceedings of Annual Meeting on Association for Computational Linguistics*, 2004, p. 111.

[73] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[74] E. Borenstein and J. Malik, "Shape guided object segmentation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 969–976.

[75] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[76] B. Epshtein and S. Ullman, "Feature hierarchies for object classification," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 220–227.

[77] N. Ahuja and S. Todorovic, "Learning the taxonomy and models of categories present in arbitrary images," in *Proceedings of IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

[78] F. Fleuret and D. Geman, "Coarse-to-fine face detection," *International Journal of Computer Vision*, vol. 41, pp. 85–107, 2001.

[79] S. Fidler and A. Leonardis, "Towards scalable representations of object categories: Learning a hierarchy of parts," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[80] K. M. Bryan Russell, Antonio Torralba and W. Freeman, "Labelme: a database and web-based tool for image annotation," *Technical Report*, 2005.

[81] X. Chen and A. Yuille, "A time-efficient cascade for real-time object detection:. with applications for the visually impaired," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[82] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden markov support vector machines," in *Proceedings of International Conference on Machine Learning*, 2003, pp. 3–10.

[83] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *Advances in Neural Information Processing Systems*, 2003.

[84] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning, "Max-margin parsing," in *Proceedings of Annual Meeting on Association for Computational Linguistics conference on Empirical methods in natural language processing*, 2004.

[85] L. Sigal and M. J. Black, "Measure locally, reason globally: Occlusion-sensitive articulated pose estimation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2041–2048.

[86] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people," in *Proceedings of European Conference on Computer Vision*, 2002, pp. 700–714.

[87] X. Ren, A. C. Berg, and J. Malik, "Recovering human body configurations using pairwise constraints between parts," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 824–831.

[88] D. Ramanan, "Learning to parse images of articulated bodies," in *Advances in Neural Information Processing Systems*, 2006, pp. 1129–1136.

[89] M. W. Lee and I. Cohen, "Proposal maps driven mcmc for estimating human body pose in static images," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2)*, 2004, pp. 334–341.

[90] J. Zhang, J. Luo, R. T. Collins, and Y. Liu, "Body localization in still images using hierarchical models and hybrid search," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1536–1543.

[91] P. Srinivasan and J. Shi, "Bottom-up recognition and parsing of the human body," in *Proceedings of International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2007, pp. 153–168.

[92] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[93] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[94] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 130–136.

[95] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, 2000.

[96] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of International Conference on Machine Learning*, 2004.

[97] J. C. Platt, "Using analytic qp and sparseness to speed training of support vector machines," in *Advances in Neural Information Processing Systems*, 1998, pp. 557–563.

[98] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 326–333.

[99] F. Jelinek and J. D. Lafferty, "Computation of the probability of initial substring generation by stochastic context-free grammars," *Computational Linguistics*, vol. 17, no. 3, pp. 315–323, 1991.

[100] M. Collins, "Head-driven statistical models for natural language parsing," *Ph.D. Thesis, University of Pennsylvania*, 1999.

[101] K. Lari and S. J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," in *Computer Speech and Languag*, 1990.

[102] M. Shilman, P. Liang, and P. A. Viola, "Learning non-generative grammatical models for document analysis," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 962–969.

[103] Z. Tu and S. C. Zhu, "Image segmentation by data-driven markov chain monte carlo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 657–673, 2002.

[104] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 695–702.

[105] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 1284–1291.

[106] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2000.

[107] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," in *Proceedings of IEEE International Conference on Computer Vision*, 2001, pp. 105–112.

[108] A. Oliva and A. Torralba, "Building the gist of a scene: the role of global image features in recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 155, pp. 23–36, 2006.

[109] A. Kannan, J. M. Winn, and C. Rother, "Clustering appearance and shape by learning jigsaws," in *Advances in Neural Information Processing Systems*, 2006, pp. 657–664.

[110] E. B. Sudderth, A. B. Torralba, W. T. Freeman, and A. S. Willsky, "Learning hierarchical models of scenes, objects, and parts," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 1331–1338.

[111] E. Charniak and M. Johnson, "Coarse-to-fine n-best parsing and maxent discriminative reranking," in *Proceedings of Annual Meeting on Association for Computational Linguistics*, 2005.

[112] J. Verbeek and B. Triggs, "Region classification with markov field aspect models," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[113] L. Yang, P. Meer, and D. J. Foran, "Multiple class segmentation using a unified framework over mean-shift patches," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[114] J. Verbeek and B. Triggs, "Scene segmentation with crfs learned from partially labeled images," in *Advances in Neural Information Processing Systems*, vol. 20, 2008.