Title
Towards Scene Understanding: Object Detection, Segmentation, and Contextual Reasoning

Author
Mottaghi, Roozbeh

Publication Date
2013

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Towards Scene Understanding: Object Detection, Segmentation, and Contextual Reasoning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Roozbeh Mottaghi

2013

ABSTRACT OF THE DISSERTATION

Towards Scene Understanding: Object Detection, Segmentation, and Contextual Reasoning

by

Roozbeh Mottaghi

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Alan Yuille, Chair

Scene understanding is one of the holy grails of computer vision. Despite decades of research on scene understanding, it is still considered an unsolved problem. The difficulty arises mainly because of the huge space of possible images. We require models to capture this variability of scenes and their constituents (e.g., objects) given the limited memory resources. Additionally, we require efficient learning and inference techniques for our models to find the optimal solution in the enormous space of possible solutions.

In this thesis, we propose a set of novel techniques for object detection, segmentation, and contextual reasoning and take a further step towards the ultimate goal of holistic scene understanding. In particular, we propose a compositional method for representing objects and show inference can be performed for an exponential number of objects in linear time. Subsequently, we propose a series of discriminative learning methods for object detection and segmentation and show that our methods achieve the state-of-the-art performance on difficult benchmarks in the computer vision community. Finally, through a series of hybrid human-machine experiments, we try to identify bottlenecks in scene understanding to better guide future research efforts in this area.

The dissertation of Roozbeh Mottaghi is approved.

Demetri Terzopoulos

Zhuowen Tu

Yingnian Wu

Alan Yuille, Committee Chair

University of California, Los Angeles

2013

To my parents

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I owe a big thanks to Alan Yuille for being a wonderful advisor. It has been my great pleasure to work with him in the past couple of years. I have always been fascinated by his vision of future computer vision. I was very fortunate to work with Devi Parikh. It was one of the best experiences during my PhD studies (I never forget that we exchanged about 6000 emails during a three month internship). I am grateful to Raquel Urtusun and Sanja Fidler for being amazing collaborators and for teaching me how to think positively when everything goes wrong in my research. I would also like to thank all members of my PhD committee Demetri Terzopoulos, Zhuowen Tu, and Ying Nian Wu.

Many thanks to my fellow labmates especially George Papandreou for patiently describing things to me and being a great officemate during the entire PhD years and also Xianjie Chen for being a great collaborator. I would also like to thank my friends in Los Angeles with whom I had a lot fun during my PhD studies (in order of acquaintance): Hossein, Mina, Hamid, Mahdi, Mohsen, Niloofar, Armita, Afsoon, Mohsen, Shima, Niusha, Sepehr, Zhoubin, Mahnaz, Arash, Reza (I had a hard time to recall the order. Forgive me if the order is wrong or I missed your name).

All this would not have been possible without the continuous support of my parents. I could never thank them enough for their love, sacrifice, and encouragement. Thank you for being the best parents ever.

VITA

1999–2003	B.Sc. (Computer Engineering), Sharif University of Technology, Tehran, Iran
2003–2006	M.A.Sc. (Engineering Science), Simon Fraser University, Burnaby, BC, Canada
2006–2008	M.S. (Computer Science), Georgia Institute of Technology, Atlanta, GA

PUBLICATIONS

- **R. Mottaghi**, X. Chen, X. Liu, S. W. Lee, S. Fidler, R. Urtasun, and A. Yuille. *PASCAL meets Context: The Role of Context for Category Level Object Detection in the Wild*, submitted to International Conference on Computer Vision (ICCV), 2013.
- X. Chen, **R. Mottaghi**, X. Liu, N. Cho, S. Fidler, R. Urtasun, S. W. Lee, and A. Yuille. *Detect What You Can: A Multi-Component Model for Pose Variation, Occlusion, and Size*, submitted to International Conference on Computer Vision (ICCV), 2013.
- **R. Mottaghi**, S. Fidler, R. Urtasun, and D. Parikh. *Human-Machine Hybrid CRFs for Identifying Bottlenecks in Holistic Scene Understanding*, submitted to International Journal of Computer Vision (IJCV), 2013.
- A. Yuille, **R. Mottaghi**. *Complexity of Representation and Inference in Compositional Models with Part Sharing*, To appear in International Conference on Learning Representations (ICLR), 2013.
- **R. Mottaghi**, S. Fidler, J. Yao, R. Urtasun, and D. Parikh. *Analyzing Semantic Segmentation Using Human-Machine Hybrid CRFs*, To appear in Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

- S. Fidler, **R. Mottaghi**, A. Yuille, R. Urtasun. *Bottom-up Segmentation for Top-down Detection*, To appear in Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

- **R. Mottaghi**. *Augmenting Deformable Part Models with Irregular-shaped Object Patches*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

- **R. Mottaghi**, A. Ranganathan, and A. Yuille. *A Compositional Approach to Learning Part-based Models of Objects*, in International Conference on Computer Vision (ICCV), Workshop on 3D Representation and Recognition, 2011.

- J. Lee, **R. Mottaghi**, C. E. Pippin, and T. Balch. *Graph-based Planning Using Local Information for Unknown Outdoor Environments*, in International Conference on Robotics and Automation (ICRA), 2009.

- **R. Mottaghi**, M. Kaess, A. Ranganathan, R. Roberts. *Place Recognition based Fixed-lag Smoothing for Environments with Unreliable GPS*, in International Conference on Robotics and Automation (ICRA), 2008.

- **R. Mottaghi** and R. T. Vaughan. *An Integrated Particle Filter and Potential Field Method Applied to Cooperative Multi-Robot Target Tracking*, Autonomous Robots Journal, 23(1): 19-35, 2007.

- **R. Mottaghi** and R. T. Vaughan. *An Integrated Particle Filter & Potential Field Method for Cooperative Robot Target Tracking*, in International Conference on Robotics and Automation (ICRA), 2006, Orlando, Florida.

- **R. Mottaghi** and S. Payandeh. *An Overview of a Probabilistic Tracker for Multiple Cooperative Tracking Agents*, in International Conference on Advanced Robotics (ICAR), Seattle, USA, 2005.

- **R. Mottaghi** and S. Payandeh. *Coordination of Multiple Agents for Probabilistic Object Tracking*, in Canadian Conference on Computer and Robot Vision, Victoria, Canada, 2005.

- M. T. Manzuri (in alphabetical order after the first author), H. R. Chitsaz, R. Ghorbani, P. Karimian, A. R. Mirazi, M. Motamed, **R. Mottaghi**, P. Sabzmeydani. *Sharif CESR Small Size Robocup Team, A. Birk, S. Coradeschi, S. Tadokoro editors, Robocup 2001: Robot Soccer World Cup V*. Lecture notes in Artificial Intelligence 2377, p. 595, Springer-Verlag, Berlin, 2002.

CHAPTER 1

Introduction

One of the key problems in computer vision is scene understanding. We should solve several different problems to analyze an image completely. In some applications, we are interested in finding a category label for each pixel in an image. Sometimes we need to go beyond that and find the number of objects in a scene as well. In some scenarios, we should infer the type of a scene (e.g., street scene or nature scene). For some other applications, we should find the ground plane and walkable areas in a picture. Scene understanding also involves many other sub-tasks that are not mentioned here, however, most of these tasks are related to each other. For example, knowing the location of ground plane makes the object detection task easier or knowing type of a scene increases the chance of observing certain objects.

In this thesis, we focus on three highly correlated tasks: object detection, segmentation, and contextual reasoning. Because of the huge space of possible images of objects, finding an object representation enabling efficient learning and inference is essential. For this purpose, we propose compositional models for objects. We also analyze the complexity of these models (there is no such analysis in the literature) and show that *part sharing*, which is achieved by these models, provides significant speed gains in certain regimes. Subsequently, we propose a series of novel methods to improve the current object detectors and to better capture deformations of objects. Our methods not only detect the objects, but also segment out the object of interest. Our methods outperform the best previous results on difficult object detections benchmarks by a significant margin. We also show that contextual information is useful for detecting objects especially for tiny objects. The current state-of-the-art object detectors are unable to detect tiny objects due to lack of enough appearance cues. We show that contextual information provides a significant boost for detection

of tiny objects. Finally, via a series of hybrid human-machine experiments, we try to find the bottlenecks in scene understanding models. We wish to know, which of the tasks if improved, can boost performance significantly. In other words, to what degree can we expect to improve holistic understanding performance by improving the performance of individual tasks?

Below, we describe the details of the contributions of this thesis.

1.1 Contributions

Compositional Models: We propose a compositional representation for the structure of objects. These models are learnt in a generative way and enable efficient learning and inference. In this thesis, two types of compositional methods are proposed: 1) Hierarchical Compositional Models: These models are hierarchical markov random fields, where they enable part-sharing among different object categories or different views of an object category. Efficient learning and inference is possible with these models since inference/learning is performed separately for each layer of the hierarchy, and only an executive summary is propagated to the next level (Chapter 2). 2) Flat Compositional Models: Similar to the hierarchical case, we learn the parameters and structure of the graphical model simultaneously. The advantage of these models to the hierarchical case is that there is no constraint on number of children for a node in the model (Chapter 4).

In Chapter 3, we analyze the complexity of the hierarchical models in terms of computation time (for serial computers) and numbers of nodes (e.g., "neurons") for parallel computers. In particular, we compute the complexity gains by part sharing and its dependence on how the dictionary scales with the level of the hierarchy. We explore three regimes of scaling behavior and show that in some regimes the use of shared parts enables performing inference in time linear in the number of levels for an exponential number of objects.

Deformation Modeling: We propose a novel set of techniques to better model deformations of objects. Our models also provide richer descriptions for objects in terms of silhouettes and semantic parts. In Chapter 5, we propose a method to provide silhouettes for objects and also to better capture their deformations. The performance of object detectors generally degrades for highly

flexible objects. The limited topological structure of models and pre-specified part shapes are two main factors preventing these detectors from fully capturing large deformations. To better capture the deformations, we automatically find a reliably detectable part of an object and then attach other image patches to that part to cover the whole body (refer to Figure 1.1). To segment out the objects, we augment the partial detections with missing patches, and also refine detections that include background clutter.

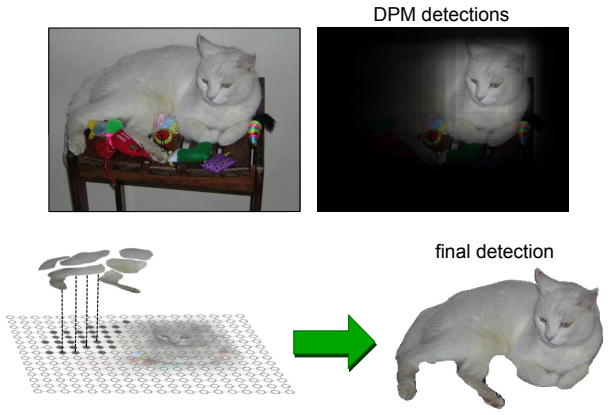


Figure 1.1: The current object detectors are often unable to capture large deformations of objects. For example, the learned models for cats by DPM [FGM10] usually correspond to the cat head, which is a rigid part. As illustrated in the top-right image, the detections are concentrated on the head (the visibility is encoded by the value obtained by accumulating the score of detections over each pixel). Our goal is to append the missing patches shown in the bottom-left panel (and remove the background patches) through a CRF framework to better estimate the object location.

In Chapter 6, we take a step further and combine segmentation with detection to model object deformations. The idea is that fixed templates are not suitable for deformable categories such as cats and dogs and we should rely on regional properties for these types of object categories. On the other hand, segments are not reliable for some categories due to over or under segmentation, and template-based detectors work better. Our detector *blends* between segmentation and detection to overcome the mistakes that each task makes. The method automatically learns whether the segments are reliable for a particular category or the fixed templates. Refer to Figure 1.2. This approach outperforms the best previous result on PASCAL VOC dataset, which is a challenging object detection dataset.

The detection task becomes difficult when we should deal partial occlusion or small scale of objects in addition to their deformation. In Chapter 7, we propose an object model to i) handle

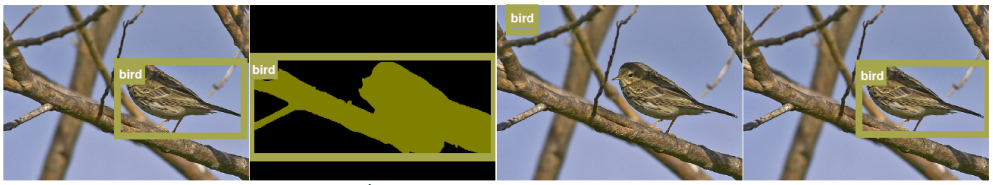


Figure 1.2: Bottom-up segmentation is used for top-down detection. The segmentation method (CPMC [CLS12]) is not accurate as it leaks to the background. The detector, which is based on HOG templates (DPM) cannot find the object. However, we find the correct bounding for the bird when we combine segmentation and detection.

partial occlusion and pose variation, ii) describe objects in terms of semantic parts, and iii) detect objects when the semantic parts are hard to detect (e.g., small objects). There are multiple components in the model corresponding to these different cases. During inference one of the components is selected (refer to Figure 1.3), and the semantic parts are localized if the component includes parts. This method improves our state-of-the-art even further.

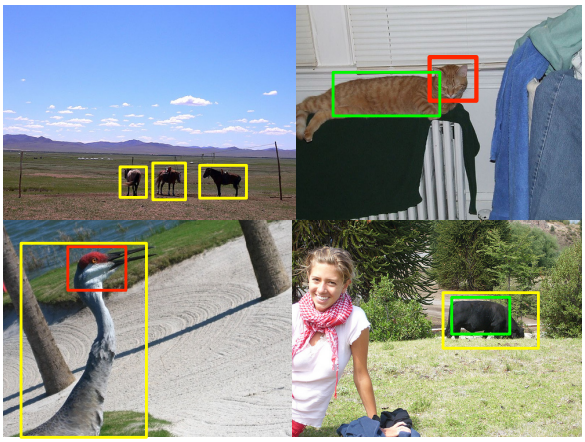


Figure 1.3: Different components are required in a model to handle different sizes, pose variations and occlusion patterns. Top left: Body-only model is typically suitable for small objects, where there are not enough appearance cues for detecting semantic parts. Top Right: Part-Part component is used to capture the cases that the object is highly deformed. In this case, the parts can be detected more reliably. Bottom Row: Body-Part components used for detecting partially occluded objects. The torso and head are missing in the left and right panels, respectively. Full body bounding box is shown in yellow. Torso is green and head is shown in red.

Contextual Reasoning: Humans use contextual information in every day tasks in order to perform

Figure 1.4: Example annotations of our novel dataset which contains pixel accurate segmentations of 623 categories on the PASCAL VOC 2010 dataset.

fast and accurate inference about the world they live in. In this thesis, we argue that the community is in need of large-scale densely labeled datasets to study the importance of inter-object and object-scene contextual interactions in tasks such as segmentation, detection and classification. Our contribution is two-fold. First, we provide a novel pixel accurate dataset which labels the PASCAL VOC challenge with 623 different contextual classes, including "things" as well as "stuff". We analyze the dataset statistically through local and global contextual co-occurrences, showing significant correlation between certain classes. We believe that this dataset will provide a rich testbed to study holistic models and will help to significantly push forward the research on semantic understanding. The second contribution is a simple extension of the deformable part-based model, which explores local and global context to better detect tiny objects. We demonstrate the performance of our detector as well as simple semantic segmentation techniques in our novel dataset.

Bottlenecks in Scene Understanding: Recent trends in image understanding have pushed for holistic scene understanding models that jointly reason about various tasks such as object detection, scene recognition, shape analysis, and contextual reasoning. Another contribution of this thesis is to show the roles of these different tasks in improved scene understanding, in particular semantic segmentation, object detection and scene recognition. Towards this goal, we "plug-in" human subjects for each of the various components in a state-of-the-art conditional random field model. Comparisons among various hybrid human-machine CRFs give us indications of how much "head room" there is to improve scene understanding by focusing research efforts on various individual

tasks.

One of the interesting findings from our slew of studies was that human classification of isolated super-pixels, while being worse than current machine classifiers for semantic segmentation, provides a significant boost in performance when plugged into the CRF. Fascinated by this finding, we conducted in depth analysis of the human generated potentials. We find that humans and machines make complimentary mistakes which the CRF can effectively exploit, resulting in improved segmentation performance. This inspired a new machine potential which significantly improves state-of-the-art *automatic* performance. We also explored a variety of shape priors. Through a series of human studies and machine experiments, we find that humans are unable to leverage the state-of-art UCM segment boundaries to reason about the shape of the object with more accuracy than a machine. We also study how well humans can leverage the contextual information provided by the CRF.

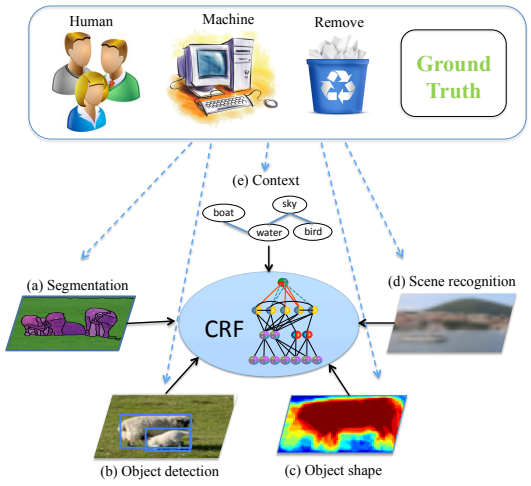


Figure 1.5: We use a holistic scene understanding approach that consists of a conditional random field (CRF) model and jointly reasons about: (a) classification of local patches (segmentation), (b) object detection, (c) shape analysis, (d) scene recognition and (e) contextual reasoning. We analyze the relative importance of each of these components by building an array of hybrid human-machine CRFs where each component is performed by a machine (default), or replaced by human subjects or ground truth, or is removed all together (top).

CHAPTER 2

Hierarchical Compositional Models for Objects

One of the key challenges in computer vision is to detect and classify object categories. The main difficulties arise because: (I) There can be considerable variation in the appearance of objects from the same category. For example, chairs can have very different shapes, colors and textures. (II) Natural images typically have complex background which makes it difficult to detect objects. (III) Finally, in many images parts of an object are truncated or occluded.

In this chapter, we present a method to learn probabilistic models of objects and apply them to classification and detection. We propose a hierarchical model constructed by recursively combining sub-parts, and hence are called recursive compositional models (RCMs). These models are learnt in a weakly supervised manner (i.e. we know that an object is present in the training image but we do not know its pose).

These models are generative so that they can be used to parse the object (i.e. locate parts of the object) and we can stochastically sample from them to validate the model (i.e. show that the samples are similar to examples of the object being modeled). In this chapter, we concentrate on the geometric modeling of the objects, where our goal is to learn the structure of the object.

In order to validate the learned models, we use them for object detection and classification tasks. The classification using RCMs is performed in two stages. Firstly, we perform inference over the RCMs for each object category to detect their optimal configurations in each image. For each model, we compute the energy of its best fit e_{min}, a geometric similarity measure s_{geom}, and the number of unmatched points in the model s_{un}. The similarity measure is the difference between the detected shape of the object and the mean shape with respect to the RCM. The number of unmatched points is the number of points in the model which are unmatched in the test image.

Secondly, we train a support vector machine (SVM) with radial basis functions (RBF) as kernels to perform classification. The input to the SVM is the output of the model inference stage, the score vector $\boldsymbol{s}= (e_{min}, s_{geom}, s_{un})$. It should be noted that for certain experiments, each RCM model can have multiple matches. Figure 2.1 shows an overview of this procedure. The contribution of this chapter is using RCMs for the classification task.

2.1 Background

One of the popular approaches in object detection is to learn a codebook of object features and use them later for recognition of new instances. Fergus et al. [FPZ03] propose an unsupervised generative model for configurations of the codebook words of objects. Leibe et al. [LLS08] learn a shape model to specify where on an object a codebook entry may appear. Zhang et al. [ZML07] investigate combining different detectors and descriptors with a classifier that can effectively use the provided information by the features. These features are sparse, so even if a generative model is learned, only a sparse set of object points can be generated.

Part-based models have proved successful on difficult object detection datasets recently. Deformable part models (DPM) by Felzenszwalb et al. [FGM10] is an example successful method. The advantage of our method to theirs is that they specify the number of object parts before learning or inference, while our method automatically learns the number of parts.

The other class of approaches to the problem of object detection is generative probabilistic modeling of objects, where the idea is to build a model that is able to generate training and testing examples. Wu et al. [WSG09] have proposed the Active Basis model, a generative model for learning deformable templates of objects. Su et al. [SSF09] describe a generative model for recognizing object classes and their 3D viewpoints. Our approach falls in this category as we learn a generative model for objects and object parts.

Hierarchical representation of spatial structure of objects has been investigated by many researchers since it resembles the structure of the biological systems. Ahuja and Todorovic [AT08] present a hierarchical approach to model objects in terms of photometric and geometric properties

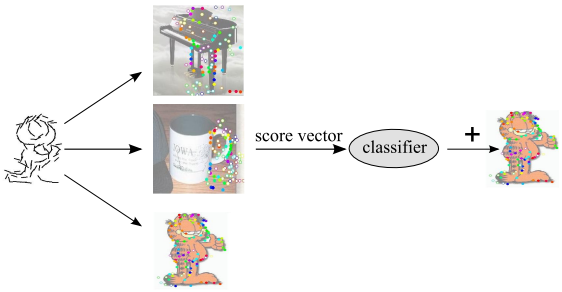


Figure 2.1: The learnt model (left) is applied to the test images to generate a score vector. The score vector is used as the input to the classifier.

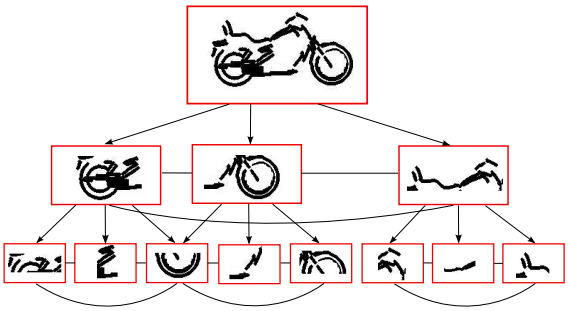


Figure 2.2: A recursive compositional model (RCM). Each node is part of a clique formed together with its siblings. The nodes are also connected to a set of children models.

and to find a relationship between the neighboring parts. Schnitzspan et al. [SFR09] describe a type of hierarchy used for finding a set of edges determining object feature interactions. Fidler el al. [FBL08] also propose a hierarchical method for part learning, where an object is composed of subparts belonging to different layers of a hierarchy.

Our work is most similar in spirit to the work of Lee et al. [LGR09] in that they propose a generative approach to learn image features. We make comparisons with their method in section (2.2.5) and show that our method results in better classification accuracy.

2.2 Recursive Compositional Models

We now describe recursive compositional models (RCMs) [ZLH08, ZCY10] which are hierarchical Markov Random Fields. They are *compositional* because the higher level parts of the object are compositions of the lower level sub-parts and they are *recursive* because the state of a part in the hierarchy is recursively computed based on the state of its constituents i.e. the states of a set of lower level sub-parts. We illustrate an RCM for a motorbike in Figure 2.2 and the mean shape of the Garfield-RCM is shown in Figure 2.1. The number of levels in the hierarchy is learnt automatically but, due to the scale of objects in the dataset, there are 4 for all the RCMs used in the experiments.

The structure of an RCM makes it straightforward to perform inference and learning. The inference task is to find the best configuration of the RCM in an image. This can be done intuitively by finding the best configurations of the low-level subparts and using them to propose configurations for the higher level parts. The RCM is defined to be robust so that high level parts can be detected even if some of the low-level subparts are missing (e.g., due to occlusion or failure of detectors). The learning task is to learn the structure of the RCM and the parameters of the model from a set of training images. The learning proceeds by modeling the low-level subparts and using these to learn models for the higher-level parts.

2.2.1 RCM representation

More formally, an RCM is defined on a hierarchical graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. There are two types of nodes in the graph: 1) The leaf nodes whose states are directly defined on the image data (image edges in this chapter). The leaf nodes are denoted by $\mathcal{V}_L$ and their state $X_\nu,\ \nu \in \mathcal{V}_L$ correspond to the position (x_ν, y_ν), orientation θ_ν, and scale s_ν of edgelets which are the smallest sub-parts of the object. 2) The second type $\mathcal{V}_I$ are the remaining nodes of the graph (i.e. $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_I$ and $\mathcal{V}_L \cap \mathcal{V}_I = \emptyset$). Their state variables represent the position (x_ν, y_ν), orientation θ_ν, and scale s_ν of parts and sub-parts of the object. The graph edges $\mathcal{E}$ show how sub-parts are combined to form parts – a parent node represents a part and its child node represents its component sub-parts. The graph structure is restricted so that each child node has a single parent (which enables the use of dynamic programming for inference). We refer to child nodes of $\nu \in \mathcal{V}_I$ by $ch(\nu)$.

An RCM is specified by a Gibbs probability distribution over the graph (which will be learnt):

$$P(X, I; \mu, \Sigma) = \frac{1}{Z(\mu, \Sigma)} e^{-E(X, I; \mu, \Sigma)}, \tag{2.1}$$

where X is the state of the graph nodes , I is the input image and μ and Σ are the parameters of the model (which will be learnt). $Z(\mu, \Sigma)$ is the partition function and E is the energy term. From now on, we refer to (μ, Σ) as θ.

There are two types of terms in the energy function $E(X, I; \theta)$. The first term E_L is defined on the leaf nodes $\mathcal{V}_L$ and specifies their interaction with the image. It requires that the leaf nodes

representing the edgelets occur at places in the image where the intensity gradient is large:

$$E_L(X, I) = \sum_{\nu \in V_L} f(X_\nu, \nabla I), \tag{2.2}$$

where f is the negative logarithm of a Gaussian defined on the image gradient ∇I (variance 1 for x, y, .12 for θ).

The second energy term E_M imposes spatial constraints on the configuration. It is defined on the cliques defined by the edges $\mathcal{E}$ in the graph and imposes soft relationships between the configurations of the parts and their sub-parts (i.e. between the states of parent nodes X_ν and child nodes $X_{ch(\nu)}$). We specify E_M by:

$$E_M(X) = \sum_{\nu \in V_I} h(X_\nu, X_{ch(\nu)}), \tag{2.3}$$

where h is a function that imposes Gaussian distribution on the relative location of the subparts and also deterministically specifies the state of the parent node according to its children. In this chapter, we consider two versions of the distribution over the child nodes. Firstly, we use the formulation in [ZLH08] which is a Gaussian distribution defined on the Invariant Triplet Vector (which is invariant to scaling and rotating the object). Secondly, we use a Gaussian distribution defined over the relative positions (x_ν, y_ν) of the child nodes. We use these different versions in sections (2.2.4,2.2.5) respectively.

2.2.2 Inference for Part and Model Detection

The inference task is to find the states X which minimize the energy $E(X, I; \theta)$ and hence maximize the probability $P(X, I; \theta)$. This can be performed efficiently by exploiting the recursive compositional structure of the RCM which enables us to exploit efficient dynamic programming techniques [ZLH08]. Although we have closed loops in the graph, dynamic programming is still possible because the number of closed loops are small [LS88]. More formally, we can express the energy for any node $\nu \in \mathcal{V}_I$ recursively by:

$$E_M(X_\nu) = h(X_\nu, X_{ch(\nu)}) + \sum_{\mu \in ch(\nu)} E_M(X_\mu), \tag{2.4}$$

which is initialized by the energies $f(X_\nu, \nabla I)$ of the leaf nodes. We use this recursive formulation to estimate the low energy states for the higher level nodes using previous estimates for the lower level states. At each level, we remove the compositions whose energy $E_M(X_\nu)$ is above a certain threshold. To ensure robustness we keep configurations for parent nodes even if one child is missing. The procedure is repeated until we find the lowest energy configurations for the root nodes.

In our experiments using object-RCMs, in section (2.2.4), we will output the graph configuration with best energy score only. But for part-RCMs, in section (2.2.5), we will output all configurations with energy scores above a threshold since the parts may appear multiple times in the image.

2.2.3 Learning

We adopt the learning algorithm described in [ZLH08]. Firstly, we use a bottom-up learning followed by top-down model refinement. This was used to learn the object-RCMs which we report in section (2.2.4). Secondly, we developed a variant which we applied on the harder task of learning of models for objects that have large variations. This was used to learn the part-RCMs used in section (2.2.5).

The basic idea of learning is sketched as follows. We input a small set of images (e.g. 10-15) which contain examples of the object and have variable background. We search for an RCM with multiple levels by starting with elementary RCMs defined over single nodes and tuned to edges at different orientations. We detect the instances of the elementary RCMs in the training images and search for frequently occurring compositions of these RCM instances (e.g., sets of horizontal and vertical edges that occur with regular spatial relationships). Higher level RCMs are constructed based on these compositions. The intuition is that frequently occurring compositions may correspond to sub-parts and parts of the objects in the image.

We repeat this procedure to form RCMs with increasing higher levels, corresponding to increasingly bigger parts of the object. This process stops automatically at the level when we fail

to discover bigger compositions which frequently occur (i.e. we have found the largest repeatable structure in the image). This defines the bottom-up pass of the algorithm which outputs a set of RCMs at different levels. A top-down pass selects the best model at the top-level and refines it by adding lower level RCMs provided they yield better fits to the images.

The challenge for this algorithm is to keep the number of RCMs small, while creating new RCMs by composition, but to make sure that we have RCMs which describe the sub-parts and parts of the object. We can allow some 'false positive' RCMs at low levels which correspond to accidental regularities in the image background because these will fail to make larger compositions (unless the background contains other regular objects). But too many false positives, which can easily occur because of the exponentially large number of compositions, can cause severe memory problems. The number of 'false RCMs' is kept small by pruning based on frequency and using non-maximal suppression to eliminate RCMs which overlap too much with others.

We also developed a second variant of the method which we used in the experiments in section (2.2.5). The major differences between this approach and the one we mentioned above are as follows: (i) The top-down stage of the learning is omitted. The intuition is that the models at the highest level look like parts rather than full object models. This makes inference more robust if parts of an object are occluded or do not appear in the image. (ii) We used a stricter pruning strategy so we enforce the learning method to learn parts of the object that appear frequently in the training images. (iii) We use a more systematic pruning approach instead of pruning the overlapping models. We compare two models by computing the Kullback-Leibler distance between the distributions of their leaf nodes. We merge two models if the KL distance is small. Our variant proved more robust when applied to the more difficult datasets according to the experiments of section (2.2.5).

2.2.4 Experimental Results: Object-RCM

For the experiments of this chapter, we use Caltech 101 is a dataset that contains images of 101 object categories and has been studied in detail for category classification so it provides a bench-

mark for us to evaluate our RCM-based classification method. We compare our results to some recent work [ZBM06, OB06, WZF06, ZML07, JKG08, BSI08, JKL09, LGR09, WYY10] on the most difficult categories of this dataset.

Our goal is to show that a single probabilistic object model that we learn is able to capture the variations present in a category. In this section, we test classification for the 10 most difficult objects in the Caltech 101 dataset, where difficulty is estimated from recent work [ZBM06, OB06, WZF06, WYY10]. We use 10 randomly selected images to learn our RCMs as described in section (2.2.3). Then we use the inference algorithm, see section (2.2.2), to these models on the rest of the images in the dataset and compute the score vector for each RCM. After this step, we follow the protocol used by others for testing on Caltech 101 dataset. We randomly select 15 positive images including those 10 that we used for learning the model and feed their score vector to a classifier. In addition, we randomly choose 1500 images of other categories as the negative training images (the dataset has 9144 images). The output of the classifier specifies if an image contains a specific category or not.

We next proceed to two quantitative tests of the performance of our models:

I. Binary-Classification using a single object model. Firstly, we show the performance of an RCM for a single model for the binary-classification task – does the image contain an example of the object or not? – when we have learnt a model of the object category only. Then we make the measurement $\mathbf{s} = (e_{min}, s_{geom}, s_{un})$ for each image. Here e_{min} is the lowest energy configuration for the object in the image, s_{un} is the number of unmatched points, s_{geom} measures how well the shape of the matched points corresponds to the prior shape of the object. Next, we perform learning by an SVM with RBF kernel with $\gamma = 10^{-5}$. The result is shown in the 3^{rd} and 4^{th} columns of Table 2.1.

II. Binary-Classification using multiple object models. Secondly, we learn models for 13 objects. The intuition is that this allows us to exploit knowledge about multiple objects. For example, the anchor RCM will have false positive responses in images of footballs because it finds 'anchor-like' shapes within them. But an RCM for a football will have a good response for a football image, yet will respond poorly on anchor images. Hence the football RCM will enable better classifica-

tion of images as 'anchor' or 'non-anchor' because it rules out footballs as 'false positives'. Our procedure is to perform the binary-classification task using an input feature vector that consists of the feature responses of all 13 object category RCMs. The feature vector for image i is defined as $\mathbf{S}^i = (\mathbf{s}_1^i, ..., \mathbf{s}_{13}^i)$. For training, we again use SVM with RBF kernel with the same γ.

The result is shown in the first and second columns of Table 2.1. The table shows that exploiting the knowledge about the models of other objects improve the classification accuracy significantly (one reason why we use three additional learnt models – learn 13 RCMs but classify 10 object categories – is that these three extra object categories contributed a large number of false positives to the single model method). The true positive rates that [ZBM06] obtain are 69, 42, 42, 48, 50, 46, 42, 46, 57 and 56 and [WYY10] obtain 22, 14, 54, 57, 89, 44, 5, 62, 63 and 58 in the order mentioned in Table 2.1 but we can not compare our method directly with theirs as they use a multi-class classifier. In addition, we can not directly estimate the number of false positives from the color plots of the confusion matrix that they provide (in theory these can be calculated, but it requires estimating numbers from colors of a large matrix).

We show parses for different categories in Figure 2.3. Different parts of an object are shown by color dots. The dots that have the same color correspond to similar parts of an object in different images. The object-RCMs enable us to describe and locate the object, as well as classify it. In addition, we can obtain good classification even when the parsing results are poor.

2.2.5 Experimental Results: Part-RCMs

The results reported in the previous section were good, but we have two concerns: (i) our strategy used a single RCM model for each object which seems unsuited to object categories where there is considerable variation in the configuration of the object, (Figure 2.4), and (ii) Most of Caltech 101 images have comparatively simple background, which makes learning the RCMs practical, and we need to test our approach on harder datasets.

This motivates the second set of experiments where we represent each object by a set of RCMs which describe large parts of the objects. These RCMs are learnt using the variant of unsupervised

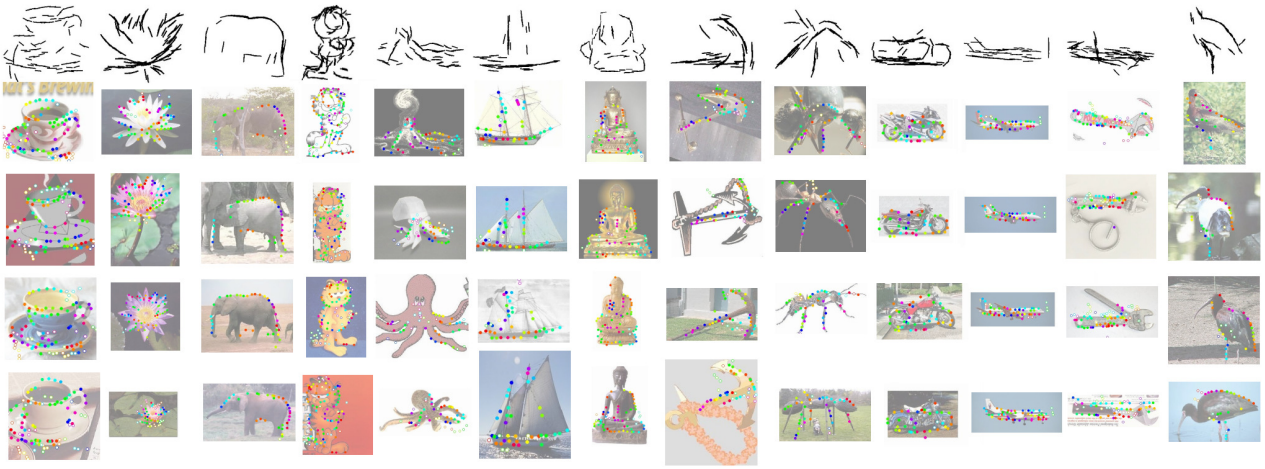


Figure 2.3: First row represents the mean of the probabilistic models learned for 13 object categories of Caltech 101 dataset. Parsing (identifying sub-parts of the object) results are shown on four example images. Corresponding parts in different images are shown by the same color.

Figure 2.4: Cougars and chairs are two categories that exhibit deformation and high intra-class variation. Two part-RCMs are shown for each category.

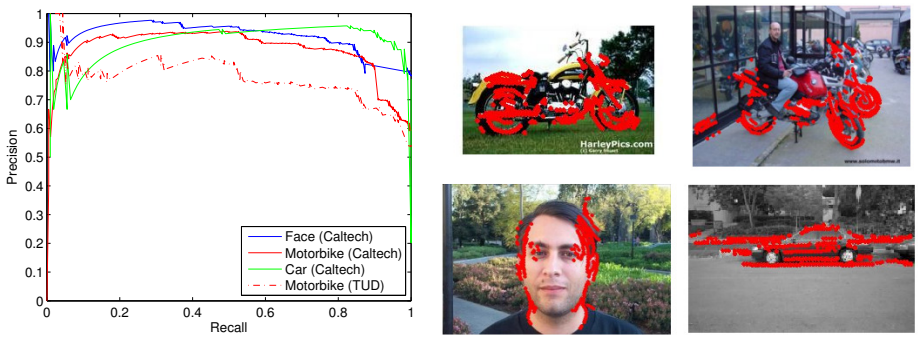
	TP (%) multiple	AP multiple	TP (%) single	AP single
Anchor	40.7±3.7	0.033±0.005	41.4±8.8	0.011±0.003
Ant	45.9±5.6	0.027±0.002	64.4±9.3	0.009±0.000
Cup	64.2±7.3	0.095±0.012	75.2±5.2	0.027±0.002
Elephant	55.0±2.8	0.081±0.012	51.6±3.7	0.021±0.002
Garfield	80.0±7.8	0.345±0.082	81.0±7.0	0.172±0.082
Ibis	68.9±9.0	0.234±0.051	57.5±12.6	0.025±0.004
Octopus	55.0±13.2	0.019±0.002	39.1±5.8	0.004±0.000
Schooner	73.7±6.3	0.348±0.023	86.2±6.6	0.075±0.015
Water Lilly	72.7±12.4	0.096±0.023	63.6±6.4	0.021±0.003
Wrench	59.1±6.8	0.194±0.081	64.1±11.6	0.009 ± 0.001

Table 2.1: Performance using single and multiple RCM models. The percentage of true positives and average precision have been shown.

learning described in section (2.2.3). We run the inference algorithm to detect all these part-RCMs in the input images. In each image, we may detect several or no instances of each part-RCM. This can help deal with different configurations of the object.

To classify images based on the learned part-RCMs, we use an SVM classifier with RBF kernel. The input is the normalized histogram of the part-RCMs – i.e. the normalized counts of the part-RCMs detected in the image (using the inference algorithm). This is not the optimal way to classify based on the part-RCM responses, because it ignores the spatial locations of the part-RCMs, but it gives good results as we report below. We expect that we will obtain better results if we use spatial relations between the positions of the part-RCMs.

First, we report our results on the same categories as the ones used by [LGR09], namely, Face, Motorbike, and Car. Their method uses unsupervised learning, in their case a convolutional net, to learn a vocabulary of image features so it is a reasonable baseline for us. We use 10 randomly



(a) (b)

Figure 2.5: (a) Precision-Recall curves of the classification task using Part-RCMs. The curves are obtained by changing the threshold on the confidence of the classifier. (b) The red lines are the parts detected on some example images.

selected images to learn part-RCMs (except for Motorbikes, see below). To train the classifier, we select 30 random images half of which are positive training examples and the other half are negative ones. Our learning method outputs 9, 22 and 11 part-RCMs for Faces, Motorbikes and Cars, respectively. The average precision of our method is shown in Table 2.2 and the precision-recall curves are shown in Figure 2.5(a). There are two face categories in Caltech 101. We run the experiments on the more difficult category.

To evaluate the discrimination power of our part-RCMs in the presence of background clutter, occlusion and scale changes, we used the TUD Motorbike dataset, which is part of PASCAL 2005 collection [LLS08], see Figure 2.5(b). This dataset contains 115 images of motorbikes and we used 15 of them for learning part-RCMs and for training the classifier. We also used 115 images randomly selected from the PASCAL VOC 2007 as the negative examples, 15 of which used for training the classifier. Bicycle images resulted in the most confusion in this experiment. The Precision-Recall curve is shown in Figure 2.5(a) and our average precision is 80%.

To show that our model is able to transfer to other datasets, both of the motorbikes experiments are performed using models that are learnt on TUD motorbikes i.e. we learned models on one dataset and tested them on both of the motorbike datasets.

	Faces	Motorbikes	Cars
Our method	0.93	0.89	0.94
[LGR09]	0.95	0.81	0.87

Table 2.2: The average precision for classification of three categories has been shown. The standard deviation of our approach is less than 0.03 in all of the cases.

To show the effectiveness of the part-RCMs on deformable objects and objects with high intra-class variations, we repeat the experiments on cougar and chair categories of Caltech 101. We learn 9 and 10 parts for cougars and chairs, two of which are shown in Figure 2.4. Our classification accuracy for cougar category is $43.7 \pm 7.9\%$ with 48.2 ± 16.3 false positives (this is the number not percentage). We use 505 negative examples, where we select 5 images uniformly from each category. Our true positive percentage for chairs is 72.3 ± 4.6, where the number of false positives is 72.6 ± 4.3. The true positive rate that [WYY10] and [ZBM06] have reported are 37% and 75% for chairs, and 36% and 48% for cougars. It should be noted that we cannot directly compare our results with theirs since they use a multi-class classifier.

We should emphasize that the normalized histogram of part-RCMs responses is the only input to the classifier and no other cues such as spatial relation of the features is used. Some example inference results for these datasets are shown in Figure 2.5(b).

2.3 Conclusion

We proposed a hierarchical generative model for objects. We applied these models to images to detect configurations of objects and measured their energy and geometric properties. We used these as inputs to train a classifier, SVM with radial basis function kernel, and obtained reasonable results on benchmarked datasets. The experiments also show that incorporating the information from the models of other categories improves the classification performance.

There are limitations to the RCM models of this chapter: our models only use simple cues

like intensity gradients so they are sensitive to local deformations of objects. In addition, missing edges of objects results in learning incomplete models. Another limitation is the fixed number of children (we assumed a part was composed of 3 sub-parts). Also, we used heuristics to prune out some of the models or compositions. Finding a more principled way of pruning is part of future work.

In the next section, we provide complexity analysis for these models and show that in certain regimes we can gain significant speed ups using these models. The theoretical analysis and our initial experiments suggest that we should parallelize these models, which will make large scale training and testing practical. The parallelization can take place both in the algorithm and the hardware. GPU-type architectures are one of the options for hardware speedup.

The proposed methods only deal with the structure of the objects and ignore any appearance information. A possible extension is to integrate appearance cues in the generative model. We believe that appearance information will significantly decrease the number of false responses.

CHAPTER 3

Complexity of Representation and Inference in Compositional Models

A fundamental problem of vision is how to deal with the enormous complexity of images and visual scenes [1]. The total number of possible images is almost infinitely large [Ker87]. The number of objects is also huge and has been estimated at around 30,000 [Bie87]. How can a biological, or artificial, vision system deal with this complexity? For example, considering the enormous input space of images and output space of objects, how can humans interpret images in less than 150 msec [TFM96]?

There are three main issues involved. Firstly, how can a visual system be designed so that it can efficiently *represent* large classes of objects, including their parts and subparts? Secondly, how can the visual system be designed so that it can rapidly *infer* which object, or objects, are present in an input image and the positions of their subparts? And, thirdly, how can this representation be *learnt* in an unsupervised, or weakly supervised fashion? In short, what visual *architectures* enable us to address these three issues?

Many considerations suggest that visual architectures should be hierarchical. The structure of mammalian visual systems is hierarchical with the lower levels (e.g., in areas V1 and V2) tuned to small image features while the higher levels (i.e. in area IT) are tuned to objects [2]. Moreover, as appreciated by pioneers such as Fukushima [Fuk88], hierarchical architectures lend themselves naturally to efficient representations of objects in terms of parts and subparts which

[1] Similar complexity issues will arise for other perceptual and cognitive modalities.

[2] But just because mammalian visual systems are hierarchical does not necessarily imply that this is the best design for computer vision systems.

can be shared between many objects. Hierarchical architectures also lead to efficient learning algorithms as illustrated by deep belief learning and others [HOT06]. There are many varieties of hierarchical models which differ in details of their representations and their learning and inference algorithms [RP99, HOT06, SWB07, AW03, PD10, ZKT10, LB95, BU02, KY11]. But, to the best of our knowledge, there has been no detailed study of their complexity properties.

This chapter provides a mathematical analysis of compositional models [GPC02], which are a subclass of the hierarchical models. The key idea of compositionality is to explicitly represent objects by recursive composition from parts and subparts. This gives rise to natural learning and inference algorithms which proceed from sub-parts to parts to objects (e.g., inference is efficient because a leg detector can be used for detecting the legs of cows, horses, and yaks). The explicitness of the object representations helps quantify the efficiency of part-sharing and make mathematical analysis possible. The compositional models we study are based on the work of Zhu et al. [ZCT10, ZLH08] that we partly described in the previous chapter. However, we make several technical modifications including a parallel re-formulation of the models. We note that in previous papers [ZCT10, ZLH08] the representations of the compositional models were learnt in an unsupervised manner, which relates to the memorization algorithms of Valiant [Val00]. This chapter explores the consequence of the representations which are learnt.

Our analysis assumes that objects are represented by hierarchical graphical probability models [3] which are composed from more elementary models by *part-subpart compositions*. An object – a graphical model with $\mathcal{H}$ levels – is defined as a composition of r parts which are graphical models with $\mathcal{H}-1$ levels. These parts are defined recursively in terms of subparts which are represented by graphical models of increasingly lower levels. It is convenient to specify these compositional models in terms of a set of dictionaries $\{\mathcal{M}_h : h = 1, .., , \mathcal{H}\}$ where the level-h parts in dictionary $\mathcal{M}_h$ are composed in terms of level-$h-1$ parts in dictionary $\mathcal{M}_{h-1}$. The highest level dictionaries $\mathcal{M}_{\mathcal{H}}$ represent the set of all objects. The lowest level dictionaries $\mathcal{M}_1$ represent the elementary features that can be measured from the input image. Part-subpart composition enables us to construct a very large number of objects by different compositions of elements from the

[3] These graphical models contain closed loops but with restricted maximal clique size.

lowest-level dictionary. It enables us to perform *part-sharing* during learning and inference, which can lead to enormous reductions in complexity, as our mathematical analysis will show.

There are three factors which enable computational efficiency. The first is *part-sharing*, as described above, which means that we only need to perform inference on the dictionary elements. The second is the *executive-summary principle*. This principle allows us to represent the state of a part coarsely because we are also representing the state of its subparts (e.g., an executive will only want to know that "there is a horse in the field" and will not care about the precise positions of its legs). For example, consider a letter T which is composed of a horizontal and vertical bar. If the positions of these two bars are specified precisely, then we can specify the position of the letter T more crudely (sufficient for it to be "bound" to the two bars). This relates to Lee and Mumford's high-resolution buffer hypothesis [LM03] and possibly to the experimental finding that neurons higher up the visual pathway are tuned to increasingly complex image features but are decreasingly sensitive to spatial position. The third factor is *parallelism* which arises because the part dictionaries can be implemented in parallel, essentially having a set of receptive fields, for each dictionary element. This enables extremely rapid inference at the cost of a larger, but parallel, graphical model.

The compositional section (3.1) introduces the key ideas. Section 4.2.2 describes the inference algorithms for serial and parallel implementations. Section 3.3 performs a complexity analysis and shows potential exponential gains by using compositional models.

3.1 The Compositional Models

Compositional models are based on the idea that objects are built by compositions of parts which, in turn, are compositions of more elementary parts. These are built by part-subpart compositions.

3.1.1 Compositional Part-Subparts

We formulate part-subpart compositions by probabilistic graphical model which specifies how a part is composed of its subparts. A parent node ν of the graph represents the part by its type τ_ν and a state variable x_ν (e.g., x_ν could indicate the position of the part). The r child nodes $Ch(\nu) = (\nu_1, ..., \nu_r)$ represent the parts by their types $\tau_{\nu_1}, ..., \tau_{\nu_r}$ and state variables $\vec{x}_{Ch(\nu)} = (x_{\nu_1}, ..., x_{\nu_r})$ [4]. The type of the parent node is specified by $\tau_\nu = (\tau_{\nu_1}, ..., \tau_{\nu_r}, \lambda_\nu)$. Here $(\tau_{\nu_1}, ..., \tau_{\nu_r})$ are the types of the child nodes, and λ_ν specifies a distribution over the states of the subparts (e.g.,over their relative spatial positions). Hence the type τ_ν of the parent specifies the part-subpart compositional model.

The probability distribution for the part-subpart model relates the states of the part and the subparts by:

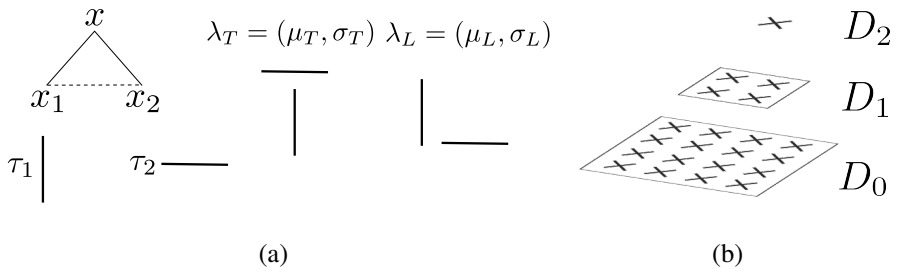
$$P(\vec{x}_{Ch(\nu)}|x_\nu; \tau_\nu) = \delta(x_\nu - f(\vec{x}_{Ch(\nu)}))h(\vec{x}_{Ch(\nu)}; \lambda_\nu). \tag{3.1}$$

Here $f(.)$ is a deterministic function, so the state of the parent node is determined uniquely by the state of the child nodes. The function $h(.)$ specifies a distribution on the relative states of the child nodes. The distribution $P(\vec{x}_{Ch(\nu)}|x_\nu; \tau_\nu)$ obeys a *locality principle*, which means that $P(\vec{x}_{Ch(\nu)}|x_\nu; \tau_\nu) = 0$, unless $|x_{\nu_i} - x_\nu|$ is smaller than a threshold for all $i = 1, .., r$. This requirement captures the intuition that subparts of a part are typically close together.

The state variable x_ν of the parent node provides an *executive summary* description of the part. Hence they are restricted to take a smaller set of values than the state variables x_{ν_i} of the subparts. Intuitively, the state x_ν of the parent offers summary information (e.g., there is a cow in the right side of a field) while the child states $\vec{x}_{Ch(\nu)}$ offer more detailed information (e.g., the position of the parts of the cow). In general, information about the object is represented in a distributed manner with coarse information at the upper levels of the hierarchy and more precise information at lower levels.

We give examples of part-subpart compositions in Figure 3.1(a). The compositions represent the letters T and L, which are the types of the parent nodes. The types of the child nodes are

[4] We assume a fixed value r for all part-subpart compositions.



(a) (b)

Figure 3.1: (a) Compositional part-subpart models for T and L are constructed from the same elementary components τ_1, τ_2, horizontal and vertical bar using different spatial relations $\lambda = (\mu, \sigma)$, which impose *locality*. The state x of the parent node gives the summary position of the object, the *executive summary*, while the positions x_1, x_2 of the components give details about its components. (b) The hierarchical lattices. The size of the lattices decrease with scale by a factor q which helps enforce executive summary and prevent having multiple hypotheses which overlap too much. $q = 1/4$ in this figure.

horizontal and vertical bars, indicated by $\tau_1 = H,\ \tau_2 = V$. The child state variables x_1, x_2 indicate the image positions of the horizontal and vertical bars. The state variable x of the parent node gives a summary description of the position of the letters T and L. The compositional models for letters T and L differ by their λ parameter which species the relative positions of the horizontal and vertical bars. In this example, we choose $h(.;\lambda)$ to a Gaussian distribution, so $\lambda = (\mu, \sigma)$ where μ is the mean relative positions between the bars and σ is the covariance. We set $f(x_1, x_2) = (1/2)(x_1 + x_2)$, so the state of the parent node specifies the average positions of the child nodes (i.e. the positions of the two bars). Hence the two compositional models for the T and L have types $\tau_T = (H, V, \lambda_T)$ and $\tau_L = (H, V, \lambda_L)$.

3.1.2 Models of Object Categories

An object category can be modeled by repeated part-subpart compositions. This is illustrated in Figure 3.2 where we combine T's and L's with other parts to form more complex objects. More generally, we can combine part-subpart compositions into bigger structures by treating the parts as subparts of higher order parts.

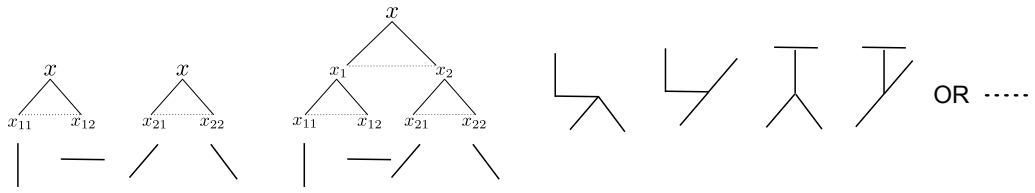


Figure 3.2: Left Panel: Two part-subpart models. Center Panel: Combining two part-subpart models by composition to make a higher level model. Right Panel: Some examples of the shapes that can be generated by different parameters settings λ of the distribution.

More formally, an object category of type $\tau_{\mathcal{H}}$ is represented by a probability distribution defined over a graph $\mathcal{V}$. This graph has a hierarchical structure with levels $h \in \{0, ..., \mathcal{H}\}$, where $\mathcal{V} = \bigcup_{h=0}^{\mathcal{H}} \mathcal{V}_h$. Each object has a single, root node, at level-$\mathcal{H}$ (i.e. $\mathcal{V}_{\mathcal{H}}$ contains a single node). Any node $\nu \in \mathcal{V}_h$ (for $h > 0$) has r children nodes $Ch(\nu)$ in $\mathcal{V}_{h-1}$ indexed by $(\nu_1, ..., \nu_r)$. Hence there are $r^{\mathcal{H}-h}$ nodes at level-h (i.e. $|\mathcal{V}_h| = r^{\mathcal{H}-h}$).

At each node ν there is a state variable x_ν which indicates spatial position and type τ_ν. The type $\tau_{\mathcal{H}}$ of the root node indicates the object category and also specifies the types of its parts.

The position variables x_ν take values in a *set of lattices* $\{\mathcal{D}_h : h = 0, ..., \mathcal{H}\}$, so that a level-h node, $\nu \in \mathcal{V}_h$, takes position $x_\nu \in \mathcal{D}_h$. The leaf nodes $\mathcal{V}_0$ of the graph take values on the image lattice $\mathcal{D}_0$. The lattices are evenly spaced and the number of lattice points decreases by a factor of $q < 1$ for each level, so $|\mathcal{D}_h| = q^h |\mathcal{D}_0|$, see Figure 3.1(b). This decrease in number of lattice points imposes the *executive summary principle*. The lattice spacing is designed so that parts do not overlap. At higher levels of the hierarchy the parts cover larger regions of the image and so the lattice spacing must be larger, and hence the number of lattice points smaller, to prevent overlapping [5].

The probability model for an object category of type $\tau_{\mathcal{H}}$ is specified by products of part-subpart relations:

$$P(\vec{x}|\tau_{\mathcal{H}}) = \prod_{\nu \in \mathcal{V}/\mathcal{V}_0} P(\vec{x}_{Ch(\nu)}|x_\nu; \tau_\nu) U(x_{\mathcal{H}}). \tag{3.2}$$

[5] Previous work [ZLH08, ZCT10] was not formulated on lattices and used non-maximal suppression to achieve the same effect.

Here $U(x_{\mathcal{H}})$ is the uniform distribution.

3.1.3 Multiple Object Categories, Shared Parts, and Hierarchical Dictionaries

Now suppose we have a set of object categories $\tau_{\mathcal{H}} \in \mathcal{H}$, each of which can be expressed by an equation such as Equation 3.2. We assume that these objects share parts. To quantify the amount of part sharing we define a hierarchical dictionary $\{\mathcal{M}_h : h = 0, ..., \mathcal{H}\}$, where $\mathcal{M}_h$ is the dictionary of parts at level h. This gives an exhaustive set of the parts of this set of the objects, at all levels $h = 0, ..., \mathcal{H}$. The elements of the dictionary $\mathcal{M}_h$ are composed from elements of the dictionary $\mathcal{M}_{h-1}$ by part-subpart compositions [6].

This gives an alternative way to think of object models. The type variable τ_ν of a node at level h (i.e. in $\mathcal{V}_h$) indexes an element of the dictionary $\mathcal{M}_h$. Hence objects can be encoded in terms of the hierarchical dictionary. Moreover, we can create new objects by making new compositions from existing elements of the dictionaries.

3.1.4 The Likelihood Function and the Generative Model

To specify a generative model for each object category we proceed as follows. The prior specifies a distribution over the positions and types of the leaf nodes of the object model. Then the likelihood function is specified in terms of the type at the leaf nodes (e.g., if the leaf node is a vertical bar, then there is a high probability that the image has a vertical edge at that position).

More formally, the prior $P(\vec{x}|\tau_{\mathcal{H}})$, see Equation 3.2, specifies a distribution over a set of points $\mathcal{L} = \{x_\nu : \nu \in \mathcal{V}_0\}$ (the leaf nodes of the graph) and specifies their types $\{\tau_\nu : \nu \in \mathcal{V}_0\}$. These points are required to lie on the image lattice (e.g., $x_\nu \in \mathcal{D}_0$). We denote this as $\{(x, \tau(x)) : x \in \mathcal{L}\}$ where $\tau(x)$ is specified in the natural manner (i.e. if $x = x_\nu$ then $\tau(x) = \tau_\nu$). We specify distributions $P(I(x)|\tau(x))$ for the probability of the image $I(x)$ at x conditioned on the type of the leaf node. We specify a default probability $P(I(x)|\tau_0)$ at positions x where there is no leaf node of the object.

[6] The unsupervised learning algorithm in [ZCT10] automatically generates this hierarchical dictionary.

This gives a likelihood function for the states $\vec{x} = \{x_\nu \in \mathcal{V}\}$ of the object model in terms of the image $\mathbf{I} = \{I(x) : x \in \mathcal{D}_0\}$:

$$P(\mathbf{I}|\vec{x}) = \prod_{x \in \mathcal{L}} P(I(x)|\tau(x)) \times \prod_{x \in \mathcal{D}_0/\mathcal{L}} P(I(x)|\tau_0). \tag{3.3}$$

The likelihood and the prior, Equations 3.3 and 3.2, give a generative model for each object category.

We can extend this in the natural manner to give generative models or two, or more, objects in the image provided they do not overlap. Intuitively, this involves multiple sampling from the prior to determine the types of the lattice pixels, followed by sampling from $P(I(x)|\tau)$ at the leaf nodes to determine the image $\mathbf{I}$. Similarly, we have a default *background model* for the entire image if no object is present:

$$P_B(\mathbf{I}) = \prod_{x \in \mathcal{D}_0} P(I(x)|\tau_0). \tag{3.4}$$

3.2 Inference by Dynamic Programming

The inference task is to determine which objects are present in the image and to specify their positions. This involves two subtasks: (i) state estimation, to determine the optimal states of a model and hence the position of the objects and its parts, and (ii) model selection, to determine whether objects are present or not. As we will show, both tasks can be reduced to calculating and comparing log-likelihood ratios which can be performed efficiently using dynamic programming methods.

We will first describe the simplest case which consists of estimating the state variables of a single object model and using model selection to determine whether the object is present in the image and, if so, how many times. Next we show that we can perform inference and model selection for multiple objects efficiently by exploiting part sharing (using hierarchical dictionaries). Finally, we show how these inference tasks can be performed even more efficiently using a parallel implementation. We stress that we are performing *exact inference* and no approximations are made. We are simply exploiting part-sharing so that computations required for performing inference for

one object can be re-used when performing inference for other objects.

3.2.1 Inference Tasks: State Detection and Model Selection

We first describe a standard dynamic programming algorithm for finding the optimal state of a single object category model. Then we describe how the same computations can be used to perform model selection and to the detection and state estimation if the object appears multiple times in the image (non-overlapping).

Consider performing inference for a single object category model defined by Equations 3.2 and 3.3. To calculate the MAP estimate of the state variables requires computing $\vec{x}^* = \arg\max_{\vec{x}}\{\log P(\mathbf{I}|\vec{x}) + \log P(\vec{x};\tau_{\mathcal{H}})\}$. By subtracting the constant term $\log P_B(\mathbf{I})$ from the right hand side, we can re-express this as estimating:

$$\vec{x}^* = \arg\max_{\vec{x}}\{\sum_{x\in\mathcal{L}} \log \frac{P(I(x)|\tau(x))}{P(I(x)|\tau_0)} + \sum_{\nu} \log P(\vec{x}_{Ch(\nu)}|x_\nu;\tau_\nu) + \log U(x_{\mathcal{H}})\}. \tag{3.5}$$

Here $\mathcal{L}$ denotes the positions of the leaf nodes of the graph, which must be determined during inference.

We estimate $\vec{x}^*$ by performing dynamic programming. This involves a bottom-up pass which recursively computes quantities $\phi(x_h,\tau_h) = \arg\max_{\vec{x}/x_h}\{\log \frac{P(\mathbf{I}|\vec{x})}{P_B(\mathbf{I})} + \log P(\vec{x};\tau_h)\}$ by the formula:

$$\phi(x_h,\tau_h) = \max_{\vec{x}_{Ch(\nu)}}\{\sum_{i=1}^{r} \phi(x_{\nu_i},\tau_{\nu_i}) + \log P(\vec{x}_{Ch(\nu)}|x^*_\nu,\tau_\nu)\}. \tag{3.6}$$

We refer to $\phi(x_h,\tau_h)$ as the *local evidence* for part τ_h with state x_h (after maximizing over the states of the lower parts of the graphical model). This local evidence is computed bottom-up. We call this the local evidence because it ignores the context evidence for the part which will be provided during top-down processing (i.e. that evidence for other parts of the object, in consistent positions, will strengthen the evidence for this part).

The bottom-up pass outputs the *global evidence* $\phi(x_{\mathcal{H}},\tau_{\mathcal{H}})$ for object category $\tau_{\mathcal{H}}$ at position $x_{\mathcal{H}}$. We can detect the most probable state of the object by computing $x^*_{\mathcal{H}} = \arg\max \phi(x_{\mathcal{H}},\tau_{\mathcal{H}})$.

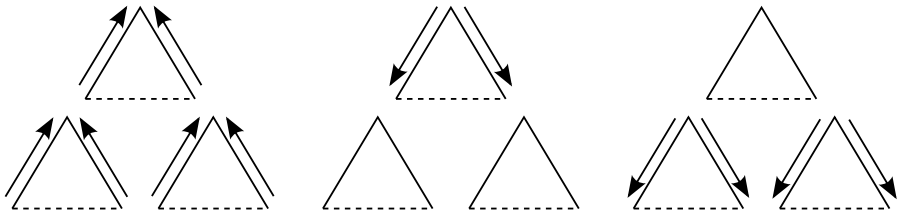


Figure 3.3: Left Panel: The feedforward pass propagates hypotheses up to the highest level where the best state is selected. Center Panel: Feedback propagates information from the top node disambiguating the middle level nodes. Right Panel: Feedback from the middle level nodes propagates back to the input layer to resolve ambiguities there. This algorithm rapidly estimates the top-level executive summary description in a rapid feed-forward pass. The top-down pass is required to allow high-level context to eliminate false hypotheses at the lower levels– "high-level tells low-level to stop gossiping".

Then we can perform the top-down pass of dynamic programming to estimate the most probable states $\vec{x}^*$ of the entire model by recursively performing:

$$\vec{x}^*_{Ch(\nu)} = \arg\max_{\vec{x}_{Ch(\nu)}} \{\sum_{i=1}^{r} \phi(x_{\nu_i}, \tau_{\nu_i}) + \log P(\vec{x}_{Ch(\nu)} | x^*_\nu, \tau_\nu)\}. \tag{3.7}$$

This outputs the most probable state of the object in the image. Note that the bottom-up process first estimates the optimal "executive summary" description of the object ($x^*_{\mathcal{H}}$) and only later determines the optimal estimates of the lower-level states of the object in the top-down pass. Hence, the algorithm is faster at detecting that there is a cow in the right side of the field (estimated in the bottom-up pass) and is slower at determining the position of the feet of the cow (estimated in the top-down pass). This is illustrated in Figure 3.3.

Importantly, we only need to perform slight extensions of this algorithm to compute significantly more. First, we can perform model selection – to determine if the object is present in the image – by determining if $\phi(x^*_{\mathcal{H}}, \tau_{\mathcal{H}}) > T$, where T is a threshold. This is because, by Equation 3.5, $\phi(x^*_{\mathcal{H}}, \tau_{\mathcal{H}})$ is the log-likelihood ratio of the probability that the object is present at position $x^*_{\mathcal{H}}$ compared to the probability that the corresponding part of the image is generated by the background image model $P_B(.)$. Secondly, we can compute the probability that the object occurs several times in the image, by computing the set $\{x_{\mathcal{H}} : \phi(x^*_{\mathcal{H}}, \tau_{\mathcal{H}}) > T\}$, to compute the "exec-

utive summary" descriptions for each object (e.g., the coarse positions of each object). We then perform the top-down pass initialized at each coarse position (i.e. at each point of the set described above) to determine the optimal configuration for the states of the objects. Hence, we can reuse the computations required to detect a single object in order to detect multiple instances of the object (provided there are no overlaps)[7]. The number of objects in the image is determined by the log-likelihood ratio test with respect to the background model.

3.2.2 Inference on Multiple Objects by Part Sharing using the Hierarchical Dictionaries

Now suppose we want to detect instances of many object categories $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$ simultaneously. We can exploit the shared parts by performing inference using the hierarchical dictionaries.

The main idea is that we need to compute the global evidence $\phi(x_{\mathcal{H}}, \tau_{\mathcal{H}})$ for all objects $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$ and at all positions $x_{\mathcal{H}}$ in the top-level lattice. These quantities could be computed separately for each object by performing the bottom-up pass, specified by Equation 3.6, for each object. But this is wasteful because the objects share parts and so we would be performing the same computations multiple times. Instead we can perform all the necessary computations more efficiently by working directly with the hierarchical dictionaries.

More formally, computing the global evidence for all object models and at all positions is specified as follows.

$$\begin{aligned} \text{Let } \mathcal{D}^*_{\mathcal{H}} &= \{x_{\mathcal{H}} \in \mathcal{D}_{\mathcal{H}} \text{ s.t. } \max_{\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}} \phi(x_{\mathcal{H}}, \tau_{\mathcal{H}}) > T_{\mathcal{H}}\}, \\ \text{For } x_{\mathcal{H}} \in \mathcal{D}^*_{\mathcal{H}}, \text{ let } \tau^*_{\mathcal{H}}(x_{\mathcal{H}}) &= \arg \max_{\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}} \phi(x_{\mathcal{H}}, \tau_{\mathcal{H}}), \\ \text{Detect } \vec{x}^*/x_{\mathcal{H}} = \arg \max_{\vec{x}/x_{\mathcal{H}}} \{\log \frac{P(\mathbf{I}|\vec{x})}{P_B(\mathbf{I})} &+ \log P(\vec{x}; \tau_{\mathcal{H}^*(x_{\mathcal{H}})})\} \text{ for all } x_{\mathcal{H}} \in \mathcal{D}^*_{\mathcal{H}}. \end{aligned} \tag{3.8}$$

All these calculations can be done efficiently using the hierarchical dictionaries (except for the $\max$ and $\arg\max$ tasks at level $\mathcal{H}$ which must be done separately). Recalling that each dictionary

[7]Note that this is equivalent to performing optimal inference simultaneously over a set of different generative models of the image, where one model assumes that there is one instance of the object in the image, another models assumes there are two, and so on.

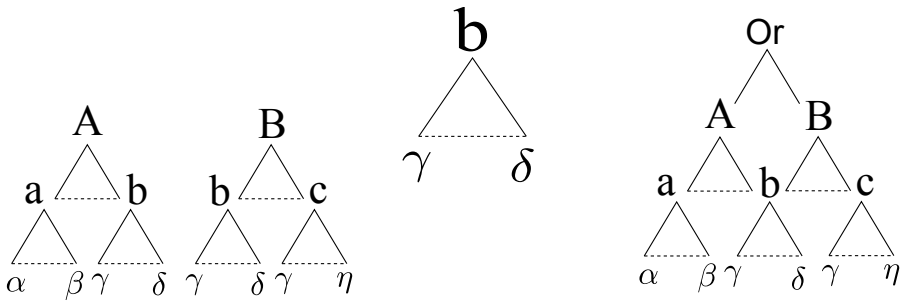


Figure 3.4: Sharing. Left Panel: Two Level-2 models A and B which share Level-1 model b as a subpart. Center Panel: Level-1 model b. Inference computation only requires us to do inference over model b once, and then it can be used to computer the optimal states for models A and B. Right Panel: Note that is we combine models A and B by a root OR node then we obtain a graphical model for both objects. This model has a closed loop which would seem to make inference more challenging. But by exploiting the shape part we can do inference optimally despite the closed loop. Inference can be done on the dictionaries, far right.

element at level h is composed, by part-subpart composition, of dictionary elements at level $h-1$. Hence we can apply the bottom-up update rule in Equation 3.6 directly to the dictionary elements. This is illustrated in Figure 3.4. As analyzed in the next section, this can yield major gains in computational complexity.

Once the global evidences for each object model have been computed at each position (in the top lattice) we can perform winner-take-all to estimate the object model which has largest evidence at each position. Then we can apply thresholding to see if it passes the log-likelihood ratio test compared to the background model. If it does pass this log-likelihood test, then we can use the top-down pass of dynamic programming, see Equation 3.7, to estimate the most probable state of all parts of the object model.

We note that we are performing exact inference over multiple object models at the same time. This is perhaps un-intuitive to some readers because this corresponds to doing exact inference over a probability model which can be expressed as a graph with a large number of closed loops, see Figure 3.4. But the main point is that part-sharing enables us share inference efficiently between many models.

The only computation which cannot be performed by dynamic programming are the $\max$ and $\arg\max$ tasks at level $\mathcal{H}$, see top line of Equation 3.8. These are simple operations and require order $M_{\mathcal{H}} \times |\mathcal{D}_{\mathcal{H}}|$ calculations. This will usually be a small number, compared to the complexity of other computations. But this will become very large if there are a large number of objects, as we will discuss in Section 3.3.

3.2.3 Parallel Formulation and Inference Algorithm

Finally, we observe that all the computations required for performing inference on multiple objects can be parallelized. This requires computing the quantities in Equation 3.8.

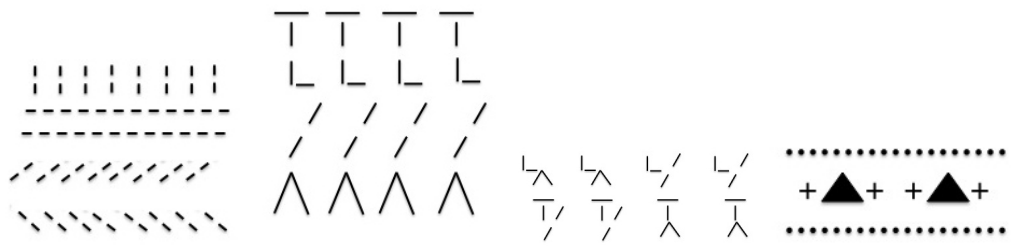


Figure 3.5: Parallel Hierarchical Implementation. Far Left Panel: Four Level-0 models are spaced densely in the image (here an 8×2 grid). Left Panel: the four Level-1 models are sampled at a lower rate, and each have 4×1 copies. Right Panel: the four Level-2 models are sampled less frequently. Far Right Panel: a bird's eye view of the parallel hierarchy. The dots represent a "column" of four Level-0 models. The crosses represent columns containing four Level-1 models. The triangles represent a column of the Level-2 models.

The parallelization is possible, in the bottom-up pass of dynamic programming, calculations are done separately for each position x, see Equation 3.6. So we can compute the local evidence for all parts in the hierarchical dictionary recursively and in parallel for each position, and hence compute the $\phi(x_{\mathcal{H}}, \tau_{\mathcal{H}})$ for all $x_{\mathcal{H}} \in \mathcal{D}_{\mathcal{H}}$ and $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$. The $\max$ and $\arg\max$ operations at level $\mathcal{H}$ can also be done in parallel for each position $x_{\mathcal{H}} \in \mathcal{D}_{\mathcal{H}}$. Similarly we can perform the top-down pass of dynamic programming, see Equation 3.7, in parallel to compute the best configurations of the detected objects in parallel for different possible positions of the objects (on the top-level

lattice).

The parallel formulation can be visualized by making copies of the elements of the hierarchical dictionary elements (the parts), so that a model at level-h has $|\mathcal{D}_h|$ copies, with one copy at each lattice point. Hence at level-h, we have m_h "receptive fields" at each lattice point in $\mathcal{D}_h$ with each one tuned to a different part $\tau_h \in \mathcal{M}_h$, see Figure 3.5. At level-0, these receptive fields are tuned to specific image properties (e.g., horizontal or vertical bars). Note that the receptive fields are highly non-linear (i.e. they do not obey any superposition principle)[8]. Moreover, they are influenced both by bottom-up processing (during the bottom-up pass) and by top-down processing (during the top-down pass). The bottom-up processing computes the local evidence while the top-down pass modifies it by the high-level context.

The computations required by this parallel implementation are illustrated in Figure 3.6. The bottom-up pass is performed by a two-layer network where the first layer performs an AND operation (to compute the local evidence for a specific configuration of the child nodes) and the second layer performs an OR, or $\max$ operation, to determine the local evidence (by max-ing over the possible child configurations)[9]. The top-down pass only has to perform an $\arg\max$ computation to determine which child configuration gave the best local evidence.

3.3 Complexity Analysis

We now analyze the complexity of the inference algorithms for performing the tasks. Firstly, we analyze complexity for a single object (without part-sharing). Secondly, we study the complexity for multiple objects with shared parts. Thirdly, we consider the complexity of the parallel imple-

[8]Nevertheless they are broadly speaking, tuned to image stimuli which have the mean shape of the corresponding part τ_h. In agreement, with findings about mammalian cortex, the receptive fields become more sensitive to image structure (e.g., from bars, to more complex shapes) at increasing levels. Moreover, their sensitivity to spatial position decreases because at higher levels the models only encode the executive summary descriptions, on coarser lattices, while the finer details of the object are represented more precisely at the lower levels.

[9]Note that other hierarchical models, including bio-inspired ones, use similar operations but motivated by different reasons.

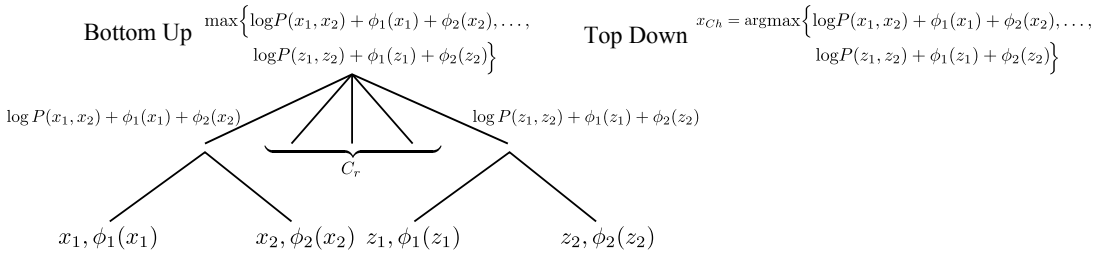


Figure 3.6: Parallel implementation of Dynamic Programming. The left part of the figure shows the bottom-up pass of dynamic programming. The local evidence for the parent node is obtained by taking the maximum of the scores of the C_r possible states of the child nodes. This can be computed by a two-layer network where the first level computes the scores for all C_r child node states, which can be done in parallel, and the second level compute the maximum score. This is like an AND operation followed by an OR. The top-down pass requires the parent node to select which of the C_r child configurations gave the maximum score, and suppressing the other configurations.

mentation.

The complexity is expressed in terms of the following quantities: (I) The size $|\mathcal{D}_0|$ of the image. (II) The scale decrease factor q (enabling executive summary). (III) The number $\mathcal{H}$ of levels of the hierarchy. (IV) The sizes $\{|\mathcal{M}_h| : h = 1, ..., \mathcal{H}\}$ of the hierarchical dictionaries. (V) The number r of subparts of each part. (VI) The number C_r of possible part-subpart configurations.

3.3.1 Complexity for Single Objects and Ignoring Part Sharing

This section estimates the complexity of inference N_{s_o} for a single object and the complexity N_{m_o} for multiple objects when part sharing is not used. These results are for comparison to the complexities derived in the following section using part sharing.

The inference complexity for a single object requires computing: (i) the number N_{bu} of computations required by the bottom-up pass, (ii) the number N_{m_s} of computations required by model selection at the top-level of the hierarchy, and (iii) the number N_{td} of computations required by the top-down pass.

The complexity N_{bu} of the bottom-up pass can be computed from Equation 3.6. This requires a

total of C_r computations for each position x_ν for each level-h node. There are $r^{\mathcal{H}-h}$ nodes at level h and each can take $|\mathcal{D}_0|q^h$ positions. This gives a total of $|\mathcal{D}_0|C_r q^h r^{\mathcal{H}-h}$ computations at level h. This can be summed over all levels to yield:

$$N_{bu} = \sum_{h=1}^{\mathcal{H}} |\mathcal{D}_0|C_r r^{\mathcal{H}}(q/r)^h = |\mathcal{D}_0|C_r r^{\mathcal{H}} \sum_{h=1}^{\mathcal{H}} (q/r)^h = |\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r}\{1-(q/r)^{\mathcal{H}}\}. \tag{3.9}$$

Observe that the main contributions to N_{bu} come from the first few levels of the hierarchy because the factors $(q/r)^h$ decrease rapidly with h. This calculation uses $\sum_{h=1}^{\mathcal{H}} x^h = \frac{x(1-x^{\mathcal{H}})}{1-x}$. For large $\mathcal{H}$ we can approximate N_{bu} by $|\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r}$ (because $(q/r)^{\mathcal{H}}$ will be small).

We calculate $N_{m_s} = q^{|\mathcal{H}|}|\mathcal{D}_0|$ for the complexity of model selection (which only requires thresholding at every point on the top-level lattice).

The complexity N_{td} of the top-down pass is computed from Equation 3.7. At each level there are $r^{|\mathcal{H}|-h}$ nodes and we must compute C_r computations for each. This yields complexity of $\sum_{h=1}^{|\mathcal{H}|} C_r r^{|\mathcal{H}|-h}$ for each possible root node. There are at most $q^{|\mathcal{H}|}|\mathcal{D}_0|$ possible root nodes (depending on the results of the model selection stage). This yields an upper bound:

$$N_{td} \leq |\mathcal{D}_0|C_r q^{|\mathcal{H}|} \frac{r^{|\mathcal{H}|-1}}{1-1/r}\{1-\frac{1}{r^{|\mathcal{H}-1|}}\}. \tag{3.10}$$

Clearly the complexity is dominated by the complexity N_{bu} of the bottom-up pass. For simplicity, we will bound/approximate this by:

$$N_{s_o} = |\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r} \tag{3.11}$$

Now suppose we perform inference for multiple objects simultaneously without exploiting shared parts. In this case the complexity will scale linearly with the number $|\mathcal{M}_{\mathcal{H}}|$ of objects. This gives us complexity:

$$N_{m_o} = |\mathcal{M}_{\mathcal{H}}||\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r} \tag{3.12}$$

3.3.2 Computation with Shared Parts in Series and in Parallel

This section computes the complexity using part sharing. Firstly, for the standard serial implementation of part sharing. Secondly, for the parallel implementation.

Now suppose we perform inference on many objects with part sharing using a serial computer. This requires performing computations over the part-subpart compositions between elements of the dictionaries. At level h there are $|\mathcal{M}_h|$ dictionary elements. Each can take $|\mathcal{D}_h| = q^h|\mathcal{D}|$ possible states. The bottom-up pass requires performing C_r computations for each of them. This gives a total of $\sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|C_r|\mathcal{D}_0|q^h = |\mathcal{D}_0|C_r \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h$ computations for the bottom-up process. The complexity of model selection is $|\mathcal{D}_0|q^{\mathcal{H}} \times (\mathcal{H}+1)$ (this is between all the objects, and the background model, at all points on the top lattice). As in the previous section, the complexity of the top-down process is less than the complexity of the bottom-up process. Hence the complexity for multiple objects using part sharing is given by:

$$N_{p_s} = |\mathcal{D}_0|C_r \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h. \tag{3.13}$$

Next consider the parallel implementation. In this case almost all of the computations are performed in parallel and so the complexity is now expressed in terms of the number of "neurons" required to encode the dictionaries, see Figure 3.5. This is specified by the total number of dictionary elements multiplied by the number of spatial copies of them:

$$N_n = \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h|\mathcal{D}_0|. \tag{3.14}$$

The computation, both the forward and backward passes of dynamic programming, are linear in the number $\mathcal{H}$ of levels. We only need to perform the computations illustrated in Figure 3.6 between all adjacent levels.

Hence the parallel implementation gives speed which is linear in $\mathcal{H}$ at the cost of a possibly large number N_n of "neurons" and connections between them.

3.3.3 Advantages of Part Sharing in Different Regimes

The advantages of part-sharing depend on how the number of parts $|\mathcal{M}_h|$ scales with the level h of the hierarchy. In this section we consider three different regimes: (I) The *exponential growth regime* where the size of the dictionaries increases exponentially with the level h. (II) The *empir-*

ical growth regime where we use the size of the dictionaries found experimentally by compositional learning [ZCT10]. (III) The *exponential decrease regime* where the size of the dictionaries decreases exponentially with level h. For all these regimes we compare the advantages of the serial and parallel implementations using part sharing by comparison to the complexity results without sharing.

Exponential growth of dictionaries is a natural regime to consider. It occurs when subparts are allowed to combine with all other subparts (or a large fraction of them) which means that the number of part-subpart compositions is polynomial in the number of subparts. This gives exponential growth in the size of the dictionaries if it occurs at different levels (e.g., consider the enormous number of objects that can be built using lego).

An interesting special case of the exponential growth regime is when $|\mathcal{M}_h|$ scales like $1/q^h$, see Figure 3.7 (left panel). In this case the complexity of computation for serial part-sharing, and the number of neurons required for parallel implementation, scales only with the number of levels $\mathcal{H}$. This follows from Equations 3.13 and 3.14. But nevertheless the number of objects that can be detected scales exponentially as $q^{\mathcal{H}}$. By contrast, the complexity of inference without part-sharing scales exponentially with q, see Equation 3.12, because we have to perform a fixed number of computations, given by Equation 3.11, for each of an exponential number of objects. This is summarized by the following result.

Result 1: If the number of shared parts scales exponentially by $|\mathcal{M}_h| \propto \frac{1}{q^h}$ then we can perform inference for order $q^{\mathcal{H}}$ objects using part sharing in time linear in $\mathcal{H}$, or with a number of neurons linear in $\mathcal{H}$ for parallel implementation. By contrast, inference without part-sharing requires exponential complexity.

To what extent is exponential growth a reasonable assumption for real world objects? This motivates us to study the empirical growth regime using the dictionaries obtained by the compositional learning experiments reported in [ZCT10]. In these experiments, the size of the dictionaries increased rapidly at the lower levels (i.e. small h) and then decreased at higher levels (roughly consistent with the findings of psychophysical studies – Biederman, personal communication). For these "empirical dictionaries" we plot the growth, and the number of computations at each

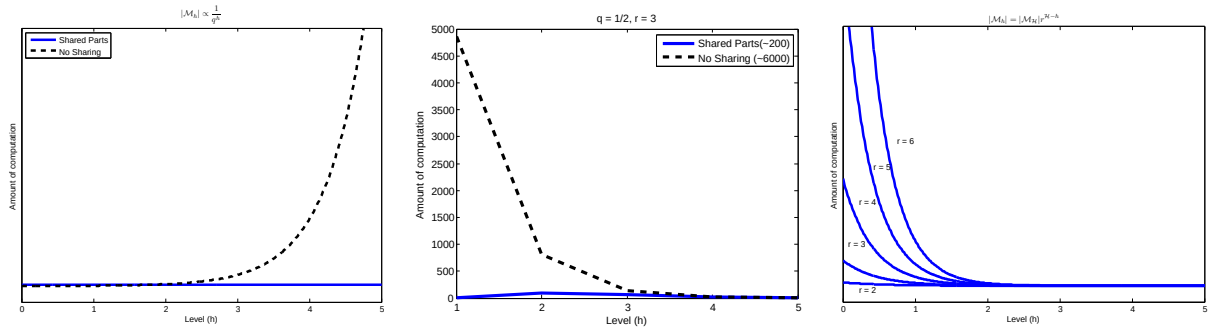
level of the hierarchy, in Figure 3.7 (center panel). This shows complexity which roughly agrees with the exponential growth model. This can be summarized by the following result:

Result 2: If $|\mathcal{M}_h|$ grows slower than $1/q^h$ and if $|\mathcal{M}_h| < r^{\mathcal{H}-h}$ then there are gains due to part sharing using serial and parallel computers. This is illustrated in Figure 3.7 (center panel) based on the dictionaries found by unsupervised compositional learning [ZCT10]. In parallel implementations, computation is linear in $\mathcal{H}$ while requiring a limited number of nodes ("neurons").

Finally we consider the exponential decrease regime. To motivate this regime, suppose that the dictionaries are used to model image appearance, by contrast to the dictionaries based on geometrical features such as bars and oriented edges (as used in [ZCT10]). It is reasonable to assume that there are a large number of low-level dictionaries used to model the enormous variety of local intensity patterns. The number of higher-level dictionaries can decrease because they can be used to capture a cruder summary description of a larger image region, which is another instance of the executive summary principle. For example, the low-level dictionaries could be used to provide detailed modeling of the local appearance of a cat, or some other animal, while the higher-level dictionaries could give simpler descriptions like "cat-fur" or "dog-fur" or simply "fur". In this case, it is plausible that the size of the dictionaries decreases exponentially with the level h. The results for this case emphasize the advantages of parallel computing.

Result 3: If $|\mathcal{M}_h| = r^{\mathcal{H}-h}$ then there is no gain for part sharing if serial computers are used, see Figure 3.7 (right panel). Parallel implementations can do inference in time which is linear in $\mathcal{H}$ but require an exponential number of nodes ("neurons").

Result 3 may appear negative at first glance even for the parallel version since it requires an exponentially large number of neurons required to encode the lower level dictionaries. But it may relate to one of the more surprising facts about the visual cortex in monkeys and humans – namely that the first two visual areas, V1 and V2, where low-level dictionaries would be implemented are enormous compared to the higher levels such as IT where object detection takes places. Current models of V1 and V2 mostly relegate it to being a large filter bank which seems paradoxical considering their size. For example, as one theorist [Len98] has stated when reviewing the functions of V1 and V2 "perhaps the most troublesome objection to the picture I have delivered is that an



(a) (b) (c)

Figure 3.7: The curves are plotted as a function of h. Left panel: This plot is the case where $M_h = a/(q^h)$. So we have a constant cost for the computations, when we have shared parts. Center panel: This plot is based on the experiment of [ZCT10]. Right panel: This plot shows the case where M_h decreases exponentially. The amount of computation is the same for the shared and non-shared cases. A set of plots with different values of r is shown.

enormous amount of cortex is used to achieve remarkably little". Our complexity studies suggest a reason why these visual areas may be so large if they are used to encode dictionaries[10].

3.4 Discussion

This chapter provides a complexity analysis of what is arguably one of the most fundamental problem of visions – how, a biological or artificial vision system could rapidly detect and recognize an enormous number of different objects. We focus on a class of hierarchical compositional models [ZLH08, ZCT10] whose formulation makes it possible to perform this analysis. But we conjecture that similar results will apply to related hierarchical models of vision.

Technically we re-formulated compositional models so that they can be defined on regular lattices (which makes them easier to compare to alternatives such as deep belief networks) and a novel parallel implementation. The analysis also clarifies the use of part-sharing to perform exact inference even on highly complex models. We note that the re-use of computations in this manner

[10]Of course, this is extremely conjectural.

might relate to methods developed to speed up inference on graphical models, which gives an interesting direction to explore.

Finally, we note that the parallel inference algorithms used by this class of compositional models have an interesting interpretation in terms of the bottom-up versus top-down debate concerning processing in the visual cortex [DZR12]. The algorithms have rapid parallel inference, in time which is linear in the number of layers, and which rapidly estimates a coarse "executive summary" interpretation of the image. The full interpretation of the image takes longer and requires a top-down pass where the high-level context is able to resolve ambiguities which occur at the lower levels. Of course, for some simple images the local evidence for the low level parts is sufficient to detect the parts in the bottom-up pass and so the top-down pass is not needed. But more generally, in the bottom-up pass the neurons are very active and represent a large number of possible hypotheses which are pruned out during the top-down pass using context, when "high-level tells low-level to stop gossiping".

CHAPTER 4

A Flat Compositional Model

In the previous two chapters, we presented hierarchical compositional models and their complexity analysis. In this chapter, we propose a flat compositional model. The proposed hierarchical models assumed a fixed number of sub-parts for a part. In this chapter, we relax that constraint. Additionally, the edgelets used in the hierarchical models are sensitive to local deformations. In this chapter, we propose a new representation for parts that are more robust compared to edgelets. Furthermore, we augment these models by appearance-based models.

To better model the variations in the appearance or 3D viewpoint of the objects, in contrast to the current part-based methods (such as [CH07] and [FGM10]), we do not specify the number of object models or their constituent parts. These numbers and the parameters corresponding to the spatial relationship of the object parts are determined automatically by the learning method. We tackle the difficult problem of simultaneous structure and parameter learning by introducing a new compositional learning approach.

Our compositional learning method proceeds by building part-based models by combining elementary components incrementally. This can be seen as a type of breadth first search in the space of object models. These part-based models are generative which enables our learning process to use model selection criteria. The learning only requires positive examples of each object category and is supervised in the sense that we know the label and bounding box of the objects during learning. However, we show the method has the capability of learning multiple 2D models to describe an object seen from different viewpoints without using any 3D viewpoint labels. The learning is also invariant to scale and in-plane rotation.

The object parts correspond to *HOG-bundles*, which are grouping of HOG features [DT05]

and are computed from images in a pre-processing step. The HOG-bundles typically correspond to parts of the object, and so our object models can be used to parse the object into parts (though this is not the main goal of this chapter). We also use histograms of vector quantized PHOW features [BZM07] for modeling the global appearance of the object. Overall, the HOG-bundles, supplemented by PHOW features, provide a rich and intuitive representation of the object.

This chapter is organized as follows. Section 4.1 describes the image features that we use and, in particular, defines the HOG-bundles. Section 4.2 describes the part-based model and its inference algorithm. Section 4.3 describes how the part-based models are learnt. Section 4.4 introduces the appearance-based models and describes how these are combined with the part-based models. Section 4.5 describes the implementation and gives the results on the datasets that include single and multi-view images of objects.

4.1 Image Features – HOG Bundles and Bags of Words

Our object models use two types of image features which are extracted from the image in a pre-processing stage: Histograms of Oriented Gradients (HOGs) [DT05] and PHOW [BZM07], which are used by the part-based and by the appearance-based models respectively.

An image $\mathbf{I}$ is represented by $\mathbf{I} = \{\mathbf{z}_i : i = 1, ..., N\}$ in terms of HOG-bundles described by $\mathbf{z} = (\mathbf{r}, \theta, \mathbf{f})$, where $\mathbf{r}$ is the position, θ is the orientation, and $\mathbf{f} = (f_h, f_w)$ is the height f_h and width f_w of the bundle. The number of HOG-bundles in an image is denoted by N. This can be supplemented – for the appearance-based models – by PHOW features $Ph(\mathbf{r})$ as a function of position $\mathbf{r}$.

The part-based models use HOG-bundles because they are robust to local variations in shape and image intensity. They also provide a richer description of the object than interest points, which are often used to capture the salient structures of the object. Moreover, compared to features such as the edge features used in [ZCY10], there are advantages to using HOG-bundles for learning because: (i) each HOG-bundle has distinguishing attributes (e.g., size, height and width), and (ii) there are only a few hundred of them in each image. The HOG features are computed following

[FGM10] where the orientations are quantized into 18 bins, resulting in a 31-dimensional feature vector for each HOG cell.

We compute *HOG-bundles*, by grouping neighboring HOG cells which share similar properties using the following grouping rules: Two HOG cells are grouped if they are neighbors in the grid image and satisfy the following criteria:

- The difference between the feature vectors of two cells are small, as computed by the χ^2 distance function over the feature vectors.
- The orientation with the maximum magnitude should be similar for two HOG cells. Usually, the cells that belong to a part have a similar orientation.
- HOG cells with orientation in many directions will not be grouped, since they usually correspond to randomly textured areas such as grass. This is quantified by the squared difference $|\Omega - \omega|^2$, where Ω is the maximum magnitude of the orientation part of the feature vector, and ω is the mean of the magnitudes. However, we group the cells that correspond to uniform intensity regions (low-magnitude gradient cells).

To build the HOG-bundles, we start from an arbitrary cell in the image and check if its neighbors satisfy the grouping criteria. If they do so, we group them and check for the neighbors of neighbors until all of the cells in the image are processed. Each HOG-bundle is approximated by a rectangle and has the following attributes: position of the center ($\mathbf{r}$), width (f_w), height (f_h), and orientation (θ), which is the mean of the orientations with the maximum magnitude in the histograms of the constituent cells.

The HOG-bundles can overlap because the grouping is performed based on two different parts of the HOG feature vector representing the gradients with left-to-right and right-to-left orientations (i.e. we consider dimensions 1 to 9 and 10 to 18, separately).

In addition to HOG-bundles that mainly capture the object contours and uniform gradient regions on the object, we use the PHOW features [BZM07] (a variant of SIFT computed at multiple scales) in our model to capture regional appearance properties of the objects. We have adopted

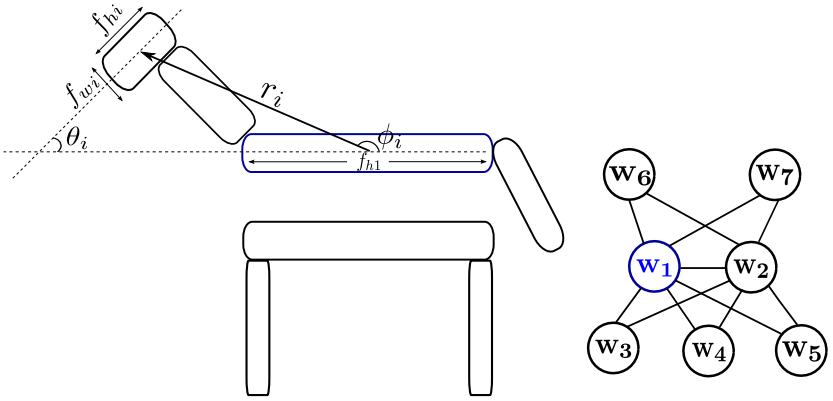


Figure 4.1: Left: the geometry of the part-based models illustrating $\theta, \phi, f_h, f_w, r$. The first reference part (corresponding to $\mathbf{w}_1$) is shown in blue. Right: the graphical model. All parts are connected to the two reference parts $\mathbf{w}_1, \mathbf{w}_2$. These reference parts are chosen by the learning algorithm.

the implementation by [VF08]. These features are relatively invariant to local spatial and intensity changes. We denote these features as $Ph(\mathbf{r})$, and they are computed densely as a function of position $\mathbf{r}$. Within any image window, we can compute the histogram $\mathcal{H}(Ph(.))$ of the PHOW features using the standard clustering techniques.

4.2 Part-based object models and inference

This section describes the part-based object models and how we perform inference using them (the learning is described in the next section). Each object O will have several different models indexed by $\tau = 1, \ldots, T_O$. In multi-view recognition tasks, these models typically correspond to different views of the object. The number of these models is unknown and will be learnt automatically.

4.2.1 The Object Models

An Object O will be represented by a set (e.g. a mixture) of models $P(\mathbf{W}|O, \tau)$, where τ indexes the mixture component and $\mathbf{W}$ denotes the state variables defined below. For notational simplicity we ignore the indices O and τ in Section 4.2.1 and reintroduce them in Section 4.2.2. Each part-

based model is a graphical model with state variables $\mathbf{W} = \{\mathbf{w}_i : i = 1, ..., M_O\}$, where $\mathbf{w}_i = (\mathbf{r}_i, \theta_i, \mathbf{f}_i)$ represents the position $\mathbf{r}_i$, the orientation θ_i, and the feature properties $\mathbf{f}_i$ of the i^{th} part. The feature properties can be decomposed into $\mathbf{f} = (f_w, f_h)$ where f_w and f_h describe the width and height of a HOG-bundle. Figure 4.1 visualizes these terms for an example bundle and also represents an example graphical model for the object. Our graphical model is similar to the 2-fan model of Crandall et al. [CFH05] but as described in Section 4.3, our learning and inference procedures are quite different since we learn the graph structure and the number of models as well.

Probabilistic modeling of these part-based models requires the specification of a prior distribution on them and also a likelihood function. The prior probability is of form, see Figure 4.1 (right):

$$P(\mathbf{W}|\mathbf{\Lambda}) = P(\mathbf{w}_1)P(\mathbf{w}_2|\mathbf{w}_1, \boldsymbol{\lambda}_2) \prod_{i=3}^{M_O} P(\mathbf{w}_i|\mathbf{w}_1, \mathbf{w}_2, \boldsymbol{\lambda}_i), \tag{4.1}$$

where the model parameters $\mathbf{\Lambda} = (\boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_{M_O})$, and the number of parts M_O will be learnt from the data, as described in Section 4.3. The form of the model enables efficient inference, invariant to scale and in-plane rotation, as discussed in Section 4.2.2.

We specify the probability distributions of Equation 4.1 as follows. First, we define a coordinate change $(\mathbf{r}_i - \mathbf{r}_1) = r_i(\cos\phi_i, \sin\phi_i)$ in radial coordinates based on the position $\mathbf{r}_1$ of the first part (the first reference part). We define $P(\mathbf{w}_1)$ to be the uniform distribution $U(\mathbf{w}_1)$. We also assume that the spatial and feature terms are independent:

$$\begin{aligned} P(\mathbf{w}_i|\mathbf{w}_1, \mathbf{w}_2) &= P(\mathbf{f}_i|f_{h1})P(\phi_i|\phi_1)P(\theta_i|\phi_1)P(r_i|\mathbf{r}_1, \mathbf{r}_2) \\ P(\mathbf{w}_2|\mathbf{w}_1) &= P(\mathbf{f}_2|f_{h1})P(\phi_2|\phi_1)P(\theta_2|\theta_1)P(r_2|\mathbf{r}_1), \end{aligned} \tag{4.2}$$

where f_{h1} represents the height of the bundle corresponding to the first reference part. It should be noted that the two reference parts are chosen by the learning algorithm automatically.

We specify these distributions in terms of Gaussian and uniform distributions, using the notation that $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. These distributions are chosen to ensure invariance to the scale of the features, the orientation of the object, and the scale of the object (features sizes and orientations are defined relative to those of the first part, and the

relative positions are scaled by the distances between the first two parts).

$$
\begin{aligned}
&P(\mathbf{f}_i|f_{h1}) = \mathcal{N}(f_{h1}\boldsymbol{\mu}_i^{\mathbf{f}}, f_{h1}^2\sigma_{\mathbf{f}}^2),\ P(\phi_i|\phi_1) = \mathcal{N}(\mu_i^{\phi} + \phi_1, \sigma_{\phi}^2),\\
&P(\theta_i|\phi_1) = \mathcal{N}(\mu_i^{\theta} + \phi_1, \sigma_{\theta}^2),\ P(r_i|\mathbf{r}_1, \mathbf{r}_2) = \mathcal{N}(r_2\mu_i^r, r_2^2\sigma_r^2),\\
&P(r_2|\mathbf{r}_1) = U(r_2).
\end{aligned} \tag{4.3}
$$

The model parameters are the mean feature properties and angles $\{(\boldsymbol{\mu}_i^{\mathbf{f}}, \mu_i^{\phi}, \mu_i^{\theta}) : i = 2, ..., M_O\}$ and positions $\{\mu_i^r : i = 3, ..., M_O\}$. These are learnt from the training data. There are an additional four parameters which are fixed $\sigma_{\mathbf{f}}^2, \sigma_{\phi}^2, \sigma_{\theta}^2, \sigma_r^2$ and set manually.

The likelihood function assumes that the HOG-bundles $\{\mathbf{z}_i : i = 1, ..., N\}$ in the image are generated either from the object model $P(\mathbf{I}|\mathbf{W})$, or from a *background model* $P_B(\mathbf{z})$ which generates HOG-bundles independently. There is a default *background model* which assumes that $P_B(\mathbf{I}) = \prod_{i=1}^{N} P_B(\mathbf{z}_i)$, i.e. each HOG-bundle is generated independently by $P_B(.)$, where N is the number of HOG-bundles in the image. We define $P(\mathbf{I}|\mathbf{W}) = \prod_{i=1}^{M_O} \delta(\mathbf{z}_i, \mathbf{w}_i)$, where $\delta(\mathbf{z}, \mathbf{w}) = 1$ if $\mathbf{z} = \mathbf{w}$, and equal to zero otherwise i.e. the HOG-bundles generated by the object model have the same size, positions, and orientations as the state variables $\mathbf{w}$ of the object parts. For simplicity we set $P_B(.) = U(.)$ the uniform distribution. Hence the likelihood function for an image assumes that:

$$
\begin{aligned}
&\mathbf{z}_i : i = 1, ..., M_O \text{ sampled from } P(\mathbf{I}|\mathbf{W})P(\mathbf{W})\\
&\mathbf{z}_i : i = M_O + 1, ..., N \text{ sampled from } P_B(\mathbf{z}).
\end{aligned} \tag{4.4}
$$

4.2.2 Inference

The inference task determines if there is an object in the image and, if so, where it is. Recall that each object O can have several different part-based models indexed by τ. These are expressed as $P(\mathbf{W}|O, \tau)$, which is of the form given by Equation 4.1 with parameters $\Lambda_{O,\tau}$ which depend on the object O and the object model τ. All part-based models for the same object have the same number of parts and the likelihood functions are the same for all part-based models.

There are two types of inference tasks. Firstly, to find whether the image contains an object, and if so, to determine the object and type. Secondly, to determine where the object is in the image, i.e. to determine its configuration **W**.

Detecting the optimal configuration for each object and type O, τ requires solving:

$$\hat{\mathbf{W}}_{O,\tau} = \arg\max_{\mathbf{W}} P(\mathbf{I}|\mathbf{W})P(\mathbf{W}|O,\tau). \tag{4.5}$$

The form of the likelihood term (4.4) means that the state variables $\{\mathbf{w}_i : i = 1, ..., M_O\}$ can only take values $\{\mathbf{z}_i : i = 1, ..., N\}$ from the HOG-bundles computed from the image. The form of the part-based models in Equation 4.1, means that we can express Equation 4.5 as minimizing a function of form $E(\mathbf{w}_1, \mathbf{w}_2) + E(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) + ... + E(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_{M_O})$. $E(.)$ is the negative logarithm of the probability, and so is a sum of quadratic terms if the distributions are Gaussian.

Inference can be performed in polynomial time using dynamic programming (DP). Our models are graphical models with a limited number of closed loops and so DP can be applied using a variant of the junction trees algorithm. Inference starts by evaluating all combinations for the first two (reference) parts. Because of the form of the potential terms, DP is very efficient for the remaining parts. For example, to find the fourth part, for each configuration $(\mathbf{w}_1, \mathbf{w}_2)$ of the two reference parts, we need to find only $\mathbf{w}_4$ minimizing $E(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_4)$, which involves simple calculations. This enables us to rapidly find the lowest energy configurations of each model. Candidate object configurations are the ones whose overall energy is below a threshold.

We perform model selection to determine which object, if any, is present in the image, and its type. The object and its type (O_i, τ_i) are calculated such that:

$$P(\mathbf{I}|\hat{\mathbf{W}}_{O,\tau})P(\hat{\mathbf{W}}_{O,\tau}|O,\tau) > \prod_i P_B(\mathbf{z}_i), \tag{4.6}$$

Strictly speaking, model selection should sum over all possible configurations of the models but, in practice, the locations are strongly peaked so we replace the sum by the dominant term. We assume $P_B(.)$ is a constant so this model selection reduces to keeping model configurations for which the probability $P(\mathbf{I}|\mathbf{W})P(\mathbf{W}|O,\tau)$ lies above a threshold.

For each image, this gives a set of n models and types (O_i, τ_i) together with their configurations

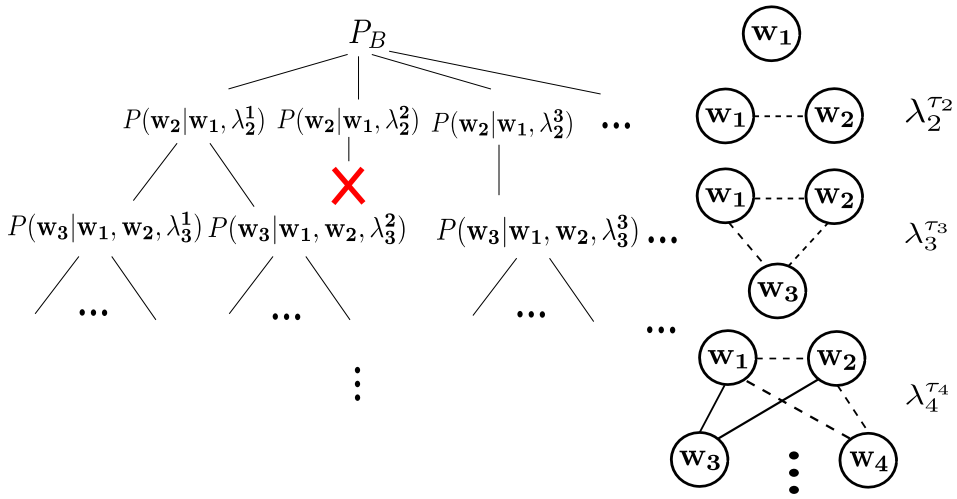


Figure 4.2: The compositional learning proceeds by adding an extra node to the model and estimating the parameters for the newly added part. $\boldsymbol{\lambda}_i^{\tau_i}$ denotes the parameters corresponding to the i^{th} part of the τ_i^{th} model. An example model that cannot be expanded, because it fails model selection or is pruned by the frequency criteria, is shown by the red cross.

$\hat{\mathbf{W}}_{O_1,\tau_1}, ..., \hat{\mathbf{W}}_{O_n,\tau_n}$. These denote possible classifications of the object and possible detection locations for it (if $n = 0$, then no object is detected). These candidate models and configurations are then combined with the results of the appearance-based model, as described in Section 4.4. Note that the part-based models described here need to be supplemented by the additional appearance cues specified by the appearance-based models.

Our inference is robust against occlusions to some extent. For example, if a model has 8 parts and 6 parts are detected with low partial energy we report this as a detection of the object (despite the two missing parts). Alternatively, we could explicitly incorporate occlusion into our likelihood term similar to [FPZ03].

4.3 Compositional Learning

The learning algorithm involves estimating the unknown (hidden) variables of the problem, namely, M_O (number of parts), T_O (number of models/viewpoints), the correspondence between the fea-

tures $\mathbf{z}$ and the variables $\mathbf{w}$, and $\mathbf{\Lambda}_{O,\tau}$ (the parameters of the model). Our situation is more complex compared to the approaches that assume M_O and T_O are known and can apply the standard EM (e.g. [CH07]). Due to the complexity of the problem, EM will not work for our case, hence, we propose a novel compositional learning algorithm to tackle the problem.

Our strategy is to build models by searching through the space of possible models and applying a set of composition rules to a *root* model. These compositional rules take n-part models and add an extra part to form $n+1$-part models. Model selection is used to check that the new models give better descriptions of the data. The procedure terminates automatically when this is not the case. This can be thought of as a breadth-first search over the space of all object models. This procedure is done for each object separately so we ignore the object label O, keeping only the type τ. The procedure is shown in Figure 4.2. The idea of learning using compositions is similar to [ZCY09] but they use a depth-first greedy search strategy.

The input is a set of images indexed by t where each image is represented by its HOG-bundles $\{\mathbf{z}_i^t : i = 1, ..., N_t\}$. We assume a default distribution for the images is specified by the background distribution: $P_B(\mathbf{I}^t) = \prod_{i=1}^{N_t} P_B(\mathbf{z}_i^t)$.

We first construct probability distributions defined for two-parts of form $P(\mathbf{w}_1, \mathbf{w}_2|\boldsymbol{\lambda}_2^{\tau_2})$, where τ_2 indexes these models. The parameters $\boldsymbol{\lambda}_2$ are determined by a clustering algorithm, described later in this section, which ensures that these models provide a better fit to the data. In a sufficient fraction η of images, this two-part model is matched (by the inference algorithm) to HOG-bundles $\mathbf{z}_i^t, \mathbf{z}_j^t$. The model selection requires that the two-part model, filling in the remaining features by the background model, gives a better fit than the pure background model – i.e.:

$$P(\mathbf{z}_j^t|\mathbf{z}_i^t, \boldsymbol{\lambda}_2^{\tau_2}) \prod_{k \neq j} P_B(\mathbf{z}_k^t) > \prod_i P_B(\mathbf{z}_i^t).$$
$$P(\mathbf{z}_j^t|\mathbf{z}_i^t, \boldsymbol{\lambda}_2^{\tau_2}) > P_B(\mathbf{z}_j^t). \tag{4.7}$$

This requirement gives us a family of two-part models indexed by τ_2 each of which is better for describing a sufficient fraction η of the images (as described in Equation 4.7) than the pure background model. We store these models and then grow them by adding extra parts. Each $n-1$ part model is specified by a set of parameters $\boldsymbol{\lambda}_2^{\tau_2}, ..., \boldsymbol{\lambda}_{n-1}^{\tau_{n-1}}$ and can be extended to an n-part model

by specifying new parameters $\boldsymbol{\lambda}_n^{\tau_n}$, which determine the probability $P(\mathbf{w}_n|\mathbf{w}_1, \mathbf{w}_2)$ for the state of the n^{th} part in relation to the states of the first two parts $\mathbf{w}_1, \mathbf{w}_2$. We extend the model selection criterion from Equation 4.7 in the natural way to obtain the requirement that $P(\mathbf{z}_n^t|\mathbf{z}_i^t, \mathbf{z}_j^t, \boldsymbol{\lambda}_n^{\tau_n}) > P_B(\mathbf{z}_n^t)$ for a sufficient fraction η of images. Or in other words, the requirement that the n-part model gives a better description of a sufficient fraction of the images than the previous $n-1$-part model. We repeat until we fail to generate models with more parts. The learning procedure also stops in the case that at least one of the training images is not described by the newly created models. It should be noted that each n-part model is a child of an $n-1$-part model, and by construction the n-part model describes the data better than its parent (but does not necessarily give a better description than other $n-1$ part models).

Several grouping criteria can be used to estimate the parameters $\boldsymbol{\lambda}$. Equation 4.7 cannot be applied directly. Therefore, we have experimented with two clustering methods, the DBSCAN [EKS96] and Affinity Propagation algorithms [BD07], which give roughly equal success and neither of them requires a pre-determined number of clusters. For example, DBSCAN performs a sequential search in feature space using clustering procedures equivalent to thresholding the left hand side of Equation 4.7 (provided they are Gaussian with fixed variance). The $\boldsymbol{\lambda}$ parameters correspond to the mean and variance of the clusters. After performing clustering, we can evaluate model selection in a validation step. The reported results in the experiments section are based on the Affinity Propagation method.

The learning procedure results in a set of part-based models for each object indexed by τ. For objects viewed from different viewpoints this includes models capturing each viewpoint. We prune the set of models based on two additional criteria: (i) remove those whose bounding box are significantly smaller than the bounding boxes of the training data, and (ii) eliminate those models which occur least frequently in the training images.

4.4 The appearance-based model

The part-based model described in Sections 4.2 and 4.3 is limited because it only uses appearance cues that can be represented by HOG-bundles. These correspond to dominant edges of the objects and, sometimes, regions with uniform gradients. Hence the models are generally poor at dealing with regional appearance cues.

In this section, we augment the part-based model with additional cues which are sensitive to regional properties of the objects. This corresponds to supplementing the HOG-bundles with the additional PHOW features, so that $\mathbf{I} = (\{\mathbf{z}_i\}, Ph(\mathbf{r}))$ where $Ph(\mathbf{r})$ are the PHOW features (refer to Section 4.1). This introduces a new *appearance variable* $\mathbf{w}_A$ for the model, which corresponds to the region occupied by the object. In addition, we add a new likelihood term, which couples the appearance variable to the histograms of PHOWs $\mathcal{H}(Ph(.))$ computed in the corresponding image region:

$$P(\mathcal{H}(Ph(.))|\mathbf{w}_A, O, \tau) = \frac{1}{Z} e^{-\min_a \mathcal{M}(\mathcal{H}(Ph(.)), \mathcal{H}_a^{O,\tau})}. \tag{4.8}$$

where $\mathcal{M}(.,.)$ is a measure of similarity between the histogram $\mathcal{H}(Ph(.))$ computed in the image region $\mathbf{w}_A$ and the histogram of one of several 'prototype histograms' $\mathcal{H}_a^{O,\tau}$ indexed by a for object and type O, τ. These prototypes are the histograms of the regions in the training images surrounded by object bounding boxes. The model chooses the nearest prototype using the $\min$ operation. We assume a default distribution $P(\mathcal{H}(Ph(.))$ to be uniform in regions where the object is not present. We specify $\mathbf{w}_A(\mathbf{W})$ to be a deterministic function, e.g. bounding box, of the state variables $\mathbf{W}$ estimated from the part model.

During inference, we estimate $\hat{\mathbf{W}}_{O,\tau}$ for each object type O, τ by Equation 4.5. Then we compute $\mathbf{w}_A(\hat{\mathbf{W}}_{O,\tau})$ to obtain the position of the bounding box, followed by computing the overall fitness score for the object type by combining the contributions from the parts model and the appearance model.

This procedure for combining the appearance-model with the part-model is suboptimal: (i) the generative model is hand specified and not learnt from the data, (ii) the appearance variable $\mathbf{w}_A$ is a deterministic function of the part-based variables $\mathbf{W}$ but its relationship to them should be learnt,

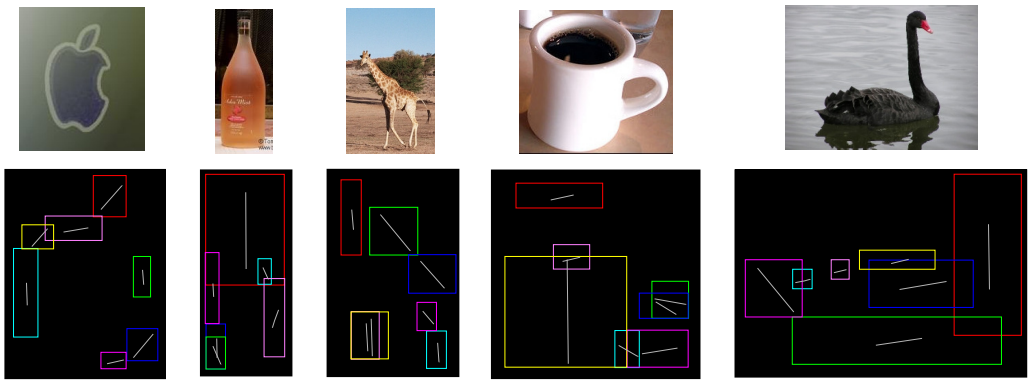


Figure 4.3: One of the learned models for each category of the ETHZ dataset is shown. The rectangles represent the HOG bundles and the line shows the dominant orientation of the HOG bundle. The number of parts and their relative position and orientation is determined by the learning procedure.

(iii) inference should estimate $\mathbf{W}$ and $\mathbf{w}_A$ together, instead of estimating $\mathbf{W}$ from the parts model and then computing $\mathbf{w}_A$, (iv) when combining the part and the appearance cues we should compute the normalization term.

Nevertheless, we obtain reasonable results using our current procedure. A method that addresses these issues (following [CZY09], for e.g.) would presumably perform even better.

4.5 Implementation and Results

In order to validate our method, we learned object models for the categories of the ETHZ dataset [FJS09] and the INRIA horses [FFJ08]. Additionally, to show the performance of our method for 3D multi-view recognition, we applied our learning method to a multi-view car dataset [SSF09], where the viewpoint labels were unknown to the learning method. After learning, the inference is performed in two stages. First, a set of candidate part configurations are obtained by thresholding the distributions of the part-based models (Equations 4.5 and 4.6). Then, we score each candidate bounding box with the appearance-based model (Equation 4.8). We plot the result curves by varying a threshold over these scores.

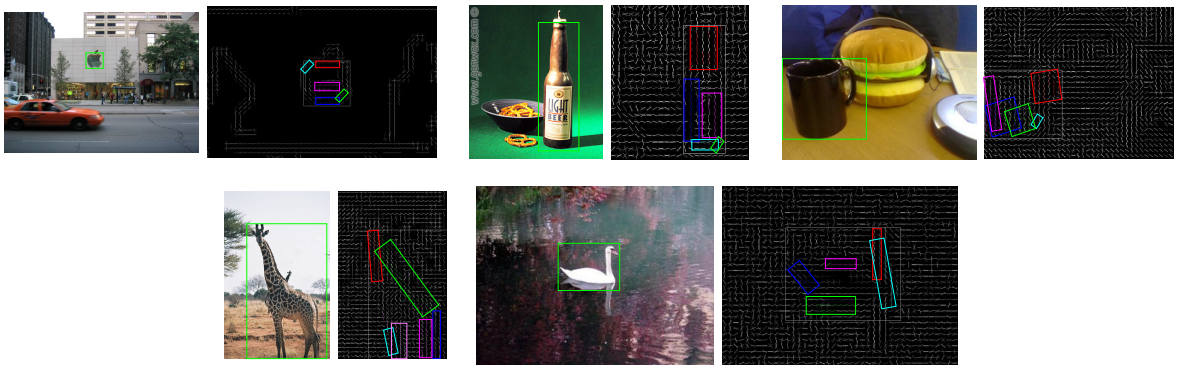


Figure 4.4: **(Better viewed if zoomed in)** Parsing and detection results of our inference method. Parts of the objects are shown with different colors. The number of detected parts are not necessarily the same as the number of parts in the full models. We have zoomed in the apple and swan inference images for better quality but the inference was performed on the full size image.

Initially, we show the performance of the method on single view datasets and then we show how the same learning procedure can be adopted to learn models for different 3D viewpoints of objects. ETHZ dataset consists of five object categories: Apple logos, Bottles, Giraffes, Mugs and Swans (255 images in total). The evaluation protocol for the ETHZ dataset is as follows. Half of the images of one category are used for learning and the other half and all of the images of the other categories are used for testing. Our part-based models are learned using only 6 training images. The appearance-based models use half of the images of one category (including those six used for part learning). We could use half of the training images for learning the part-based models as well but the learning becomes inefficient as the number of compositions of HOG-bundles grows. We do not use any negative examples and the reported results are obtained from 5 trials of random selection of training images. We also pruned the models that appeared in less than two-third of the training images (the parameter η in Section 4.3). One of the learned models for the ETHZ categories is shown in Figure 4.3.

We have compared our inference results quantitatively with the results of some of the recent methods [FFJ08, FJS09, VK10, SZS10] in Figure 4.5. Also, we compare our results with the result of Felzenszwalb et al.'s code [FGM10], which is obtained by [SZS10]. Detection Rate versus False Positive per Image (FPPI) is used as the evaluation criteria. Our result is close to the others and

we obtain the state-of-the-art results on Giraffe and Horse categories at 0.3/0.4 FPPI. The state-of-the-art for the horse category at 1.0 FPPI is obtained by [TTD10] but their result is much worse than ours in the low FPPI regions. It should be noted that the dataset is small and an error in one image results in a significant drop in the curves. The reason that our method does not perform well on Mugs and Bottles categories is that their part-based models are usually rectangular and the rectangular structures are common in man-made environments. Also, the appearance-based model is not strong since the textures on the mugs and bottle labels vary greatly. Some examples of bounding box detection and object parsing are shown in Figure 4.4.

We now describe the multi-view recognition result that is obtained by learning multiple 2D models to describe different 3D viewpoints. The models are learned jointly and the viewpoint labels of the training images are not provided to the learning method. We provided our learning method with images of cars used by [SSF09] which include images of 8 viewing angles, 3 scales, and 2 heights. Similar to [SSF09] and [HZ10], we use the first 5 instances for training (240 images, 24 of which were used for the part-based model), and the remaining 5 for testing (240 images). Unlike [SSF09] which does not use the smallest scale, we perform the detection task on all of the scales. Our method outperforms [SSF09] and [HZ10] that use 3D cues in addition to the 2D cues. Stark et al. [SGS10] obtain high performance on this dataset, but it is not fair to compare their method with ours as they use an approach based on 3D CAD models, which is highly specialized and is not purely image based. Their reported average precision is 89.9%. Our result together with two of the learned models representing two different viewpoints are shown in Figure 4.6. The η parameter for this experiment is 0.125.

Our method is not too sensitive to the scale of HOG cells and we only used 6×6 windows to compute the HOG features in all of our experiments. Our inference takes about a second on average for 10 models with 6 parts on a 200×150 image on a desktop with a 2.66 GHz CPU.

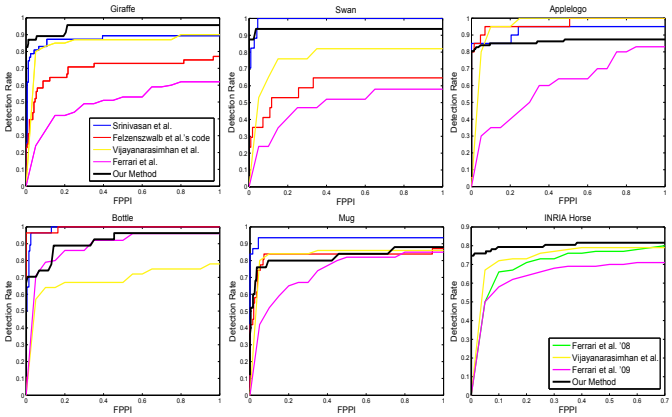


Figure 4.5: Quantitative comparison of our results with [SZS10], [VK10],[FGM10], [FJS09] and [FFJ08]. The PASCAL criterion is used for the evaluation. Detection Rate versus False Positives per Image (FPPI) is shown. Unlike the other approaches, we do not use negative examples.

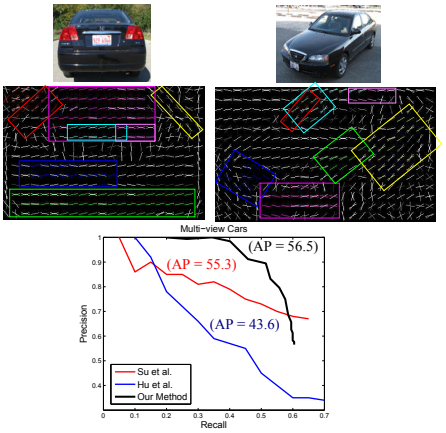


Figure 4.6: The curves show the result of our method (black line) compared to [HZ10] and [SSF09] in terms of precision-recall. Two of the automatically learned models representing two different viewpoints are also shown.

4.6 Conclusion

We proposed a novel approach for learning part-based models of objects. We also introduced *HOG-Bundles* as a novel representation for object parts and used them as the building blocks of our part-based model. The advantage of using HOG-bundles is that they are robust against local deformations of objects, and each image contains only a small number of them. We augmented our part-based models by a global appearance model based on PHOW features to model the regional properties of objects. The obtained results are remarkable even though we do not use any negative examples.

Our learning method learns the structure of the graphical model and its parameters simultaneously. The reason for estimating the number of object models and the number of parts from the training data is to provide a better description for intra-class variability and the changes in the viewpoint. The standard EM algorithm seems impractical in this situation, where the number of graph nodes and the number of models are hidden variables as well.

Our learning method is invariant to scale and in-plane rotations but to capture general 3D rotations, we need to provide enough training images of different viewpoints. We performed our experiments in a supervised setting, where we knew the object bounding box and object label during learning. However, by applying our method on a multi-view car dataset, we showed that the labeling constraint can be relaxed, and also the variation in the 3D viewpoint can be modeled by a set of 2D models.

CHAPTER 5

Deformation Modeling

The methods we have described so far are generative learning methods. In the rest of this thesis, we focus on discriminative methods for object detection, which have proven to be successful in discriminating objects from the background.

Part-based object detectors have shown remarkable performance in recent years [BM09, CZY10, CH07, FGM10, FH05, FL07, SRS10, ZCY10]. Among them, the Deformable Part Model (DPM) by Felzenszwalb et al. [FGM10] and its variants such as [CZY10] and [ZCY10] demonstrate the state-of-the-art performance on difficult object detection benchmarks. However, there are several limiting factors that prevent these types of part-based detectors from achieving the ideal performance. A major challenge is to define a topological structure for the object model. The structure should be flexible enough to capture the deformations of objects. On the other hand, learning and inference should be performed efficiently on these models, which often restricts the flexibility of the structure. Although impressive results are achieved using star structures [FGM10] or hierarchies of deformable parts [ZCY10], these structures are not flexible enough to reliably capture variations in highly deformable objects such as cats or plants.

Additionally, specifying parts as rectangular blocks in DPMs imposes even more restriction on modeling the flexibility of objects. Rectangular blocks of features have been used as a common representation for parts in object detectors (e.g., [BM09, CH07]). Although rectangular blocks capture the shape or context around the parts, they might not be the most appropriate choice for deformable objects due to the huge space of part deformations.

In this chapter, we propose a method to improve the family of Deformable Part Models ([FGM10] and its variants) to better capture large deformations. Hence, we develop a framework to integrate

the detections of DPMs with patches of irregular shape (HOG-bundles). Firstly, we extend the idea of HOG-bundles (refer to Chapter 4). In the previous chapter, we approximated HOG-bundles by rectangular blocks and used only width and height of the HOG-bundles for describing them. However, width and height do not carry much information, especially for bundles with irregular shape. We propose a set of descriptors for the bundles so we recognize the object category they belong to. There are advantages in using HOG-bundles in our framework: (i) they have irregular shape allowing better representation of the subparts of highly deformable objects, (ii) they are computed efficiently (less than 0.5s per image on a CPU), (iii) there are only a few hundred of them in each image. Example HOG-bundles are shown in Fig. 5.1.

Secondly, we formulate the problem of augmenting the detections of Deformable Part Models with HOG-bundles as the MAP estimation of a multi-label CRF. The goal is to find the labeling that minimizes a defined energy function, where the node label specifies to which object of the scene a pixel belongs. As a result, our method extends the detections with missing parts, shrinks the detections that include background clutter, and increases the confidence of the detections with low confidence (caused by partial occlusion or unusual pose).

A number of methods have been proposed to overcome the deficiencies of DPMs in modeling the flexibility of objects. For example, Girshick et al. [GFM11] have proposed an object grammar to handle flexibility and partial occlusion of objects. Their result is limited only to people detection. Schnitzspan et al. [SRS10] describe a method for automatic discovery of parts and their topological structure. Parkhi et al. [PVJ11] improve the detection of cats and dogs by training a DPM for a distinctive part (head detector) and extending the detected regions using color cues. The advantage of our method to theirs is that we do not require a distinctive part detector. Also, our method generalizes to other categories as we do not rely on color cues only.

Several works, for example [GBW10, LRK09, LJ08] to mention a few, have used CRFs with or without object detection priors to address the problem of object class segmentation. These approaches specify the object class that a pixel belongs to but they cannot determine how many instances of an object category are present.

OBJ CUT, proposed by Kumar et al. [KTZ05], combines CRFs with Pictorial Structures to

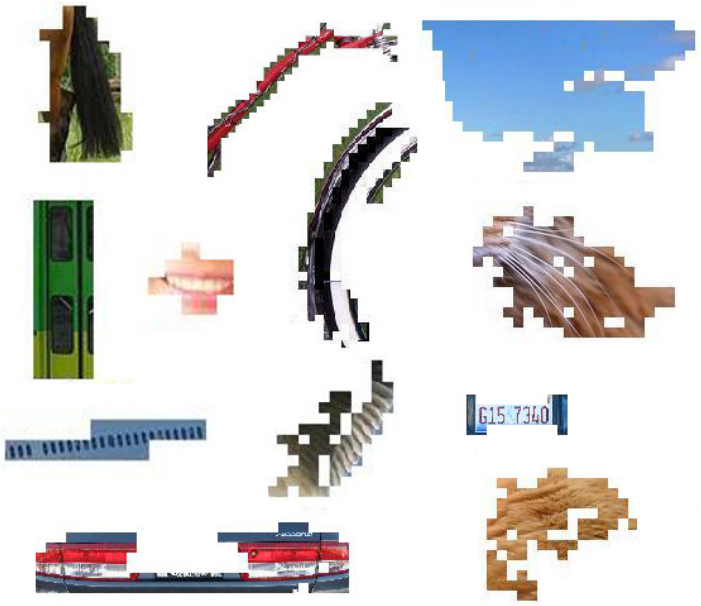


Figure 5.1: Examples of HOG-bundles. From top-left to bottom-right: horse tail, bicycle frame, sky, bus door, lips, bike wheel, cat whiskers, airplane windows, bird feather, license plate, car tail lights, and cat fur.

solve a joint segmentation-detection problem. LayoutCRF [WS06] addresses the problem of detecting and segmenting partially occluded objects using CRFs. These approaches have not been applied to highly articulated objects and they have not been designed to handle viewpoint and scale changes.

The method of Gould et al. [GGK09] integrates multi-class segmentation with object detection using a region-based approach. Work by Ladicky et al. [LSA10] proposes another CRF-based framework for estimating the class category, location, and segmentation of objects in a scene. The goal of these methods is different from ours as they find a labeling for scene contents while we try to improve the object detectors. Therefore, they have not provided results on challenging *object detection* datasets.

5.1 HOG-bundle Description and Classification

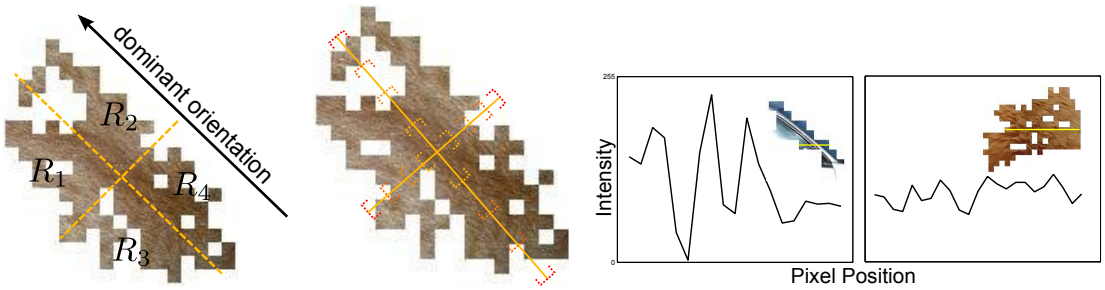
Our goal in this section is to develop a binary classifier to distinguish the HOG-bundles of a certain object category from the others. HOG-bundles are formed by unsupervised grouping of neighboring HOG cells that have similar features and dominant orientations. We propose a set of descriptors for HOG-bundles and use them as inputs to the classifiers. In addition to SIFT and color, we propose two other descriptors more specific to HOG-bundles.

5.1.1 HOG-bundle Descriptors

Gradient Descriptor: is defined based on the histograms of the gradients of pixels in four regions of a HOG-bundle. This descriptor is somewhat similar to SIFT, where the main difference is that the pixels that do not belong to the bundle are not used in building the histograms. Also, the gradient histograms are computed over irregular-shaped regions instead of rectangular patches. A gradient orientation histogram is built for each region separately, where each bin represents the sum of the magnitude of the gradients whose orientation corresponds to that bin. The regions intersect at the center of the bundle and are rotated according to the dominant orientation of the HOG-bundle (see Fig. 5.2(a)). The dominant orientation of a HOG-bundle is defined as the mean of the orientations with maximum magnitude in the constituent HOG cells.

The gradient descriptor $\mathbf{g}$ is constructed by concatenation of the vectors $\frac{1}{|R_1|}\mathbf{g}_1$ through $\frac{1}{|R_4|}\mathbf{g}_4$, where $\mathbf{g}_i$ is the histogram corresponding to region R_i and $|R_i|$ is the number of pixels that belong to region R_i. We quantize the orientations into 18 bins so this descriptor has $72 = 4 \times 18$ dimensions.

SIFT Descriptor: is defined based on the histogram of SIFT features computed over the bundles. First, a vocabulary of 4096 SIFT words corresponding to four different SIFT scales is generated (1024 words for each scale) using the standard K-means clustering. Then, for each cell of a bundle, the SIFT descriptor is computed at four scales (8×8, 12×12, 16×16, 20×20 patches). Using the Soft-binning technique [PCI08], we find a 4096 dimensional histogram for each cell. We accumulate all of the histograms corresponding to different cells by summing over the corresponding bins and normalizing them according to the number of the HOG cells in a HOG-bundle.



(a) (b) (c)

Figure 5.2: (a) Regions used for computing the gradient descriptor. The regions are rotated according to the bundle dominant orientation. (b) Line segments used for the jaggedness descriptor. The start and end of each segment are represented by the same color. (c) Intensity profiles of a car and a cat bundle computed over the yellow line. The profile is almost periodic with small changes for a cat bundle, while there are sharp peaks in the profile of a car bundle.

It should be noted that the patch for the SIFT descriptors is centered at the center of the HOG cells. We denote the SIFT descriptor by $\mathbf{s}$.

Jaggedness Descriptor: provides a rough estimate for the changes of intensity over a HOG-bundle. For example, as shown in Fig. 5.2(c), a cat fur bundle is usually composed of parallel lines, and the intensity over the bundle changes smoothly and almost periodically, while there is a sharp peak in bundles that correspond to edges of car parts.

To compute the descriptor, we consider 8 line segments over a HOG-bundle, 4 in the direction of the dominant orientation, and 4 in the perpendicular direction. All of the line segments are centered around the bundle center, and their length is $w/4$, $w/2$, $3w/4$, and w for the segments along the width of the bundle and $h/4$, $h/2$, $3h/4$, and h for the segments along the height of the bundle. Fig. 5.2(b) shows the line segments for an example bundle. If p and q are neighboring pixels along a line segment where p precedes q and both of them belong to the HOG-bundle, the feature vector for the i^{th} line segment, $\mathbf{j}_i$, is defined as: $\mathbf{j}_i = \frac{1}{L_i}\left[\sum_{p:I_p>I_q} \nabla I_{pq}, \sum_{p:I_p<I_q} \nabla I_{qp}, \sum_{p:I_p=I_q} \mathbb{1}(I_p = I_q)\right]$, where $\nabla I_{pq} = I_p - I_q$, and I_p and I_q are the intensities of pixels p and q, respectively. L_i is the number of pixels of segment i that belong to the bundle, and $\mathbb{1}(.)$ is the indicator function. The first two components are intensity differences while the third component is a normalized count. We

leave it to the classifier to find the proper weights for combining these different quantities. Since we have 8 line segments, the bundle descriptor, $\mathbf{j} = [\mathbf{j}_1, \ldots, \mathbf{j}_8]$, has 24 dimensions.

Color Descriptor: is the normalized color histogram of pixels of a bundle. We use HSV color space and compute a histogram with 16 bins for each channel. The color descriptor, $\mathbf{c}$, is defined by the concatenation of the histograms for each channel and has $48 = 3 \times 16$ dimensions. The histograms are normalized by the number of pixels in a bundle.

5.1.2 Feature Combination

One of the unary terms of our CRF formulation in Section 5.2.1 is the likelihood of a HOG-bundle being part of a certain object category. This likelihood can be computed by a classifier. Our goal in this section is to classify HOG-bundles according to the descriptors mentioned previously. For this purpose, we develop a two-stage binary SVM classifier to combine our features.

In the first stage, we learn a binary SVM classifier with RBF kernel for each descriptor separately (for the SIFT descriptor we use a linear kernel due to its high dimensionality). As the result, we learn the parameters $\{\alpha_i^d, b^d\}$ that are used in the SVM discriminant function for bundle $\mathbf{z}$: $o(\mathbf{z}^d) = \sum_{\mathbf{v}_i^d \in \mathcal{S}} \alpha_i^d y_i K(\mathbf{z}^d, \mathbf{v}_i^d) + b^d$, where $\mathbf{z}^d$ is one of the bundle descriptors (d indexes the descriptors $\mathbf{g}$, $\mathbf{s}$, $\mathbf{j}$, and $\mathbf{c}$ defined above), and $\mathbf{v}_i^d$ is a member of the support vectors set $\mathcal{S}$. The bundle label y_i is binary-valued and determines if the bundle belongs to a certain category or not. Also, the kernel is defined as $K(\mathbf{z}^{(.)}, \mathbf{v}_i^{(.)}) = \exp(-\gamma \left\| \mathbf{z}^{(.)} - \mathbf{v}_i^{(.)} \right\|^2)$ for the Gradient, Jaggedness, and Color descriptors, and a linear kernel is used for the SIFT descriptor $K(\mathbf{z}^{\mathbf{s}}, \mathbf{v}_i^{\mathbf{s}}) = (\mathbf{z}^{\mathbf{s}})^T \mathbf{v}_i^{\mathbf{s}}$.

To combine the features, we use a linear SVM classifier that uses the output of the first stage as its input. Our approach is similar to [GN09] with the difference that we use a binary linear SVM classifier in the second stage instead of their LP-B method. As the result of the first stage of classification, we obtain four real valued functions, $o(\mathbf{z}^{\mathbf{g}})$, $o(\mathbf{z}^{\mathbf{s}})$, $o(\mathbf{z}^{\mathbf{j}})$, and $o(\mathbf{z}^{\mathbf{c}})$, corresponding to the four different features we defined. The linear SVM classifier in the second stage finds the discriminant hyperplane parametrized by β_d and c: $O(\mathbf{z}) = \sum_{d \in \{\mathbf{g},\mathbf{s},\mathbf{j},\mathbf{c}\}} \beta_d o(\mathbf{z}^d) + c$. Therefore, a real value is assigned to each bundle $\mathbf{z}$ in an image. The positive training examples for these

classifiers are the bundles completely contained inside the bounding boxes of a particular category in the training images. The rest of the bundles are considered as negative.

5.2 Integration of Deformable Part Models with HOG-bundles

Our object detection framework fits into the paradigm of image labeling, where the labels correspond to object instances or background. In this section, we explain the Conditional Random Field (CRF) model that we use to integrate the information provided by the Deformable Part Models and HOG-bundles.

Our CRF model has a random variable for each image pixel, where the variables take a value from a discrete set of labels $\mathcal{L} = \{l_0, l_1, \ldots, l_M\}$. l_0 corresponds to the background and M denotes the number of objects of a *particular class* in the image ($l_1 = \text{car}_1$ and $l_2 = \text{car}_2$, for instance). We define a Gibbs distribution for the posterior pixel labeling $\mathbf{x}$ given an observed image $\mathcal{I}$: $P(\mathbf{x}|\mathcal{I}) = \frac{1}{Z}\exp\{-E(\mathbf{x})\}$, where Z is the partition function and $E(\mathbf{x})$ is the energy function. To determine the most probable assignment of pixels to objects, we compute the maximum-a-posteriori (MAP) estimate labeling, $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x}|\mathcal{I}) = \arg\min_{\mathbf{x}} E(\mathbf{x})$. The energy function is computed according to a set of unary and pairwise potential functions that we describe next.

5.2.1 Unary Terms

Superposition term (P_s)**.** This term is defined based on the detections from a DPM and measures the likelihood of presence of an object of a certain category in the image. We use a variation of the part-based detector of [CZY10] so using their notation, the discriminant function for region $\mathcal{X}$ with root part $\mathbf{p}_1$ (see [CZY10] for details) is defined as:

$$F(\mathcal{X}, \mathbf{p}_1) = \max_{\mathbf{p}} \ \mathbf{w}.\Phi(\mathcal{X}, \mathbf{p}), \tag{5.1}$$

where $\mathbf{w}$ represents the learned parameters, and $\Phi(.,.)$ is a vector of shape and appearance features defined over region $\mathcal{X}$ and parts $\mathbf{p}$. The set of detections, $\mathcal{D}$, is defined as: $\mathcal{D} = \{\mathcal{X} : F(\mathcal{X}, \mathbf{p}_1) > 0\}$.

Since a DPM is usually not able to capture large deformations, it generates multiple imperfect detections corresponding to the deformed object. The idea here is that we superimpose all of the detections to obtain a more robust estimate for the location of objects. Let x_i denote the label for the i^{th} node (pixel). The likelihood function is written as:

$$P_s(x_i \neq l_0|\mathcal{D}) = \frac{1}{C} \sum_{\substack{\mathcal{X}\in\mathcal{D} \\ \text{s.t.:} i\in\mathcal{X}}} F(\mathcal{X}, \mathbf{p}_1), \tag{5.2}$$

where by $i \in \mathcal{X}$, we mean $\mathcal{X}$ includes the pixel that corresponds to node i. Also, C is a normalizing constant that is equal to the maximum superposition value that we obtain from the images of the category of interest in the validation dataset. The definition of this term is along the lines of the voting strategies used for object detection (e.g., [FEH10]).

It should be noted that we do not prune out the detections in $\mathcal{D}$ by non-maximal suppression or any other heuristics at this stage. Pruning the overlapping detections discards the information that the superposition term relies on. Column (c) of Fig. 5.6 visualizes this term for some example images.

HOG-bundle term (P_h)**.** This term corresponds to the output of the HOG-bundle classifier (O) that we defined in Section 5.1.2. We use a logistic function to convert the real-valued predictions to probabilities. So we obtain the probability of a HOG-bundle belonging to a particular object category. This probability is uniform across a bundle i.e. the same probability is assigned to the constituent pixels of a bundle. The HOG-bundle term is defined as:

$$P_h(x_i \neq l_0|\mathbf{z}) = \max\big(P(x_i \neq l_0|\mathbf{z}_+), P(x_i \neq l_0|\mathbf{z}_-)\big), \tag{5.3}$$

where $\mathbf{z} = \{\mathbf{z}_+, \mathbf{z}_-\}$ is the set of HOG-bundles that cover the i^{th} node. Images have two sets of overlapping HOG-bundles constructed according to two different parts of the HOG feature vector. We refer to these sets as positive and negative bundles. $\mathbf{z}_+$ and $\mathbf{z}_-$ are members of the positive and negative set, respectively. If a pixel belongs to two overlapping bundles, the higher probability is assigned to that pixel. This term is visualized in column (b) of Fig. 5.6.

The Superposition and HOG-bundle terms are combined linearly: $P_{lm}(x_i) = \eta P_s(x_i|\mathcal{D}) + (1-\eta)P_h(x_i|\mathbf{z})$, where $\eta \in [0, 1]$. We refer to this linear combination as the *likelihood map* in the rest

of this section. Hence, the unary term of the energy function, $\psi(x_i)$, is given by:

$$\psi(x_i) = \begin{cases} P_{lm}(x_i) & x_i \in \{l_1, \ldots, l_M\} \\ P_b & x_i = l_0, \end{cases} \tag{5.4}$$

where P_b is an adaptive background likelihood that varies for different images.

The procedure to compute the background likelihood (P_b) is as follows. There are sharp discontinuities in the likelihood map (P_{lm}) at the object boundaries, where one side of the discontinuity belongs to an object and the other side belongs to the background. Our idea is to find the sharp discontinuities and estimate the background likelihood from the pixels around the discontinuities that potentially belong to the background. The discontinuities can be found by thresholding the magnitude of the difference of a pixel in the likelihood map from its neighbors. We compute the difference map, $|\nabla P_{lm}|$, by accumulating $|\nabla^t P_{lm}|$, $|\nabla^{tr} P_{lm}|$, $|\nabla^r P_{lm}|$, and $|\nabla^{br} P_{lm}|$ corresponding to difference with the top, top right, right, and bottom right neighbors, respectively. The sharp discontinuities happen at pixels with $|\nabla P_{lm}|$ greater than a threshold T (the values less than T usually correspond to the internal discontinuities of an object). For each pixel at a discontinuity, we find the neighboring pixel that has the minimum likelihood in the likelihood map P_{lm}. The background likelihood P_b for each image is estimated as the median of the likelihood of these neighboring pixels with the lowest likelihood.

5.2.2 Pairwise Term

We define a pairwise potential function based on the color histogram of HOG-bundles to represent the relationship between the neighboring nodes. It is more robust to define this term based on the color difference of HOG-bundles rather than the individual pixels. Since an internal edge of an object and its surrounding region are usually included in a single HOG-bundle, assigning two different labels to the two sides of internal intensity or color discontinuities typically has a large

penalty. The pairwise term $\phi(x_i, x_j, \{\mathbf{z^c}\})$ for two neighbor nodes i and j is given by:

$$\phi(x_i, x_j, \{\mathbf{z^c}\}) = \begin{cases} \exp(-\left\|\mathbf{z^c}(i) - \mathbf{z^c}(j)\right\| / \kappa) & x_i \neq x_j \\ 0 & x_i = x_j, \end{cases} \tag{5.5}$$

where $\mathbf{z^c}(i)$ is the color histogram for the bundle that node i belongs to (refer to Section 5.1.1 for the definition of this color histogram) and $\|.\|$ is the $L2$ norm. The parameter κ is set to the average of the distances between the color histogram of neighboring bundles. Note that the pixels that belong to a single HOG-bundle have the same histogram, and the changes happen only at the boundary pixels of each HOG-bundle.

Recall that there are two sets of overlapping bundles in each image. Therefore, a pairwise term is defined over each set separately, and we denote them by $\phi_+(x_i, x_j, \{\mathbf{z^c}\})$ and $\phi_-(x_i, x_j, \{\mathbf{z^c}\})$ corresponding to the positive and negative bundles, respectively. The histogram differences are shown for some example images in Fig. 5.3.

Figure 5.3: The color histogram difference of the positive and negative HOG-bundles. Darker lines represent sharper discontinuities.

5.2.3 Object Detection

Our goal is to determine which pixels belong to the object category of interest, and also to find the regions corresponding to the different instances of that category in the image. To find the most probable labels, we minimize an energy function that is computed according to the unary

and pairwise potentials we defined above. Given the random field defined over a standard 8-neighborhood grid $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$, the energy function is defined as:

$$E_s(\mathbf{x}) = -\sum_{i \in \mathcal{V}} \log \psi(x_i) + \lambda \sum_{(i,j) \in \mathcal{E}} \phi_s(x_i, x_j, \{\mathbf{z}^{\mathbf{c}}\}), \tag{5.6}$$

where $s \in \{+, -\}$ specifies whether the pairwise term corresponds to the positive bundles or negative bundles, and λ is a weighting coefficient that is estimated empirically from validation data. We could combine these two energy functions corresponding to positive and negative bundles to a single energy function but we obtain better results by decoupling these two terms. We find the MAP estimates $\mathbf{x}_+^*$ and $\mathbf{x}_-^*$ corresponding to $E_+(\mathbf{x})$ and $E_-(\mathbf{x})$, respectively, by the sequential tree-reweighted message passing (TRW-S) [Kol06]. The final labeling $\mathbf{x}^*$ is computed as follows. If $\mathbf{x}_+^*$ and $\mathbf{x}_-^*$ agree on a label for a node, that label is assigned to that node. If they disagree, we assign the background label (l_0) to the node.

5.2.4 Multiple Instance Detection

So far we have treated all of the foreground objects similarly. Now we explain how we distinguish multiple instances of the object category of interest. We use the information from the bounding boxes generated by DPM for this purpose. The idea is to assign a label to a small set of pixels in each bounding box, and then let the energy minimization find the best labeling for the other pixels in the image. We choose a small set of pixels from each box since a bounding box might include background clutter that should be removed. A fixed label is assigned to the pixels in each bounding box whose value in the likelihood map (P_{lm}) is greater than a certain fraction f of the maximum value in that box. Hence, the threshold varies for each box. We set f to 0.7 in our experiments. We denote the nodes with a fixed label by $\mathbf{x}_k$ and find the MAP estimate conditioned on these known pixels.

The procedure for assigning fixed labels is as follows. First a non-maximal suppression similar to [FGM10] is performed to prune out the overlapping detections. Then, we start from the top-scoring bounding box and assign label l_1 to the nodes inside the box that satisfy the above threshold

criteria. We continue by checking the second top-scoring detection and assign label l_2 to the nodes satisfying the criteria and so on. Since the bounding boxes might overlap, we do not change the label for a node that has already been assigned a label.

We are interested in finding the most probable labeling for the nodes with unknown label $\mathbf{x}_u$ conditioned on the known labels $\mathbf{x}_k$ (specified above): $\mathbf{x}_u^* = \arg\max_{\mathbf{x}_u} P(\mathbf{x}_u|\mathbf{x}_k)$. Since $P(\mathbf{x}_u|\mathbf{x}_k) = P(\mathbf{x}_u, \mathbf{x}_k)/P(\mathbf{x}_k)$ and $P(\mathbf{x}_k)$ is a constant and $\mathbf{x} = \mathbf{x}_k \cup \mathbf{x}_u$, we can write $P(\mathbf{x}_u|\mathbf{x}_k) \propto \exp(-E(\mathbf{x}))$ (we have dropped the positive and negative subscripts for simplicity of the notation). Hence, the energy function is similar to the previously defined energy, and the procedure for finding the optimal labeling does not change.

For bounding box-based evaluation methods, we fit a bounding box around all of the pixels with the same label and output those bounding boxes as our new detections. So there is a bounding box corresponding to each label. The score of a detection is obtained by averaging the value of the likelihood map over the pixels inside the bounding box whose label is the same as the bounding box label. We also add the score of the original detection from DPM to the computed score.

5.3 Experiments

We evaluate our method on the PASCAL VOC 2007 and 2010 datasets, which contain 20 object classes. The evaluation details are described below.

5.3.1 Evaluation of HOG-Bundle Classification

First, we evaluate the discrimination power of the HOG-bundle descriptors by learning bundle classifiers. The positive training examples are the bundles completely contained inside the bounding boxes corresponding to a particular category in `train` images of PASCAL 2007 dataset. The rest of the bundles form the negative example set. Since the number of negative examples exceeds the number of positive examples, we use a randomly sampled subset of the negative set that has the same size as the positive set for training. For testing, we use the `test` set of the dataset that

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
w/o Grad.	81.4	59.2	63.2	69.5	60.5	60.8	59.6	58.7	43.9	67.8	68.8	57.8	68.7	58.5	46.5	59.5	72.3	66.8	66.0	68.6
w/o SIFT	77.5	54.9	60.7	64.5	59.6	56.0	55.2	56.4	60.8	61.8	62.3	56.0	65.1	54.5	46.4	54.1	68.1	60.2	62.6	60.5
w/o Jagged.	82.1	60.3	64.7	70.0	60.7	61.6	60.3	59.5	44.0	68.3	69.5	57.8	68.7	59.2	46.7	59.8	72.6	66.7	67.4	69.1
w/o Color	81.5	57.6	61.0	68.2	60.3	60.9	58.5	59.2	55.3	68.8	66.1	58.0	64.1	57.9	46.9	57.3	72.5	64.2	65.3	69.8
all	82.5	60.6	64.8	70.4	61.6	62.3	60.6	59.9	44.9	69.3	69.8	58.9	69.5	59.8	47.6	60.5	73.4	67.8	68.1	69.9

Table 5.1: Classification results of the bundles of PASCAL VOC 2007 `test` dataset. The AUC of the classifier ROC curve is shown. Each row shows the results where one of the features is excluded. The shown AUCs correspond to the average result of five different random selections of the negative set during training.

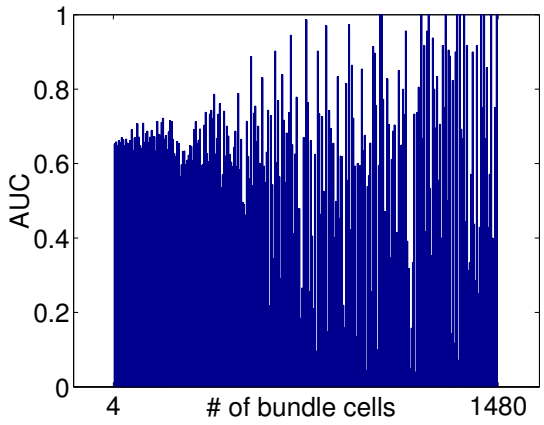


Figure 5.4: The effect of bundle size on the classification of Horse bundles.

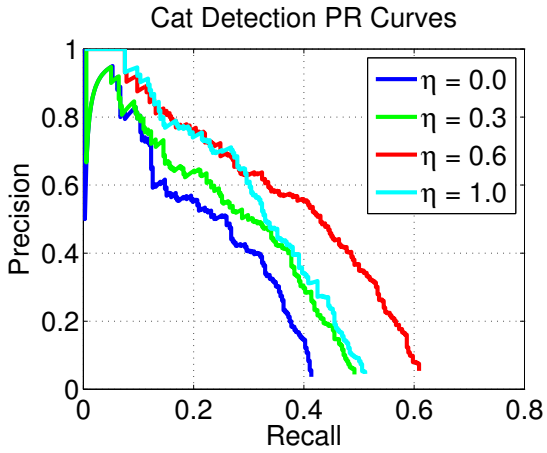


Figure 5.5: The effect of varying η on the PR curve of Cat detection.

includes around 900K bundles.

Table 5.1 shows the area under the ROC curve (AUC) of the classifiers for different combinations of features. Fig. 5.4 also shows for an example category that the AUC of the final classifier becomes larger as the bundle size increases.

5.3.2 Detection Results

To evaluate the performance of our method, we use the DPM of the second winner of PASCAL Challenge 2010 (similar to [CZY10], but excluding the shape masks) as our base object detector. As we show later, our method is independent of the base detector, and other DPMs can be used as well. We prune out most of the irrelevant detections and keep the ones whose score is greater than a certain fraction of the score of the top-scoring detection in the corresponding image. To estimate this fraction for each category, we compute the ratio of the score of the correct detection with the lowest score to the score of the top-scoring detection for all of the validation images containing objects of the category of interest. The fraction is estimated by averaging all of these ratios. On average we keep the detections whose score is above 50% of the score of the top detection in each image. The number of foreground labels M in each image (defined in the beginning of Section 5.2) is equal to the number of the remaining detections in that image.

Table 5.2 shows that our method provides significant improvement (up to 8.0%) over the base detector on PASCAL 2007 dataset, especially for the highly deformable classes such as cat, bird, dog, plant, and also for diningtable that has a highly variable appearance. We also show how the performance changes when we use the superposition potential only as our unary term. We combine the superposition and HOG-bundle terms linearly but more sophisticated methods can be used. Fig. 5.5 shows the effect of choosing different weights for the linear combination on the detection of an example category. We set η to 0.6 in all of our experiments.

We also provide comparisons to some of the state-of-the-art methods [CZY10, FGM10, SCH11, VGV09]. Unlike [FGM10] and [SCH11], we do not use contextual cues. Nevertheless, our method outperforms these methods on highly deformable object classes (colored columns in Table 5.2) in terms of mean AP. It should be noted that our performance is dependent on the underlying detector. There are several cases that the HOG-bundle classifier has a high response on the object but we cannot detect the object because of no response from the base detector.

Fig. 5.6 shows the bounding boxes generated by the base detector and our detections for some example images. For the cat image, the base detector finds only the head since the DPMs cannot

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	H. D. Avg.
Base Detector	33.5	54.8	12.3	15.6	29.3	51.4	52.8	31.2	21.8	25.4	34.9	18.1	52.9	43.6	40.2	17.5	22.3	31.9	42.7	44.2	28.3
SP only (ours)	30.5	51.7	18.3	11.1	25.4	38.7	53.9	33.8	19.8	22.9	40.1	21.1	41.7	41.6	26.5	13.1	18.9	33.9	44.5	45.4	26.3
SP + HB (ours)	34.5	55.7	**18.6**	17.9	28.4	53.1	52.9	**39.2**	22.1	26.7	**41.0**	**24.3**	55.8	47.3	38.7	**21.1**	**23.9**	34.0	47.3	46.0	**32.1**
gain	+1.0	+0.9	+6.3	+2.3	-0.9	+1.7	+0.1	+8.0	+0.3	+1.3	+6.1	+6.2	+2.9	+3.7	-1.5	+3.6	+1.6	+2.1	+4.6	+1.8	+3.8
MKL [VGV09]	37.6	47.8	15.3	15.3	21.9	50.7	50.6	30.0	17.3	**33.0**	22.5	21.5	51.2	45.5	23.3	12.4	**23.9**	28.5	45.3	**48.5**	25.9
UoCTTI [FGM10]	31.2	**61.5**	11.9	17.4	27.0	49.1	**59.6**	23.1	23.0	26.3	24.9	12.9	60.1	**51.0**	**43.2**	13.4	18.8	36.2	49.1	43.0	26.1
Active Masks [CZY10]	34.8	54.4	15.5	14.6	24.4	50.9	54.0	33.5	20.6	22.8	34.4	24.1	55.6	47.3	34.9	18.1	20.2	30.3	41.3	43.3	28.8
NUS-Context [SCH11]	**38.6**	58.7	18.0	**18.7**	**31.8**	**53.6**	56.0	30.6	**23.5**	31.1	36.6	20.9	**62.6**	47.9	41.2	18.8	23.5	**41.8**	**53.6**	45.3	31.5

Table 5.2: Detection performance on PASCAL VOC 2007 `test` set. The evaluation metric is Average Precision (%). The colored columns represent the highly deformable classes. "SP only" refers to the case that we use only the Superposition potential as our unary term. "SP+HB" refers to the case that we use the combination of Superposition and HOG-bundle potentials. The row labeled by "gain" shows the improvement of "SP+HB" over the "Base Detector". The last column is the mean AP for the highly deformable classes.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
Baseline [FGM10]	17.3	35.2	7.1	8.2	21.4	49.3	38.9	22.9	9.9	9.2	**12.7**	10.5	16.0	26.8	31.9	14.0	10.8	13.4	25.2	27.1	20.4
Ours	**21.7**	**47.3**	**8.9**	**15.0**	**30.9**	**59.9**	**40.6**	**32.4**	**10.5**	**18.2**	10.9	**19.01**	**16.2**	**38.2**	**32.1**	**15.7**	**17.2**	**31.5**	**32.4**	**41.8**	**27.0**

Table 5.3: Detection performance on PASCAL VOC 2010. The evaluation metric is Average Precision obtained using the instance-based pixel-wise benchmark of [YR11].

reliably model the deformation of cats. Our method extends the detection to include the whole body. The dog image shows a case that the background clutter is included in the detection since the base detector does not have a reliable model for dog deformations either. Our method shrinks the detection window to exclude the background clutter. Also, the base detector's confidence is low for the chair case (it can be observed in the superposition image). We improve the confidence by incorporating the information from the HOG-bundles.

The reason that our method does not provide significant improvement for categories such as bicycle or car is that the underlying DPMs are rich enough for those objects, and the HOG-bundles

do not provide any additional detection cues. We also observe a high performance decrease for the Person category. We believe it is mainly due to the poor performance of the HOG-bundle classifier for this category (refer to Table 5.1). Our bundle descriptors are reliable for bundles that correspond to lips, eyes, and skin but they cannot describe the variation in clothing for example. Therefore, a better result is obtained by using shape cues alone.

Instance-based Pixel-wise Scoring: PASCAL evaluation method is not suitable for our method because it is based on 50% intersection over union overlap between the bounding boxes, while our method is able to generate object masks. The segmentation evaluation methods are not suitable for our method either because they evaluate only the correctness of the object category and ignore object instance information. Therefore, we design a new evaluation method such that pixel-wise mismatch between the generated object masks and the groundtruth is penalized. We sort all of the generated detections for all of the images by their score. Note that these detections correspond to a particular object category. We start from the top-scoring detection and find the groundtruth object segment that has the highest overlap with it. Intersection over union between the segments is used as the measure of overlap. The matched pixels are true positives, and unmatched pixels of the detection contribute to the false positives. A groundtruth segment can only match to one of the detections so we remove the groundtruth object from the available segments set when we find a match for it. For some of the detections, we can not find any match. All of the pixels of these detections are considered as false positives. To compute the precision-recall curve, we vary a threshold over the detection scores, where only the detections whose score is above the threshold contribute to the true and false positive computations. Note that all of the pixels of a detection have the same score.

We use the pixel-wise instance labels for PASCAL VOC detection dataset provided by [HAB11]. Table 5.3 shows the average precisions obtained by using the evaluation method of on PASCAL 2010 detection subset of the dataset, which includes about 10,000 images. On average, our method provides 6.6 gain over the baseline in terms of AP. To show that our method is applicable to other DPMs, we use [FGM10] as our base for this experiment. It should be noted that our learning is performed on PASCAL 2007 `trainval` set. In all of our experiments, we use only a single

scale of HOG features (8×8 cells) to construct HOG-bundles. Performing the experiments using multiple scales of HOGs could probably produce even better results.

5.4 Conclusion

We propose a novel approach to augment the detections of Deformable Part Models with patches of irregular shape to better capture the object masks. We develop a CRF framework to find the most probable assignment of pixels to each object instance in images. Our method provides significant improvement over the base DPMs on bounding box-based and mask-based benchmarks.

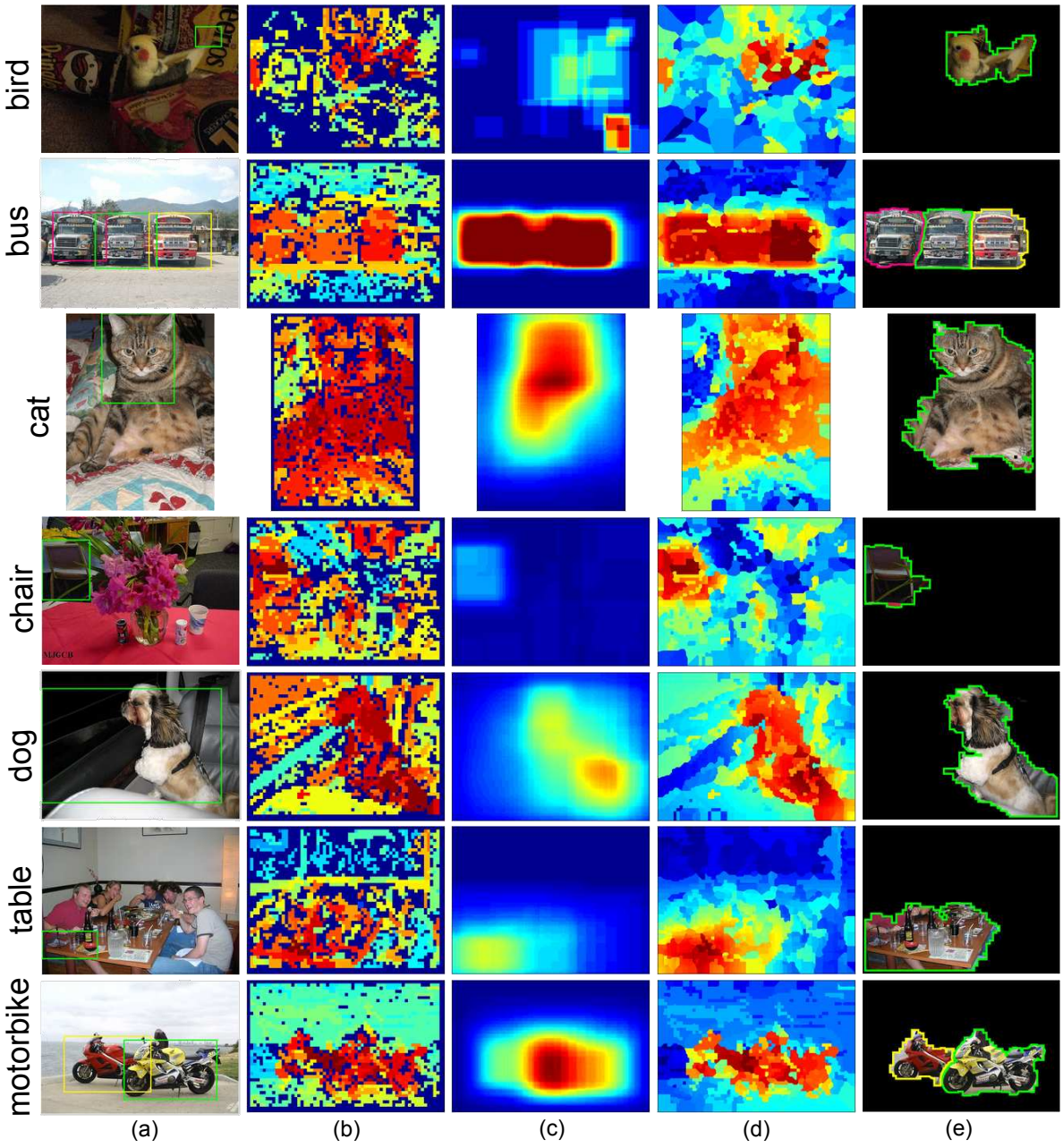


Figure 5.6: (a) Original detections of the base detector, (b) Heat map of the HOG-bundle term (P_h), (c) Heat map of the Superposition term (P_s), (d) Smoothed heat map of the likelihood map (P_{lm}), (e) Final output of our method. The boundary for each object instance is illustrated by a different color (corresponding to the color of the detections in column (a)).

CHAPTER 6

Combining Segmentation with Detection

In the previous chapter, we proposed a method to better capture deformations of objects. The method was based on HOG-bundles, which typically correspond to tiny parts of an object. In contrast, in this chapter, our aim is to use segments that correspond to the whole body of an object.

Over the past few years, we have witnessed a push towards holistic approaches that try to solve multiple recognition tasks jointly [STF05, GBW10, LRK10, HGS08, YFU12]. This is important as information from multiple sources should facilitate scene understanding as a whole. For example, knowing which objects are present in the scene should simplify segmentation and detection tasks. Similarly, if we can detect where an object is, segmentation should be easier as only figure-ground segmentation is necessary. Existing approaches typically take the output of a detector and refine the regions inside the boxes to produce image segmentations [LKR09, BBM11, AHG12, FGM10]. An alternative approach is to use the candidate detections produced by state-of-the-art detectors as additional features for segmentation. This simple approach has proven very successful [GBW10, KK11] in standard benchmarks.

In contrast, we are interested in exploiting semantic segmentation in order to improve object detection. While bottom-up segmentation has been often believed to be inferior to top-down object detectors due to its frequent over- and under- segmentation, recent approaches [CLS12, AHG12] have shown impressive results in difficult datasets such as PASCAL VOC challenge. Here, we take advantage of region-based segmentation approaches [CCB12], which compute a set of candidate object regions by bottom-up clustering, and produce a segmentation by ranking those regions using class specific rankers. Our goal is to make use of these candidate object segments to bias sliding window object detectors to agree with these regions. Importantly, unlike the aforementioned holis-

tic approaches, we reason about all possible object bounding boxes (not just candidates) to not limit the expressiveness of our model.

Deformable part-based models (DPM) [FGM10] and its variants [AL12, ZCY10, CZY10], are arguably the leading technique to object detection [1]. However, so far, there has not been many attempts to incorporate segmentation into DPMs. In this chapter we propose a novel deformable part-based model, which exploits region-based segmentation by allowing every detection hypothesis to select a segment (including void) from a small pool of segment candidates. Towards this goal, we derive simple features, which can capture the essential information encoded in the segments. Our detector scores each box in the image using both the traditional HOG filters as well as the set of novel segmentation features. Our model "blends" between the detector and the segmentation models by boosting object hypotheses on the segments. Furthermore, it can recover from segmentation mistakes by exploiting a powerful appearance model. Importantly, as given the segments we can compute our features very efficiently, our approach has the same computational complexity as the original DPM [FGM10].

We demonstrate the effectiveness of our approach in PASCAL VOC 2010, and show that when employing only a root filter our approach outperforms Dalal & Triggs detector [DT05] by 13% AP, and when employing parts, we outperform the original DPM [FGM10] by 8%. Furthermore, we outperform the previous state-of-the-art on VOC2010 by 4%.

In the remainder of this chapter, we first review related work and then introduce our novel deformable part-based model, which we call **segDPM**. We then show our experimental evaluation.

Deformable part-based model [FGM10] and its variants have been proven to be very successful in difficult object detection benchmarks such as PASCAL VOC challenge. Several approaches have tried to augment the level of supervision in these models. Azizpour et al. [AL12] use part annotations to help clustering different poses as well as to model occlusions. Hierarchical versions of these models have also been proposed [ZCY10], where each part is composed of a set of subparts. The relative rigidity of DPMs has been alleviated in [CZY10] by leveraging a dictionary of shape masks. This allows a better treatment of variable object shape. Desai et al. [DRF09]

[1] Poselets [BMB10] can be shown to be very similar in spirit to DPMs

proposed a structure prediction approach to perform non-maxima suppression in DPMs which exploits pairwise relationships between multi-class candidates. The tree structure of DPMs allows for fast inference but can suffer from problems such as double counting observations. To mitigate this, [PVG11] consider lateral connections between high resolution parts.

In the past few years, a wide variety of segmentation algorithms that employ object detectors as top-down cues have been proposed. This is typically done in the form of unary features for an MRF [KK11], or as candidate bounding boxes for holistic MRFs [YFU12, LSA10]. Complex features based on shape masks were exploited in [YFU12] to parse the scene holistically in terms of the objects present in the scene, their spatial location as well as semantic segmentation. In [PVJ11], heads of cats and dogs are detected with a DPM, and segmentation is performed using a GrabCut-type method. By combining top-down shape information from DPM parts and bottom-up color and boundary cues, [YR11] tackle segmentation and detection task simultaneously and provide shape and depth ordering for the detected objects. Dai et al. [DH12] exploit a DPM to find a rough location for the object of interest and refine the detected bounding box according to occlusion boundaries and color information. Also, in the previous chapter, we found silhouettes for objects by extending or refining DPM boxes corresponding to a reliably detectable part of an object.

DPMs provide object-specific cues, which can be exploited to learn object segmentations [BYV11]. In [MO12], masks for detected objects are found by employing a group of segments corresponding to the foreground region. Other object detectors have been used in the literature to help segmenting object regions. For instance, while [BMB10] finds segmentations for people by aligning the masks obtained for each Poselet [BMB10], [MYP11] integrates low level segmentation cues with Poselets in a soft manner.

There are a few attempts to use segments/regions to improve object detection. Gu et al. [GLA09] apply hough transform for a set of regions to cast votes on the location of the object. More recently, [SZS10] learn object shape model from a set of contours and use the learned model of contours for detection. In contrast, we propose a novel deformable-part based model, which allows each detection hypothesis to select candidate segments. Simple features express the fact

that the detections should agree with the segments. Importantly, these features can be computed very efficiently, and thus our approach has the same computational complexity as DPM [FGM10].

6.1 Semantic Segmentation for Object Detection

We are interested in utilizing semantic segmentation to help object detection. In particular, we take advantage of region-based segmentation approaches, which compute candidate object regions by bottom-up clustering, and rank those regions to estimate a score for each class. Towards this goal we frame detection as an inference problem, where each detection hypothesis can select a segment from a pool of candidates (those returned from the segmentation as well as void). We derive simple features, which can be computed very efficiently while capturing most information encoded in the segments. In the remainder of the section, we first discuss our novel DPM formulation. We then define our new segment-based features and discuss learning and inference in our model.

6.1.1 A Segmentation-Aware DPM

Following [FGM10], let p_0 be a random variable encoding the location and scale of a bounding box in an image pyramid as well as the mixture component id. As shown in [FGM10] a mixture model is necessary in order to cope with variability in appearance as well as the different aspect ratios of the training examples. Let $\{p_i\}_{i=1,\cdots,P}$ be a set of parts which encode bounding boxes at double the resolution of the root. Denote with h the index over the set of candidate segments returned by the segmentation algorithm. We frame the detection problem as inference in a Markov Random Field (MRF), where each root filter hypothesis can select a segment from a pool of candidates. We thus write the score of a configuration as

$$\begin{aligned} E(\mathbf{p}, h) &= \sum_{i=0}^{P} w_i^T \cdot \phi(x, p_i) + \sum_{i=1}^{P} w_{i,def}^T \cdot \phi(x, p_0, p_i) + \\ &\quad + w_{seg}^T \phi(x, h, p_0) \end{aligned} \tag{6.1}$$

where $h \in \{0, 1, \cdots, H(x)\}$, with $H(x)$ the total number of segments for this class in image x. Note that $h = 0$ implies that no segment is selected. We will use $S(h)$ to denote the segment

that h indexes. As in [FGM10], we employ a HOG pyramid to compute $\phi(x, p_0)$, and use double resolution to compute the part features $\phi(x, p_i)$. The features $\phi(x, h, p_0)$ link segmentation and detection. We define features at the level of the root, but our formulation can be easily extended to include features at the part level.

6.1.2 Segmentation Features

Given a set of candidate segments, we would like to encode features linking segmentation and detection while remaining computationally efficient. We would also like to be robust to over- and under- object segmentations, as well as false positive or missing segments. Towards this goal, we derive simple features which encourage the selected segment to agree with the object detection hypothesis. Most of our features employ integral images which makes them extremely efficient, as this computation can be done in constant time. We now describe the features in more details.

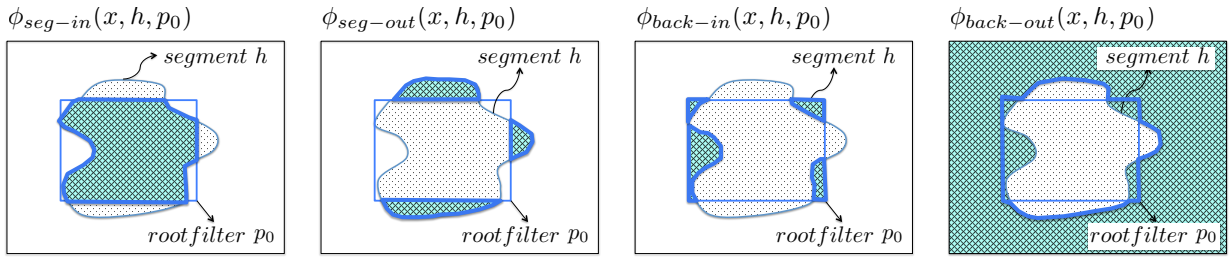


Figure 6.1: The box-segment features: ϕ_{seg-in} and $\phi_{seg-out}$, encourage the box to contain as many segment pixels as possible. This pair of features alone could result in box hypotheses that "overshoot" the segment. The purpose of the second pair of features, $\phi_{back-in}$ and $\phi_{back-out}$, is the opposite: it tries to minimize the number of background pixels inside the box and maximize its number outside. In synchrony these features would try to tightly place a box around the segment.

Segment-In: Given a segment $S(h)$, our first feature counts the percentage of pixels in $S(h)$ that fall inside the bounding box defined by p_0. Thus

$$\phi_{seg-in}(x, h, p_0) = \frac{1}{|S(h)|} \sum_{p \in B(p_0)} \mathbb{1}\{p \in S(h)\}$$

where $|S(h)|$ is the size of the segment indexed by h, and $B(p_0)$ is the set of pixels contained in the bounding box defined by p_0. This feature encourages the bounding box to contain the segment.

Segment-Out: Our second feature counts the percentage of segment pixels that are outside the bounding box,

$$\phi_{seg-out}(x, h, p_0) = \frac{1}{|S(h)|} \sum_{p \notin B(p_0)} \mathbb{1}\{p \in S(h)\}$$

This feature discourages boxes that do not contain all segment pixels.

Background-In: We additionally compute a feature counting the amount of background inside the bounding box as follows

$$\phi_{back-in}(x, h, p_0) = \frac{1}{N - |S(h)|} \sum_{p \in B(p_0)} \mathbb{1}\{p \notin S(h)\}$$

with N the size of the image. This feature captures the statistics of how often the segments leak outside the true bounding box vs how often they are too small.

Background-Out: This feature counts the amount of background outside the bounding box

$$\phi_{back-out}(x, h, p_0) = \frac{1}{N - |S(h)|} \sum_{p \notin B(p_0)} \mathbb{1}\{p \notin S(h)\}$$

It tries to discourage bounding boxes that are too big and do not tightly fit the segments.

Overlap: This feature penalizes bounding boxes which do not overlap well with the segment. In particular, it computes the intersection over union between the candidate bounding box defined by p_0 and the tighter bounding box around the segment $S(h)$. It is defined as follows

$$\phi_{overlap}(x, h, p_0) = \frac{B(p_0) \cap B(S(h))}{B(p_0) \cup B(S(h))} - \lambda$$

with $B(S(h))$ the tighter bounding box around $S(h)$, $B(p_0)$ the bounding box defined by p_0, and λ a constant, which is the intersection over union level that defines a true positive. We employ in practice $\lambda = 0.7$.

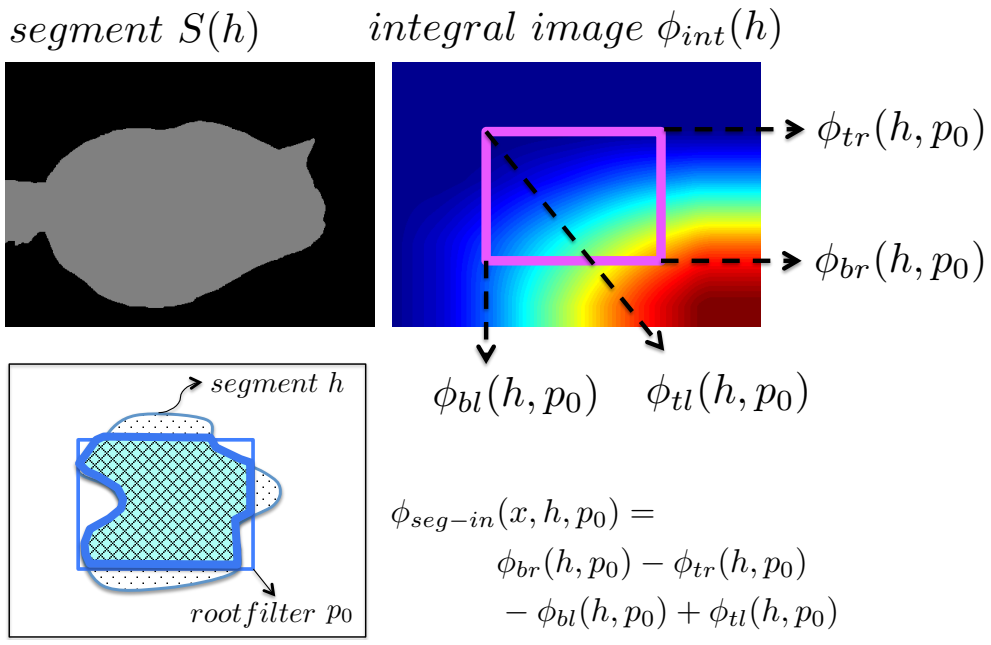


Figure 6.2: Segment feature computation via integral image.

Background bias: The value of all of the above features is 0 when $h = 0$. We incorporate an additional feature to learn the bias for the background segment ($h = 0$). This puts the scores of the HOG filters and the segmentation potentials into a common referential. We thus simply define

$$\phi_{bias}(x, h, p_0) = \begin{cases} 1 & \text{if } h = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 6.1 depicts our features computed for a specific bounding box p_0 and segment $S(h)$. Note that the first two features, ϕ_{seg-in} and $\phi_{seg-out}$, encourage the box to contain as many segment pixels as possible. This pair of features alone could result in box hypotheses that "overshoot" the segment. The purpose of the second pair of features, $\phi_{back-in}$ and $\phi_{back-out}$, is the opposite: it tries to minimize the number of background pixels inside the box and maximize its number outside. In synchrony these features would try to tightly place a box around the segment. The overlap feature has a similar purpose, but helps us better tune the model to the VOC IOU evaluation setting.

6.1.3 Efficient Computation

Given the segments, all of our proposed features can be computed very efficiently. Note that the features have to be computed for each segment h, but this is not a problem as there are typically only a few segments per image. We start our discussion with the first four features, which can be computed in constant time using a single integral image per segment. This is both computationally and memory efficient. Let $\phi_{int}(h)$ be the integral image for segment h, which, at each point (u, v), counts the % of pixels that belong to this segment and are contained inside the subimage defined by the domain $[0, u] \times [0, v]$. This is illustrated in Fig. 6.2. Given the integral image $\phi_{int}(h)$ for the h-segment, we compute the features as follows

$$\begin{aligned}
\phi_{seg-in}(x, h, p_0) &= \phi_{br}(h, p_0) - \phi_{tr}(h, p_0) \\
&\quad -\phi_{bl}(h, p_0) + \phi_{tl}(h, p_0) \\
\phi_{seg-out}(x, h, p_0) &= |S(h)| - \phi_{seg-in}(x, h, p_0) \\
\phi_{back-in}(x, h, p_0) &= |B(p_0)| - \phi_{seg-in}(x, h, p_0) \\
\phi_{back-out}(x, h, p_0) &= (N - |S(h)|) - \phi_{back-in}(x, h, p_0)
\end{aligned}$$

where as shown in Fig. 6.2, $(\phi_{tl}, \phi_{tr}, \phi_{bl}, \phi_{br})$ indexes the integral image of segment $S(h)$ at the four corners, i.e., top-left, top-right, bottom-left, bottom-right, of the bounding box defined by p_0.

The overlap feature between a hypothesis p_0 and a segment $S(h)$ can also be computed very efficiently. First, we compute the intersection as:

$$\begin{aligned}
&B(p_0) \cap B(S(h)) = \\
&\max\left[0, \min(x_{0,\text{right}}, x_{S(h),\text{right}}) - \max(x_{0,\text{left}}, x_{S(h),\text{left}})\right] \cdot \\
&\max\left[0, \min(y_{0,\text{bottom}}, y_{S(h),\text{bottom}}) - \max(y_{0,\text{top}}, y_{S(h),\text{top}})\right]
\end{aligned}$$

Note that the overlap will be non-zero only when each of the terms is larger than 0. Given that the segment bounding box $B(S(h))$ is fixed and the width and height of p_0 at a particular level of the pyramid are fixed as well, we can quickly compute the bounds of where in the image the feature needs to be computed (i.e., when the feature is different than 0). The denominator, $B(p_0) \cup B(S(h))$, can then be simply computed as the sum of the box areas minus the overlap.

6.1.4 Inference

Inference in our model can be done by solving the following optimization problem:

$$\max_{p_0} \Big(\sum_{i=0}^{P} w_i^T \cdot \phi(x, p_i) + \sum_{i=1}^{P} \max_{p_i} (w_{i,def}^T \cdot \phi(x, p_0, p_i)) + + \max_{h} (w_{seg}^T \cdot \phi(x, h, p_0)) \Big)$$

Note that this can be done efficiently using dynamic programming as the structure of the graphical model forms a tree. The algorithm works as follows: First, we compute $\max_h w_{seg}^T \phi(x, h, p_0)$ as well as $\max_{p_i}(w_{i,def}^T \cdot \phi(x, p_0, p_i))$ for each root filter hypothesis p_0. We then compute the score as the sum of the HOG and segment score for each mixture component at the root level. Finally, we compute the maximum over the mixture components to get the score of an object hypothesis.

6.1.5 Learning

We learn a different weight for each feature using a latent structured-SVM [GFM11]. Allowing different weights for the different segmentation features is important in order to learn how likely is for each class to have segments that undershoot or overshoot the detection bounding box. We employ as loss the intersection over the union of the root filters. As in DPM [FGM10], we initialize the model by first training only the root filters, followed by training a root mixture model. Finally we add the parts and perform several additional iterations of stochastic gradient descent [FGM10].

Note that we expect the weights for $\phi_{seg-in}(x, h, p_0)$, $\phi_{back-out}(x, h, p_0)$ and $\phi_{overlap}(x, h, p_0)$ to be positive, as we would like to maximize the overlap, the amount of foreground inside the bounding box and background outside the bounding box. Similarly, the weights for $\phi_{seg-out}(x, h, p_0)$ and $\phi_{back-in}(x, h, p_0)$ are expected to be negative as we would like to minimize the amount of background inside the bounding box as well as the amount of foreground segment outside. In practice, as the object's shape can be far from rectangular, and the segments are noisy, the sign of the weights can vary to best capture the statistics of the data.

6.1.6 Implementation Details

We use CPMC [CCB12] to get the candidate segments. In particular, for most experiments we use the final segmentation output of [CCB12]. For each class, we find all connected components in the segmentation output, and remove those that do not exceed 1500 pixels. Unless otherwise noted, we do not use the score of the segments. On average, this gives us one segment per image. We also provide one experiment where we used more segments (5 on average per image), which we describe in Section 6.2.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
VOC 2010 val, no parts																					
Dalal [DT05]	29.1	36.9	2.9	3.4	15.6	47.1	27.1	11.4	9.8	5.8	6.0	5.0	24.8	28.4	27.5	2.2	18.4	9.2	27.4	23.2	18.1
segDPM (no parts)	**52.4**	**43.1**	**20.9**	**15.7**	**18.6**	**55.8**	**33.2**	**43.9**	**10.7**	**22.0**	**14.8**	**31.1**	**40.9**	**45.1**	**33.6**	**11.1**	**27.3**	**22.0**	**42.5**	**31.7**	**30.8**
VOC 2010 val, with parts																					
CPMC (no score) [CCB12]	49.9	15.5	18.5	14.7	7.4	35.0	19.9	41.4	3.9	16.2	8.5	24.4	26.0	32.1	18.9	5.7	15.3	14.1	29.8	18.7	20.8
CPMC (score) [CCB12]	53.3	19.5	22.8	15.7	8.1	42.7	22.1	**51.3**	4.3	18.9	10.5	28.1	30.5	38.3	20.9	6.0	19.2	18.6	35.4	21.1	24.4
DPM [FGM10]	46.3	49.5	4.8	6.4	22.6	53.5	38.7	24.8	14.2	10.5	10.9	12.9	36.4	38.7	**42.6**	3.6	26.9	22.7	34.2	31.2	26.6
segDPM (parts)	**55.7**	**50**	**23.3**	**16.0**	**28.5**	**57.4**	**43.2**	49.3	**14.3**	**23.5**	**17.7**	**32.4**	**42.6**	**44.9**	42.1	**11.9**	**32.5**	**25.5**	**43.9**	**39.7**	**34.7**

Table 6.1: AP performance (in %) on **VOC 2010 val** for our detector with parts, the DPM [FGM10], and the CPMC-based detector [CCB12].

6.2 Experimental Evaluation

We first evaluate our detection performance on `val` subset of PASCAL VOC 2010 detection dataset, and compare it to the baselines. We train all methods, including the baselines on the `train` subset. We use the standard PASCAL criterion for detection (50% IOU overlap) and report average precision (AP) as the measure of evaluation.

As baselines we use the Dalal&Triggs detector [DT05] (which for fairness we compare to our detector when only using the root filters), the DPM [FGM10], as well as CPMC [CCB12] when used as a detector. To compute the latter, we find all the connected components in the final segmentation output of CPMC [CCB12], and place the tightest bounding box around each

component. To compute the score of the box we utilize the CPMC ranking scores for the segments.

The comparison with [DT05] and our approach (segDPM) without parts is shown in the top Table 6.1, while the bottom table compares CPMC-based detector, DPM and our approach with parts. We significantly outperform the baselines: Dalal & Triggs detector by 13% and the CPMC baseline by 10%. Our model also achieves a significant boost of 8% AP over the DPM, which is a well established and difficult baseline to beat. Importantly, we outperform DPM in 19 out of 20 classes. The main power of our approach is that it blends between DPM (appearance) and segmentation (CPMC). When there is no segment, the method just scores a regular DPM. When there is a segment, our approach is encouraged to tightly fit a box around it. However, in cases of under- or over- segmentation, the appearance part of our model can still correctly position the box. Note that our results well demonstrate the effectiveness of using blended detection and segmentation models for object detection.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
VOC 2010 test, no post-processing																					
DPM no postproc. [FGM10]	47.2	**50.8**	8.6	12.2	**32.2**	48.9	44.4	28.1	13.6	22.7	11.3	17.4	40.4	47.7	**44.4**	7.6	30	17.3	38.5	34.3	29.9
segDPM no postproc.	**56.4**	48.0	**24.3**	**21.8**	31.3	**51.3**	**47.3**	**48.2**	**16.1**	**29.4**	**19.0**	**37.5**	**44.1**	**51.5**	**44.4**	**12.6**	**32.1**	**28.8**	**48.9**	**39.1**	**36.6**
VOC 2010 test, with post-processing																					
segDPM+rescore+classif	**61.4**	53.4	**25.6**	**25.2**	**35.5**	51.7	**50.6**	**50.8**	19.3	**33.8**	26.8	**40.4**	48.3	54.4	47.1	**14.8**	38.7	**35.0**	**52.8**	**43.1**	**40.4**
segDPM+rescore	**58.7**	51.4	**25.3**	**24.1**	33.8	52.5	**49.2**	**48.8**	11.7	30.4	21.6	37.7	46.0	53.1	46.0	13.1	35.7	29.4	**52.5**	41.8	**38.1**
NLPR_HOGLBP [YZH10]	53.3	**55.3**	19.2	21.0	30.0	54.4	46.7	41.2	**20.0**	31.5	20.7	30.3	48.6	55.3	46.5	10.2	34.4	26.5	50.3	40.3	36.8
MITUCLA_HIERARCHY [ZCY10]	54.2	48.5	15.7	19.2	29.2	**55.5**	43.5	41.7	16.9	28.5	26.7	30.9	48.3	55.0	41.7	9.7	35.8	30.8	47.2	40.8	36.0
NUS_HOGLBP_CTX [CSH12]	49.1	52.4	17.8	12.0	30.6	53.5	32.8	37.3	17.7	30.6	27.7	29.5	**51.9**	**56.3**	44.2	9.6	14.8	27.9	49.5	38.4	34.2
van de Sande et al. [SUG11]	58.2	41.9	19.2	14.0	14.3	44.8	36.7	48.8	12.9	28.1	**28.7**	39.4	44.1	52.5	25.8	14.1	**38.8**	34.2	43.1	42.6	34.1
UOCTTI_LSVM_MDPM [FGM10]	52.4	54.3	13.0	15.6	35.1	54.2	49.1	31.8	15.5	26.2	13.5	21.5	45.4	51.6	**47.5**	9.1	35.1	19.4	46.6	38	33.7
Gu et al. [GAL12]	53.7	42.9	18.1	16.5	23.5	48.1	42.1	45.4	6.7	23.4	27.7	35.2	40.7	49.0	32.0	11.6	34.6	28.7	43.3	39.2	33.1
UVA_DETMONKEY [SUG11]	56.7	39.8	16.8	12.2	13.8	44.9	36.9	47.7	12.1	26.9	26.5	37.2	42.1	51.9	25.7	12.1	37.8	33.0	41.5	41.7	32.9
UVA_GROUPLOC [SUG11]	58.4	39.6	18	13.3	11.1	46.4	37.8	43.9	10.3	27.5	20.8	36	39.4	48.5	22.9	13	36.8	30.5	41.2	41.9	31.9
BONN_FGT_SEGM [CLS12]	52.7	33.7	13.2	11.0	14.2	43.1	31.9	35.6	5.7	25.4	14.4	20.6	38.1	41.7	25.0	5.8	26.3	18.1	37.6	28.1	26.1

Table 6.2: AP performance (in %) on **VOC 2010 test** for our detector with parts and the DPM [FGM10], without post processing (top table), and comparison with existing methods (only top 11 shown), with post-processing (table below).

Fig. 6.4 depicts examples illustrating the performance of our approach. Note that our approach

is able to both retrieve detections where there is no segment as well as position the bounding box correctly where there is segment evidence.

We evaluate our approach on VOC 2010 `test` in Table 6.2. Here, we trained CPMC [CCB12], as well as our model on VOC `trainval`. We compare segDPM with DPM without the post-processing steps, i.e., bounding box prediction and context-rescoring, in the top of Table 6.2. In the bottom of Table 6.2 we compare our full approach with existing top scoring approaches. For the full approach, we show results when typical context-rescoring approach is used (same as in DPM), which we refer to as segDPM+rescore. We also show results when we rescored the detections by using the classification scores for each class, kindly provided to us by [CSH12]. The classification (presence/absence of class in an image) accuracy measured by mean AP on VOC2010 is 76.2%. We refer to this approach with segDPM+rescore+classif. We outperform the competitors by 3.6%, and achieve the best result in 13 out of 20 classes.

We also experimented with using more segments, on the VOC 2010 `train` / `val` split. In particular, among 150 segments per image returned by [CLS12], we selected a top-ranking subset for each class, so that there was an average of 5 segments per image. The results are reported in Table 6.3. We compare it to CPMC when using the same set of segments. One can see that with more segments our approach improves by 1.5%. As such, it outperforms DPM by 10%.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
VOC 2012 val, more segments																					
CPMC (1 seg) [CCB12]	53.3	19.5	22.8	15.7	8.1	42.7	22.1	51.3	4.3	18.9	10.5	28.1	30.5	38.3	20.9	6.0	19.2	18.6	35.4	21.1	24.4
CPMC (5 seg) [CCB12]	59.8	27.6	27.1	19.6	12.7	53.1	31.2	56.6	8.2	25.6	17.5	34.8	39.8	42.3	25.9	10.3	29.8	26.6	46.7	33.4	31.4
segDPM (1 seg)	55.7	50.0	23.3	16.0	28.5	57.4	43.2	49.3	14.3	23.5	17.7	32.4	42.6	44.9	42.1	11.9	32.5	25.5	43.9	39.7	34.7
segDPM (5 seg)	56.1	49.0	22.9	18.2	34.0	58.9	42.9	49.8	15.4	25.0	22.7	32.3	46.2	45.6	39.2	13.6	33.3	30.6	46.7	41.5	**36.2**

Table 6.3: AP performance (in %) on **VOC 2010 val** for our detector when using more segments.

6.3 Conclusion

In this chapter, we propose a novel deformable part-based model, which exploits region-based segmentation by allowing every detection hypothesis to select a segment (including void) from a

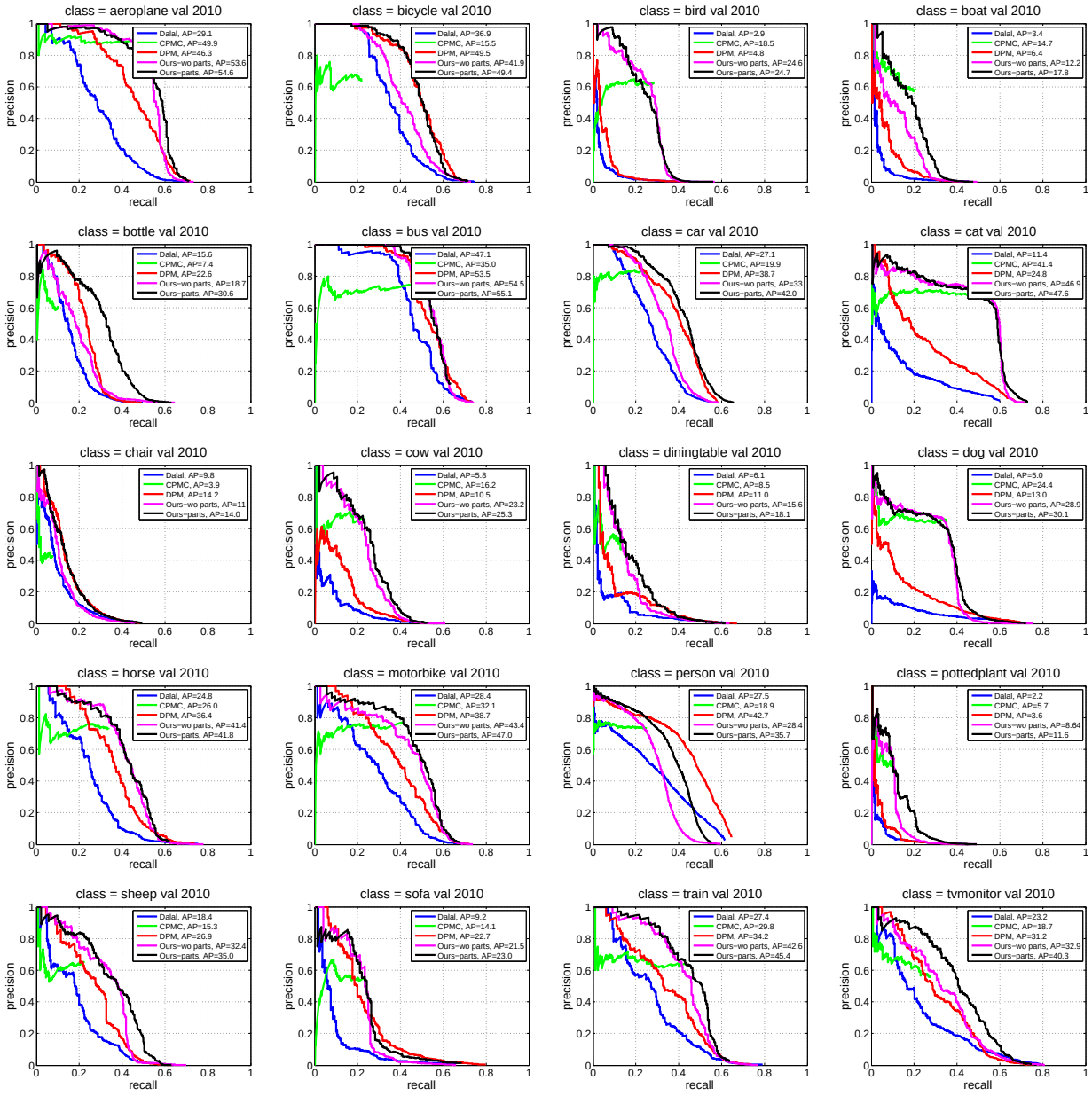
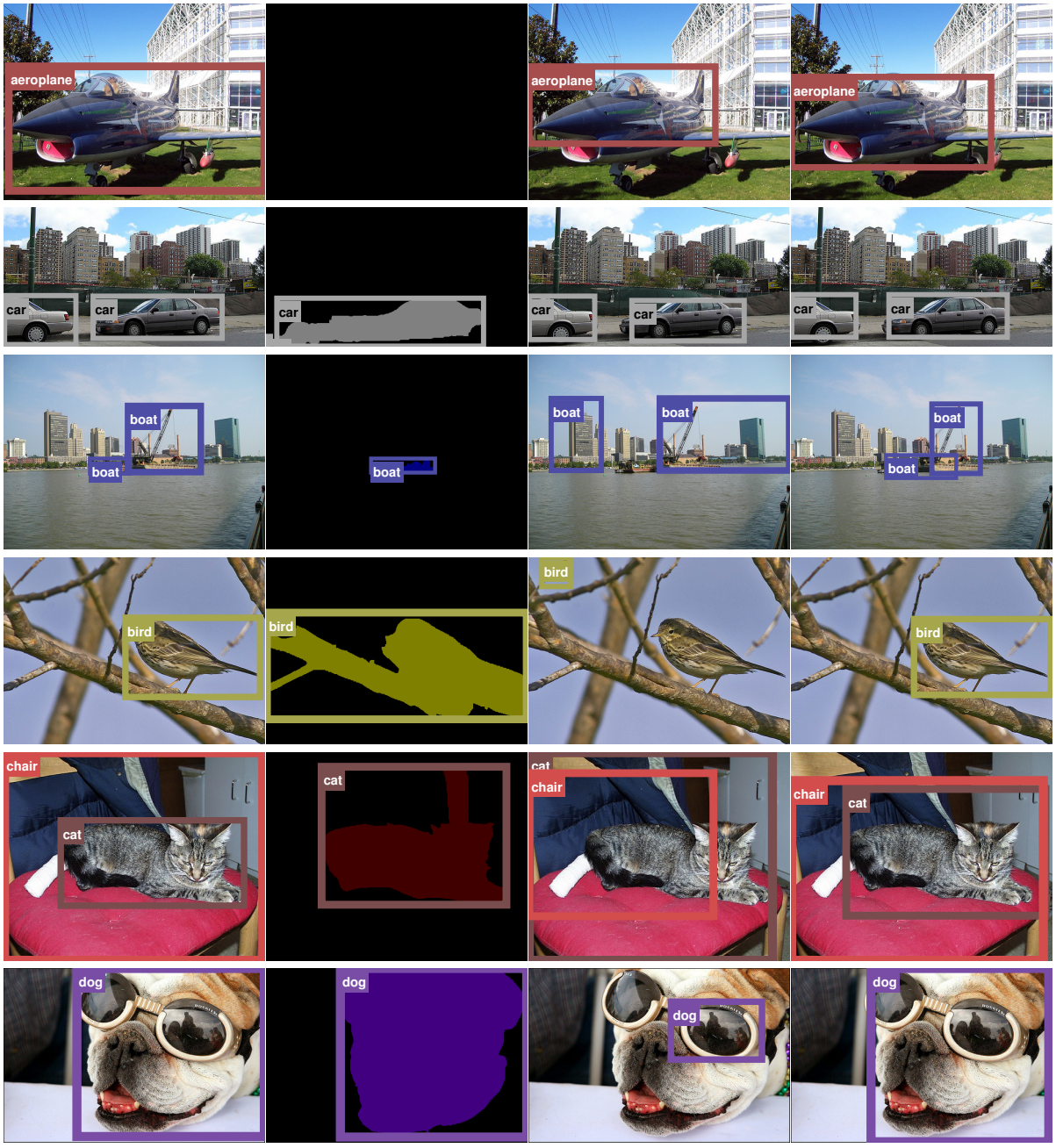


Figure 6.3: Precision-recall curves for all methods on the validation set of PASCAL VOC 2010. Note that our approach significantly outperforms all baselines.

pool of segment candidates. We derive simple yet very efficient features, which can capture the essential information encoded in the segments. Our detector scores each box in the image using both the HOG filters as in original DPM, as well as a set of novel segmentation features.

This way, our model "blends" between the detector and the segmentation model, by boosting



(a) GT (b) CPMC (c) DPM (d) segDPM

Figure 6.4: For each method, we show top k detections for each class, where k is the number of boxes for that class in GT. For example, for an image with a chair and a cat GT box, we show the top scoring box for chair and the top scoring box for cat.

object hypotheses on the segments, while recovering from making mistakes by exploiting a powerful appearance model. We demonstrated the effectiveness of our approach in PASCAL VOC 2010, and show that when employing only a root filter our approach outperforms Dalal & Triggs detector [DT05] by 13% AP and when employing parts, we outperform the original DPM [FGM10] by 8%. We believe that this is just the beginning of a new and exciting direction. We expect a new generation of object detectors which are able to exploit semantic segmentation yet to come.

CHAPTER 7

Richer Description with Semantic Parts

In this chapter, we go beyond bounding box or silhouettes and localize semantics parts for objects. Obtaining richer descriptions for objects is important as other tasks such as activity recognition or pose estimation rely on these types of representation.

Despite decades of research, dealing with occlusions, deformations and pose variations remains amongst the fundamental challenges in object detection. Most current approaches try to detect the full object and all its parts, making the estimation problem particularly difficult, as the full object might not be visible or it may be difficult to detect certain (or all) parts due to occlusion and low resolution (in the case of small objects).

Here we take an alternative approach and advocate trying to *detect as much as we can*. Towards this goal, we propose to employ a collection of components designed to treat different types of occlusion, deformation, and size. In particular, we employ a (i) full body component, which handles the cases where detection of semantic parts is difficult, e.g., small scale objects, where there are no strong appearance cues for parts, (ii) part-part component capturing the cases that the detection of full body is difficult, e.g., highly deformed objects, and (iii) body-part component, which enforces geometric consistencies between the full body and a 'subset' of semantic parts to handle partial occlusion.

To cope with the complexity of inference, hypotheses are generated by each semantic part or full body models and combined to generate proposals for the components of the model. A global scoring function is then learned and used to select the best component for an object instance. The hypotheses for full body are generated by deformable part-based models [FGM10] and also by the detector we described in the previous chapter. Similarly, we use [FGM10] to generate semantic

part hypotheses. The advantage of our approach over these models is that we search over different structures. As shown in our experiments this is very important for very deformable objects, as a fix structure is too rigid to represent all pose and occlusion variations.

We demonstrate the effectiveness of our approach on the animal categories of PASCAL VOC 2010 dataset, and show that our approach achieves 6.5 AP improvement over DPM [FGM10] and 3.0 AP improvement over the results mentioned in the previous chapter. We note that our method also outputs the positions of semantic parts, while the baseline codes only output a bounding box or shape mask for the full body. We require object part annotations to train the semantic part detectors. For this purpose, we annotated semantic parts for all of the animal categories of PASCAL VOC 2010 (see the experiments section for more details).

Our work can be considered as a descendant of Pictorial Structure [FH05] or the work of Kumar and Hebert [KH04] since we use a classifier for parts to generate hypotheses. However, our model is more powerful than these approaches as it allows a variable structure with or without the full body model.

There is a considerable body of work on part-based object detectors which encode parts using latent variables [CH06, ZLH08, DBB08, FGM10, SRS10, KDH10]. Typically these methods formulate object detection within a discriminative framework and so learn parts that are most effective for discriminating the object from the background clutter of the images. Hence many of these methods are not suitable for recovering the semantic parts of the object. On the other hand, using supervised parts has been explored to recover semantic parts, for example, [YR11, SS11, AL12] to mention a few.

Most work on human pose estimation in recent years has relied on anatomical parts [SNH10, STT10, TF10, YR11]. These methods can be divided into two major categories: approaches that learn the models for parts and body simultaneously and the ones that learn the models for parts independently from the body model. Our approach is close to the latter category as we learn full body and semantic part models separately and then combine them together to boost object detection performance.

There are fewer works dealing with the semantic parts of other object classes [AL12, SS11, BM09, BPB11]. Strong supervision is used in these approaches mainly for configuration clustering and occlusion reasoning.

Park et al. [PRF10] propose a multi-resolution model to better detect small objects, where parts are ignored at the small scales. In contrast, we do not explicitly incorporate size into the model and let the model choose if a body-only model better describes an object instance or a model with parts. Additionally, unlike [PRF10], we are able to localize semantic parts. Our model is also able to handle partial occlusions when a semantic part is missing.

Girshick et al. [GFM11] also propose a multi-component model for different patterns of occlusion. The difference between our method and theirs is that we consider occlusion of semantic parts, while they model occlusion for latent parts of the model. Gu et al. [GAL12] also consider a multi-component model to capture different configurations of objects.

7.1 Model

The overall idea of our method is to generate a set of *proposals* for the components based on a set of full body and semantic part *hypotheses*, and then predict how well a proposal fits the objects of interest in the image.

Each proposal belongs to only one of the four *components* of an object model (shown in Figure 7.1), and our goal is to predict the score of the proposal based on learned weights for the features of that particular component. We define the quality of a proposal as overlap of full body or semantic parts with groundtruth (details are discussed later in Section 7.1.2). For training, we have access to groundtruth full body and semantic part annotations, so we can compute the quality for each proposal in the training images. During inference, we obtain a score for each proposal in a test image, which represents its quality.

To generate full body or semantic part hypotheses we use an object detector (e.g., [FGM10]) that generates multiple hypotheses in an image. The proposals are basically all possible combinations of these hypotheses that conform to a component of the model. Since there are a limited

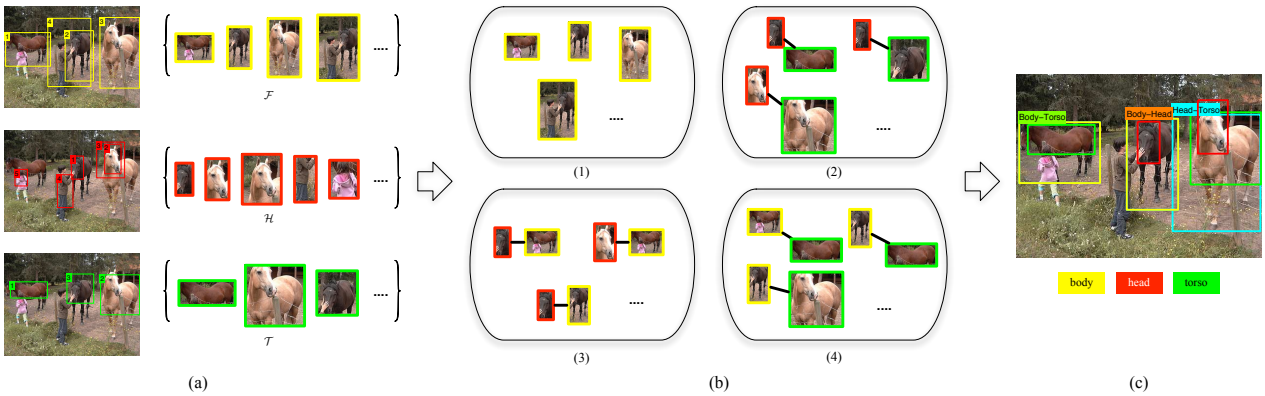


Figure 7.1: (a) The hypotheses for different semantic parts and the full body are shown. These hypotheses are generated by a baseline detector. The number in each box shows the rank given by the hypothesis score. (b) The proposals corresponding to the four components of the model are shown: (1) Body-only (2) Head-Torso (3) Body-Head (4) Body-Torso. The proposals for each component are made by combining the hypotheses shown in (a). (c) The final detection of our method.

number (typically 15-20) of hypotheses per image, there are not many combinations to consider.

The first *component* of the model represents only the full body (Figure 7.1 (b-1)). The idea is that, in some cases, it is hard to find the semantic parts of an object, and a model for the full body should work better than a model with semantic parts. This case typically happens for small scale objects, where it is hard to detect the object part because of lack of enough appearance information. The second *component* includes only the head and torso of the animal (and no full body). Often it is hard to find a full body model that fits a highly deformed object. However, in some situations, the object parts can be detected strongly. This component defines the relationship between head and torso and does not rely on a model for the full body (Figure 7.1 (b-2)). The third and fourth *components* represent the case that one of the semantic parts is missing because of occlusion or truncation (Figure 7.1 (b-3) and (b-4)).

We learn a classifier that outputs a score for the proposals based on the spatial and scale consistency terms between the hypotheses used to generate a proposal and also the hypotheses score. These consistency terms and the quality are computed for the proposals in the training images.

During inference, we apply the learned classifier to the component proposals in the test images to obtain a score for them. For the final evaluation of the method, we rank the proposals based on this score.

To be more specific, our goal is to learn a linear classifier model of the following form:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i (max(0, 1 - q_i\mathbf{w}^T\Phi_i))^2, \tag{7.1}$$

where Φ_i is a feature vector representing the i-th proposal, and q_i is the label for the proposal, which is computed according to its quality. If the quality is greater than a certain threshold, q_i will be 1. Otherwise it will be -1. Below, we describe how we compute the feature vector Φ and the quality.

7.1.1 Component Features

Let $\mathcal{F} = \{f_1, \ldots, f_p\}$ be the set of hypotheses obtained by the full body detector in an image and $\mathcal{H} = \{h_1, \ldots, h_n\}$ and $\mathcal{T} = \{t_1, \ldots, t_m\}$ be the head and torso hypotheses. Therefore, we have p proposals for the body-only component, $n \times m$ proposals for the head-torso component, and $p \times m$ and $p \times n$ for body-torso and body-head components, respectively.

Any hypothesis z has a score $\phi(z)$ which is given by the base classifier (detector). Each hypothesis also has a geometric attribute, i.e. the bounding box $B(z) = [x_1(z), y_1(z), x_2(z), y_2(z)]$ (also output by the detector). From this we can compute other geometric properties such as the size of each dimension $s_x(z)$ and $s_y(z)$ and the center coordinates $c_x(z)$ and $c_y(z)$.

The feature vector for a proposal of type $\tau \in \{1, \ldots, 4\}$ has the following form, where τ

indexes the components:

$$\Phi_{\cdot}^{\tau}(f_i, h_j, t_l) = \begin{bmatrix} \boldsymbol{\phi}(f_i) \\ \boldsymbol{\phi}(h_j) \\ \boldsymbol{\phi}(t_l) \\ \boldsymbol{\psi}(f_i, h_j) \\ \boldsymbol{\psi}(f_i, t_l) \\ \boldsymbol{\psi}(h_j, t_l) \\ \mathbf{b}_{\tau} \end{bmatrix}. \tag{7.2}$$

The first three elements are the hypothesis scores we defined above. Some of the components/proposals do not include certain types of hypotheses, for example, a body-torso component does not include 'head', so the corresponding element in the feature vector, $\boldsymbol{\phi}(h_j)$ will be zero. Note that we normalize the scores with a sigmoid function of form $f(x) = \frac{1}{1+exp(-1.5x)}$.

The pairwise relationship between parts of a component is denoted by $\boldsymbol{\psi}$ in the feature vector and it consists of spatial $\boldsymbol{\psi}_{sp}$ and scale $\boldsymbol{\psi}_{sc}$ terms. Basically, we have: $\boldsymbol{\psi}(.,.) = \begin{bmatrix} \boldsymbol{\psi}_{sp}(.,.) & \boldsymbol{\psi}_{sc}(.,.) \end{bmatrix}^T$. We use similar spatial relationships used by [FGM10]: $\boldsymbol{\psi}_{sp}(z_i, z_j) = [dx, dy, dx^2, dy^2]$. The difference is that we normalize the distances by the sum of the size of the involved hypotheses. The scale term for two hypotheses z_i and z_j is defined by $\boldsymbol{\psi}_{sc}(z_i, z_j) = \left[ds, ds_x, ds_y, ds^2, ds_x^2, ds_y^2\right]$, where:

$$ds = \frac{s_x(z_i) \times s_y(z_i)}{s_x(z_j) \times s_y(z_j)},\ ds_x = \frac{s_x(z_i)}{s_x(z_j)},\ ds_y = \frac{s_y(z_i)}{s_y(z_j)} \tag{7.3}$$

We refer the reader to the beginning of the section for the definition of s.

Since there is no pairwise term for the body-only component, the $\boldsymbol{\psi}$ terms are zero for proposals that belong to that component. Similarly, a pairwise term is zero if one of its corresponding semantic parts or full body (as applicable) does not appear in the component. Also, we use the same sigmoid function as above to normalize the values for the pairwise terms.

The last element of the feature vector, $\mathbf{b}_{\tau}$, is a bias term that has four dimensions corresponding to the four components. The i-th element of $\mathbf{b}_{\tau}$ is defined as $\mathbf{b}_{\tau}(i) = \mathbb{1}(i = \tau)$. Therefore, the element corresponding to the proposal's component is 1, and the rest of elements are 0.

7.1.2 Proposal quality

We now describe the quality measure and explain how we compute label q from the quality. Each proposal consists of full body and/or semantic part hypotheses. To obtain the quality of each proposal, we compute the overlap O of its constituent hypotheses with the corresponding groundtruth objects or parts. For instance, for a proposal corresponding to the body-torso component, we compute O_t and O_f the overlaps between its torso and full body hypotheses and the groundtruth torso and full body. We define quality measure as $(O_t + O_f)/2$ in this case. Computing the quality for other components of the model is similarly defined as the average overlap of the constituent hypotheses and ground truth. We use intersection-over-union between the bounding box of full body/part hypothesis and the bounding box of the groundtruth full body/part as the measure of overlap. The proposals whose quality is greater than 0.5 will be our positive examples ($q = 1$). The rest of the proposals form the negative set ($q = -1$).

Next, we describe how we learn $\mathbf{w}$ in Equation 7.1 and how we perform inference and find the final detections for objects.

7.2 Learning & Inference

To estimate the model parameter $\mathbf{w}$, we learn a linear SVM classifier. We adopt the SVM implementation of LIBLINEAR [FCH08]. First, we train detectors for full body and semantic parts (i.e. head and torso) separately. Then, we perform inference on the training set using the learned full body and semantic part models to generate a set of hypotheses for the full body and semantic parts. We make proposals by combining different hypotheses and compute the feature vector Φ_i for each of them. The quality of each proposal is computed according to the groundtruth annotations of full body and semantic parts.

For inference, we repeat the same procedure to generate hypotheses and make proposals, but the goal is to predict the score for each proposal. We output a bounding box for each proposal according to the following procedure. For proposals that correspond to body-only, body-head,

body-torso components, we use the bounding box of their body hypothesis. There is no full body hypothesis associated to a head-torso proposal. Therefore, we predict the bounding box based on the head and torso hypotheses. The learned classifier (Equation 7.1) produces an output for each test proposal. We use that output as the score of the proposals in the final evaluation.

Bounding Box prediction for Head-Torso component: For a head-torso proposal, we use the configuration of its head and torso hypotheses h_i and t_j to predict a bounding box for the full body. For this purpose, we learn a mapping from the bounding boxes of head and torso to the upper-left (x_1, y_1) and lower-right (x_2, y_2) corners of the body bounding box. More specifically, the object bounding box prediction is performed based on the eight dimensional vector $g(z) = [B(h_i), B(t_j)]$. We ensure $c_x(h_i) < c_x(t_j)$ for the hypotheses centers by flipping the image horizontally when necessary. We use the annotated head, torso, and body bounding boxes in training data to learn a linear function for predicting $[x_1, y_1, x_2, y_2]$ from $g(z)$. This is done via least-squares regression, independently for each category.

Part-based Non-Maximal suppression: Using the inference procedure described above, a single full body, head or torso hypothesis can belong to multiple proposals. After performing inference, we employ a greedy procedure to prevent these double associations.

There is a score assigned to each detection (proposal). We sort the detections by their score and start from the highest scoring detection and remove the ones whose full body, head or torso hypotheses are shared with any previously selected detection. Note that there is no body associated to the head-torso proposals during this pruning step. We predict the body bounding box for the head-torso proposals using the above procedure. Then, we will follow the commonly known non-maximal suppression with 50% intersection over union criteria to eliminiate overlapping bounding boxes.

7.3 Experimental Evaluation

In this section, we explain the details of our experiments and show that our method obtains a significant gain over the state-of-the-art on the animal categories of PASCAL dataset. Before we

Figure 7.2: Example annotations of the dataset. The dataset provides masks for each semantic part of the object, but we use the bounding box around the mask for training and evaluation.

explain details about the experiments, we discuss the dataset we provide.

Dataset: We use the PASCAL VOC 2010 dataset for training and testing our models. Our focus is on the six animal categories because those are hard to detect due to their highly flexible body structure. For training, we have supplemented PASCAL dataset by providing labeling for the masks/bounding boxes of object parts. We note that this is a different annotation than that used in [AL12, SS11]. Currently, we do not use the masks for the parts and only use the bounding box around the part masks. Figure 7.2 shows the annotations for some example instances.

For all of the experiments of this chapter, we use `trainval` subset of PASCAL VOC 2010 detection for training and the `test` subset for testing and evaluation.

To generate full body hypotheses we use two alternative detectors and report the results for both cases: 1) DPM [FGM10], which is commonly used as the baseline by the community. We use the latest implementation of DPM (`voc-release5`). 2) The state-of-the-art detector explained in the previous chapter (segDPM) to show that this method provides improvement over the state-of-the-art. Since segDPM is not designed for detecting parts, we only use DPM for generating part hypotheses.

We form the proposals by combining the hypotheses we obtain in the training images as described in Section 7.1. Then, we compute the the quality for each proposal according to the groundtruth full body and semantic part annotations. We learn the parameters of the model and use them to perform inference on proposals we obtain for the test images.

7.3.1 Comparison with other methods

We first provide a comparison with Poselet [BM09], [AL12], [PVJ11], and [GAL12], which are also considered as strongly supervised methods. For this experiment, we use our detector described in the previous as the baseline. As shown in Table 7.1, our method provides 3 AP improvement over the results mentioned in the previous chapter, which is considered a significant improvement on PASCAL dataset. Although the part hypotheses have been generated by a less powerful detector, they help our method to improve the state-of-the-art result on average. It should be noted that we use the hypotheses from segDPM that have been rescored by context. Our approach does not have any assumption about the baseline method, and any of the above detectors can be used as the baseline instead.

7.3.2 Evaluating the effect of each component

We then evaluate each component of the model separately and check how each component performs in absence of the other components. For example, to evaluate head-torso component, we only use the head-torso proposals during training and testing and ignore the rest. Note that the body-only component performs exactly the same as the baseline so we do not report the results for that case.

The result for this experiment is shown in Table 7.2. DPM is used as the baseline. As shown in the table, the maximum average gain is obtained when we use all of the components together. Some of our detection results with the full model are shown in Figure 7.4. The head-torso model alone performs worse than the baseline method. The main reason is that the head-torso component requires the proposals to have both head and torso present, which is not always valid for the

	Bird	Cat	Cow	Dog	Horse	Sheep	mAP
segDPM	**25.4**	48.8	30.4	37.6	45.9	35.9	37.3
Our method	**25.4**	**52.4**	**34.4**	**42.0**	**51.0**	**36.6**	**40.3**
Poselets [BM09]	8.5	22.2	20.6	18.5	48.2	28.0	24.3
DPM [FGM10]	11.0	23.6	23.2	20.5	42.5	29.0	25.0
Sup-DPM [AL12]	11.3	27.2	25.8	23.7	46.1	28.0	27.0
M-Comp [GAL12]	18.1	45.4	23.4	35.2	40.7	34.6	32.9
DisPM [PVJ11]	-	45.3	-	36.8	-	-	-

Table 7.1: Average precision for detection of PASCAL VOC 2010 animals. Our method provides a significant improvement over segDPM (our baseline in this experiment). We have also shown the results for other strongly supervised methods [AL12], [BM09], [GAL12], and [PVJ11], and also for DPM [FGM10].

PASCAL dataset that includes several occluded and truncated instances.

7.3.3 Two variants of our approach

We can also formulate the problem as regression. Instead of thresholding the quality and solving a binary classification problem, we can regress directly to the quality. Regressing to a quality measure is similar in spirit to [CLS12]. However, the task and methodology is quite different here. The result is shown in Table 7.3. The result of classification when we use minimum overlap instead of average overlap for computing the quality is also shown in the table (strict positives). The performance degrades when we use these alternative methods.

7.3.4 Size based evaluation

In this part of experiments, we show that the body-only component is more suitable for extra-small objects. We divide the instances of one category into multiple size classes. We follow the

	Bird	Cat	Cow	Dog	Horse	Sheep	mAP
DPM [FGM10]	11.0	23.6	23.2	20.5	42.5	29.0	25.0
Head-Torso	7.2	**38.3**	20.1	26.9	36.3	16.0	24.1
Body-Head	8.9	33.2	23.6	28.8	39. 1	23.7	26.2
Body-Torso	13.7	28.2	22.5	22.6	42.8	27.2	26.2
Full model	**14.8**	34.6	**30.3**	**30.9**	**49.0**	**29.2**	**31.5**
Gain	3.8	11.0	7.1	10.4	6.5	0.2	6.5

Table 7.2: Average precision for detection of animals of PASCAL VOC 2010. Both full body and part models are trained using DPM. We show the evaluated for each component of the model.

	Our method	Regression	Strict positives
mAP	31.5	30.2	30.7

Table 7.3: Comparison of different variants of our method.

same division as [HCD12], where each instance is assigned to a size category, depending on its percentile size: extra-small (XS: bottom 10%); small (S: next 20%); medium (M: next 40%); large (L: next 20%); extra-large (XL: next 10%). Then, we compute what percentage of each size class has been assigned to the body-only component. The baseline for this experiment is DPM.

As shown in Table 7.4, the body-only component has the highest rate for the instances of the extra-small class and it describes fewer instances of the other size classes. This follows our argument that typically it is hard to detect head and torso for tiny objects, and the body component without parts seems more desirable for the small instances. This is also observed by [PRF10] for the task of pedestrian detection.

	XS	S	M	L	XL
Bird	47.6%	13.9%	5.8%	4.6%	4.6%
Cat	26.4%	5.5%	10.3%	6.8%	12.3%
Cow	61.5%	19.2%	9.6%	7.7%	18.5%
Dog	35.5%	8.5%	6.1%	3.2%	3.2%
Horse	52.3%	12.5%	6.8%	5.7%	11.1%
Sheep	37.8%	18.9%	8.0%	2.7%	7.9%

Table 7.4: The percentage of instances assigned to the body-only component. We divide the instances of a category into 5 size classes and show the ratio of instances of each class that are assigned to the body-only component. The body-only component is more effective for the XS class.

7.3.5 Part-based evaluation

Since our method is able to localize parts, the common evaluation method, which is based on the overlap of the *object* bounding box with the groundtruth is not ideal for us. Therefore, we implement a stricter criteria for evaluation: a proposal counts as a true positive if all of its constituent hypotheses are localized correctly as well. For instance, a body-head proposal might be correct in terms of object bounding box, but we count it as a false positive if the head is not localized correctly. By correct localization, we mean that head and torso hypotheses should have at least 40% intersection over union overlap with groundtruth parts (we use a more relaxed criteria since sometimes it is difficult to annotate an exact bounding box for a part). Table 7.5 shows the result of our method with the new evaluation method. We use our own annotation for the test set in this experiment.

7.3.6 Part localization

Our method is able to localize semantic parts for a subset of detections (detections for the body-only component do not have parts). So far, we have shown the results for object detection. Now,

	Bird	Cat	Cow	Dog	Horse	Sheep	mAP
Full	11.6	28.7	25.1	24.4	38.4	26.9	25.9
Head-Torso	3.8	19.9	11.2	7.2	20.0	9.1	11.9
Body-Head	6.6	26.7	20.7	22.6	25.2	10.6	18.7
Body-Torso	11.0	13.4	21.0	7.8	34.4	24.6	18.7

Table 7.5: Part-based evaluation result on PASCAL VOC 2010.

we evaluate the precision for semantic part localization. A part detection is considered as true positive if it has more than 40% overlap with ground truth part annotation. Otherwise, we consider it as a false positive. For this evaluation, we consider only the proposals that have more than 50% overlap with ground truth and include the part of interest. We show the curves for precision of part vs. the recall of object in Figure 7.3. Our part detection precision is fairly good even in the high recall regions of the curve.

7.4 Conclusion

Our aim is to provide richer descriptions for objects in terms of semantic parts. Therefore, we propose a model for object detection that consists of multiple components: *part-part* component which is desirable for cases that the object is highly deformed, but its semantic parts can be detected reliably; *body-part* component to represent different patterns of occlusion for semantic parts; *body-only* component to detect objects whose semantic parts cannot be detected reliably (e.g., small scale objects).

We tested the approach on the animal categories of PASCAL VOC 2010 and obtained significant improvement over DPM and also the state-of-the-art detector (segDPM of the previous chapter). Our approach is able to localize semantic parts (head and torso) if the component that is selected to describe the object includes parts. Additionally, we provide a new dataset with annotated object object parts that enables us to evaluate part detection performance as well.

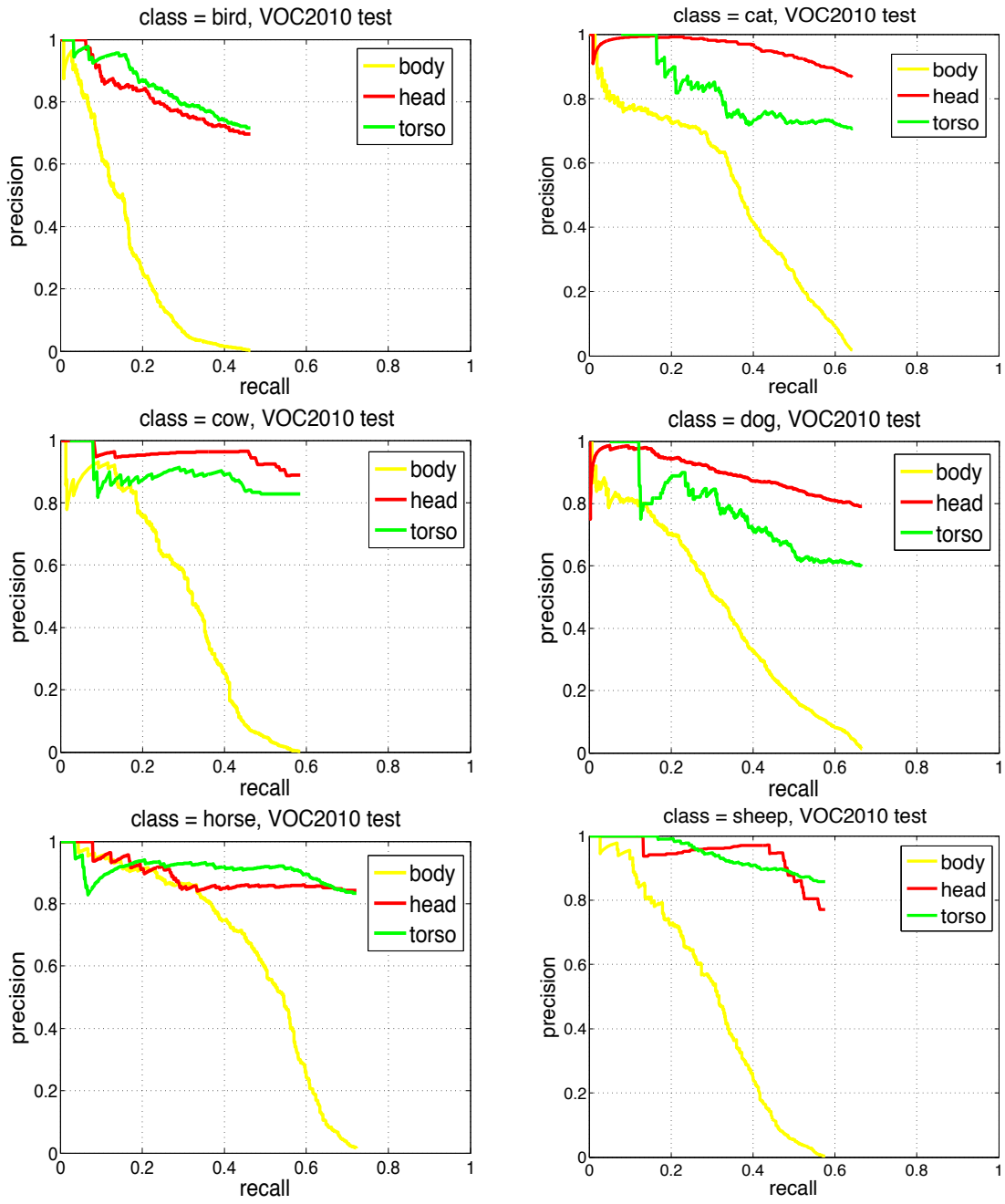


Figure 7.3: Precision-recall curves for semantic part detection. Precision of part is shown with respect to the recall of object.

Figure 7.4: The final result of our method. We show the component of the model that has been used to detect the object. Also, we visualize the location of parts if the component includes parts. The red and green boxes correspond to head and torso, respectively. The full body boxes are shown in yellow, and the cyan boxes correspond to the predicted full body bounding box for the head-torso component.

CHAPTER 8

The Role of Context for Category Level Object Detection

In this chapter, we analyze the effect of context in object detection and show that context is quite effective in cases that there are not enough appearance cues to detect an object.

Humans perceive the visual world effortlessly. We look at a complex and cluttered scene and know that the tiny object on the table is a fork and not the tail of an elephant. We know that the object hanging on the wall is more likely to be a picture or even a moose head than a car, and that a highly deformable entity stretching on the sofa is quite more likely to be a cat than a tiger. Context is a statistical property of the world we live in and is critical in helping us solve perceptual inference tasks faster and more accurately.

Cognition-based studies have proved the effect of context in various perceptual tasks such as object detection, semantic segmentation and classification. The seminal work of Biederman et al. [BMR82] and Hock et al. [HGW74] showed that contextual information such as the probability of object arrangements, relative size to other objects, scene type, and location are important cues for humans to detect objects. Furthermore, it is known that humans require a longer time to detect out of context objects. In a recent study, Parikh et al. [PZC11] showed that context is quite effective for humans to detect low-resolution (and typically small) objects in images. For object segmentation, Torralba [Tor09] showed that at lower image resolutions where only coarse scene information can be perceived, humans perform surprisingly well in delineating the most salient objects in the scene.

Context emerges from statistical properties of the world. The community is in need of large-scale, accurately and thoughtfully annotated datasets that would enable reliable estimation and exploration of this phenomena. In recent years, there has been significant effort to collect large scale datasets, among which the PASCAL VOC challenge has had probably the most impact on the

vision community to date. The competition started in 2005 with four object classes, and finished in 2012 with 20 object classes, 11,530 training images containing 27,450 ROI annotated objects and 6,929 segmentations. Additional 20,000 segmentation instance labels have been provided by [HAB11], helping to push forward the performance of segmentation approaches.

In this work, we label every pixel of the train/val detection challenge, which consists of 10103 images with 623 contextual classes. We analyze the dataset statistically, showing strong correlations and anti-correlations between certain pairs of object-context classes. We also show that there exist size-dependent correlations, which could be exploited in future models to improved detection of difficult, tiny objects. We also propose a novel deformable part-based model, which exploits both local context around each candidate detection as well as global context at the level of the scene. We show that the model significantly helps in detecting tiny objects in the dataset, but is less important when objects are bigger, and thus more easily detectable by appearance information alone.

A number of approaches have employed contextual information in order to improve object detection [CFB04, PRF10, HEH05, TMF10, HK08]. Torralba et al. [TMF10] used global scene context to improve detection. Park et al. [PRF10] use contextual information in the form of ground plane estimation to improve the detection of pedestrians that are small in size. Hoeim et al. [HEH05] used geometric context in the form of 3D surface orientations in order to improve object detection. Divvala et al. [DHH09] exploited contextual cues such as object location, spatial support and geographic information to rescore boxes produced by a base detector. Heitz and Koller [HK08] model the contextual relationship between regions found in an unsupervised manner and objects that are detected using a discriminative approach. A context-driven search is proposed by Alexe et al. [AHT12] to focus on limited areas in the image to find the objects of interest. Context is exploited in the form of statistical relations between the appearance of a window and its location relative to the object. Torralba et al. [TMF03] penalize the presence of objects in irrelevant scenes. Choi et al. [CLT10] combine a spatial and co-occurrence prior with local detector outputs and global image features to detect objects. Lee and Grauman [LG10] use the layout of familiar objects as context to find regions corresponding to unfamiliar objects. For both detection and seg-

mentation, it has been shown that representing a region larger than an object itself leads to better performance [SWR09, GBW10, CLS12].

Holistic models that reason about the scene as a whole typically build strong contextual models to improve performance over tasks in isolation. [LRK10, YFU12, GBW10] propose CRF models that reason jointly about object detection, image labeling and scene classification. Rabinovich et al. [RVG07] impose the contextual consistency of inferred segment labels through a CRF model. In [LKS10], improved performance is shown for scene classification when using responses of a bank of object detectors rather than raw image features. Auto-context [Tu08] shows long-range contextual information is important for image labeling.

In recent years, there has been significant efforts to collect densely labeled datasets. MSRC [SWR09] was one of the first datasets with pixel-wise image labels. It contained 592 images and 21 semantic classes. Camvid [BSF08] contains 708 images of street scenes with 11 semantic classes. Recently, Liu et al. [LYT11] released the SIFT-flow dataset that contains 2688 images and 33 semantic labels, which are dominated by "stuff". SUN2012 [XHE10], a smaller subset of LabelMe, consists of 16873 images and 3819 object classes, most with only few training examples. Silberman et al. [SKH12] released an RGB-D dataset of indoor scenes containing 1449 images along with depth information and 894 object labels. The PASCAL VOC challenge has 11,530 training images containing 27,450 ROI annotated objects and 6,929 segmentations pertaining to 20 object classes. In this work, we enriched this effort, by labeling the PASCAL VOC with pixel-wise accurate segmentation in terms of 623 object and stuff classes. We believe that this will allow researches to explore contextual models in difficult scenarios, pushing the performance further.

8.1 Context Labels in the Wild

Our new dataset contains pixel level contextual labels for PASCAL VOC 2010 detection challenge, including all of `train` and `val` images in the dataset, resulting in 10103 annotated images. There are 623 categories in the dataset that are divided into three types: (i) objects, (ii) stuff and (iii) hybrids. *Objects* are classes that have a specific shape. This includes the original 20 categories of

PASCAL as well as classes such as fork, keyboard, cup, etc. *Stuff* are things that do not have a specific shape and appear as regions in images, e.g., sky, water. Finally, *hybrid* are things that have shape but the shape is so variable that cannot be easily modeled. For instance, roads have clear boundary (unlike sky for example), but the road shape cannot be modeled as easily as the shape of a cup.

The dataset has been fully annotated by 6 people, trained and calibrated by us. We did not use crowd sourcing in order to ensure high quality labellings. While this increases the cost significantly, we wanted to assure the highest possible accuracy and consistency of the annotations. The full labeling process took roughly three months to complete.

The annotators were asked to draw a region and assign it a label using an interface similar to LabelMe's [RTM08]. There are about 12 regions in each image on average and the annotators spent about 3 to 5 minutes per image. Some of the images are more complex and include lots of details, which required more time to be spent on.

We provided the annotators with an initial set of carefully chosen 80 labels and asked them to include more classes if an object did not fit into any of these classes. Some cases were ambiguous to annotate; for example, the annotators were not sure how to label a tree visible through a window. In this case, a pixel can be both tree and a window. We decided to go for a rich set of annotations, and thus allowed some pixels to have multiple labels, tree and window in this example. Moreover, if the annotators were unable to recognize a region, they labeled it as unknown. We double checked the annotations for each single image in the dataset and revised the ones that were not correct in terms of category name or the region covering the object.

For the analysis conducted of this chapter, we decided to select the top 100 most frequent categories and pruned out the less frequent ones. We assign the *background* label to the pruned out classes. Some of these 100 categories were visually similar, which we merged into the final set of 60 classes (including the original 20 PASCAL categories). For example, the categories of "picture" and "poster" are similar and were merged into one class.

Among these 60 categories, only 24 of them appear in the immediate surrounding of the twenty

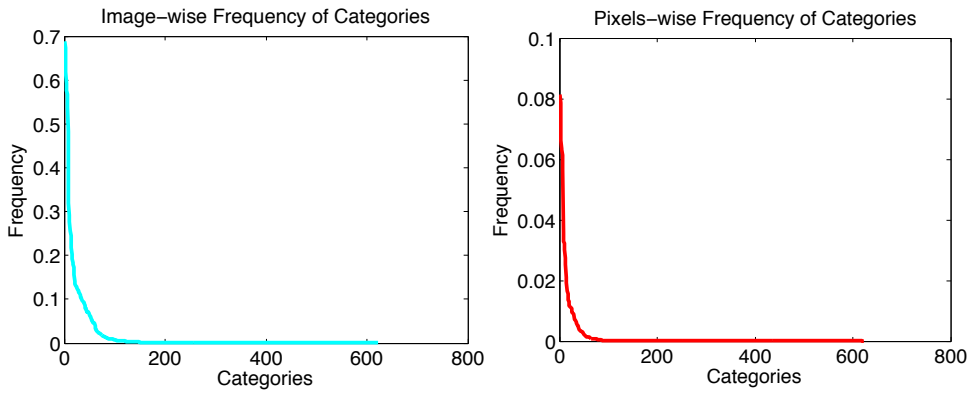


Figure 8.1: Distribution of classes in PASCAL. The frequency of appearance in images (left) and pixel-wise frequency (right) are shown.

PASCAL objects and have high correlation with them. As current automatic segmentation approaches fail to reliably detect some of these classes (e.g., the performance of cloth and bed-cloth is almost zero), for our object detection experiments, we use a final set of 14 context classes, and merge the rest into background. Recall that we assigned some pixels more than one label in GT. If a pixel is assigned to one of these 14 labels and has multiple labels, we ignore the other labels. We note that there is no confusion among these 14 categories. In the experiment section, we provide the list and segmentation accuracy for these classes.

The frequency of appearance of the categories is shown in Figure 8.1. The left panel shows in how many images each class appears. The right panel shows the frequency in terms of the number of pixels for each class. As expected, the distribution is highly heavy-tailed in both cases. We also show the 20 most frequent categories in Figure 8.2.

The type of object context varies according to its size. In Figure 8.3, we show the frequency of each context class with respect to different object size percentiles. To compute the statistics, we consider as context a box with the same size as the object in four different directions around the object (top, bottom, left, and right), with no space between these boxes and the original object box. The statistics represent the normalized number of pixels for each class. The normalization is done according to the total number of pixels that fall in the boxes of a particular direction.

There are some interesting trends. For instance, the amount of sky in the bottom region of

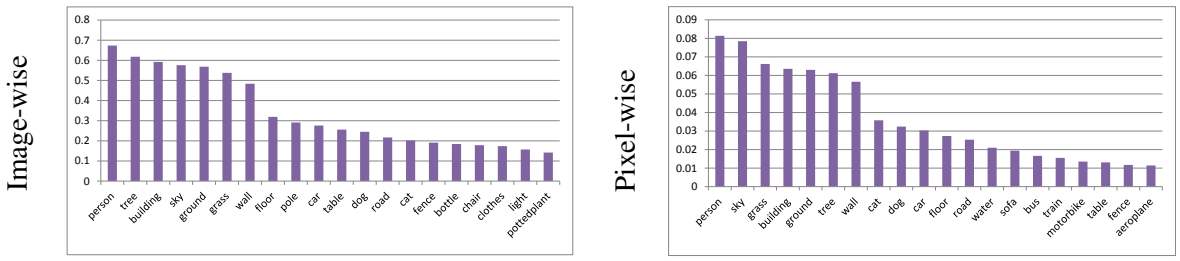


Figure 8.2: The most frequent categories of our dataset in terms of presence in images and pixel–wise.

aeroplanes increases as aeroplanes become smaller, which shows that small aeroplanes typically appear in sky. Another example is that we see more sky pixels in the top region of buses compared to cars, which shows buses are taller than cars. The histogram of the 14 context classes (irrespective of size) is shown in Figure 8.4 for two example categories.

It is evident that the surroundings of objects have a very biased distribution, which should be exploited particularly when recognizing "difficult" / ambiguous object regions. For example, for tiny objects where little of the structure is visible, or for highly occluded objects, context should play key role in recognition.

8.2 A Contextual Model for Object Detection

In order to investigate the role of context in object detection, we designed a novel category level object detector which exploits the global and local context around each candidate detection. By **global context** we mean the presence or absence of a class in the scene. By **local context**, we mean the contextual classes that are in the vicinity of the detected object.

Towards this goal we designed a deformable contextual model, which is a part-based model, with additional random variables denoting deformable contextual parts which score the "contextual stuff" around the object. Additionally, we incorporate global context representing which context classes are likely to be in the image. This allows us to bias which object detectors should be more likely to fire for a particular image. Unlike most existing approaches, we perform contextual

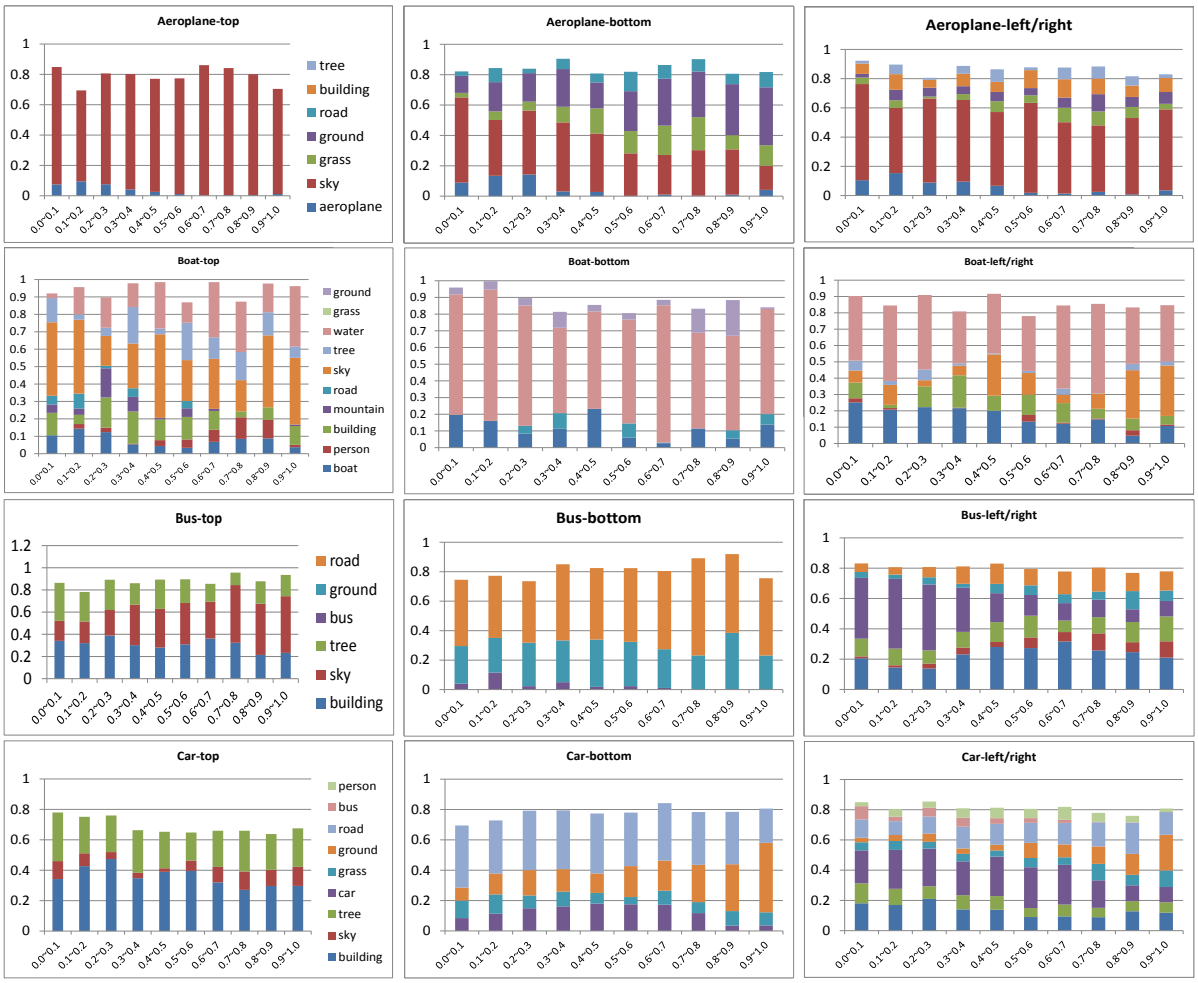


Figure 8.3: The pixel-wise frequency of context classes in top, bottom, and left/right of the objects is shown. The x-axis corresponds to size percentile and the y-axis represents the frequency of appearance. Only the most correlated classes are shown.

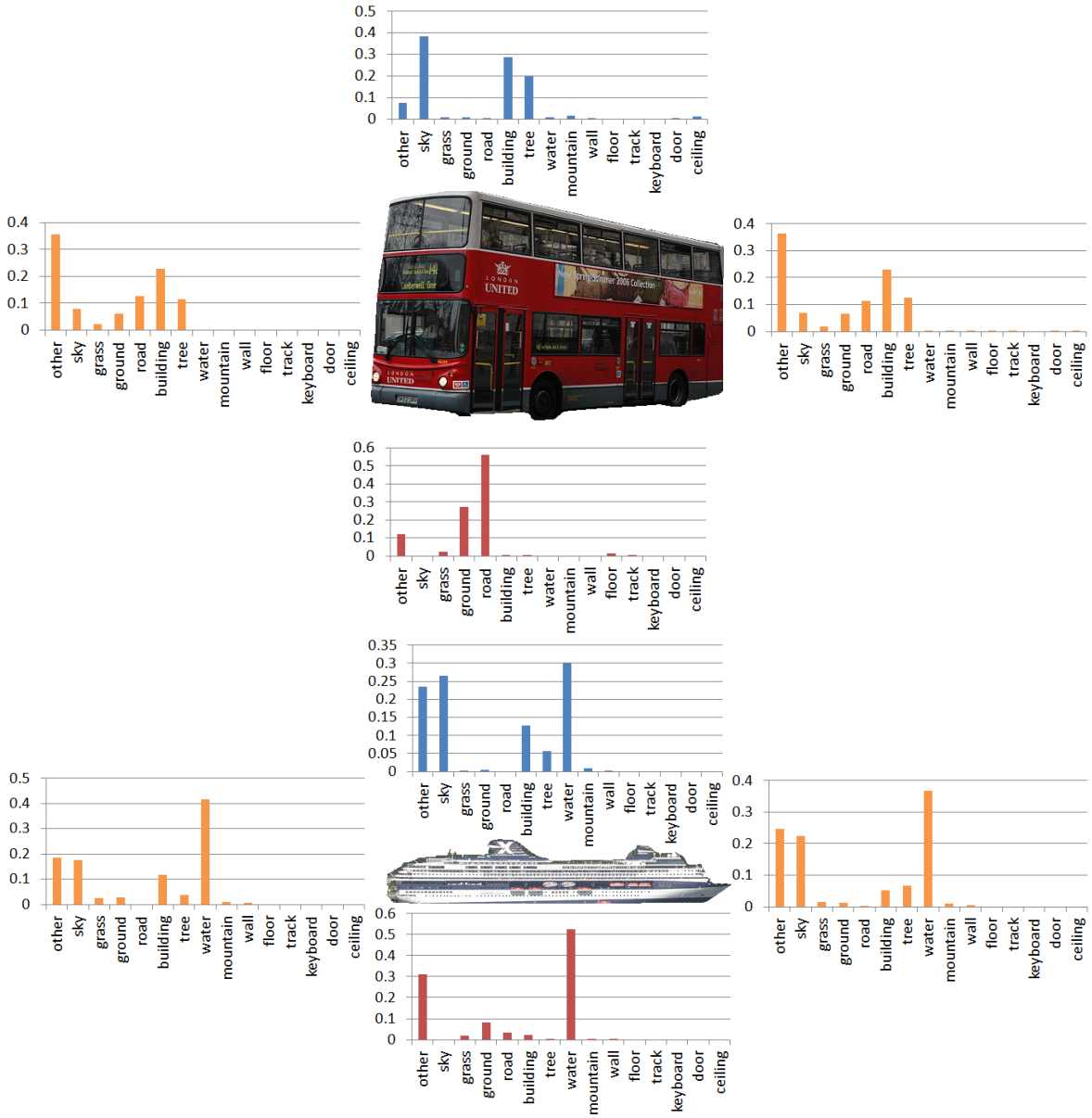


Figure 8.4: Histograms of 14 context classes in different regions around the bus and boat categories.

reasoning while reasoning about the exponentially many possible detections in each image.

We follow the notation of [FGM10], and define p_0 as a random variable encoding the location and scale of a bounding box in an image pyramid as well as the mixture component id. This mixture is employed to represent the variability in appearance and aspect ratios of the different training examples. Let $\{p_i\}_{i=1,\cdots,P}$ be a set of parts which encode bounding boxes at double the resolution of the root. Denote with $\{c_i\}_{i=1,\cdots,C}$ a set of context variables describing the location of our contextual parts.

We frame the detection problem as inference in a Markov random field (MRF), where we score a detection by scoring each root filter, as well as two types of parts: appearance and contextual. Additionally, deformations between the root and the parts (both appearance and contextual) are utilized, as both are deformable. These deformations penalize the placements of both types of parts as a function of how far they are with respect to an anchor position. These anchors represent the expected location of each part type with respect to the root.

We thus write the score of a detection as the sum of three terms, the appearance, the deformation and the context.

$$E(\mathbf{p}, \mathbf{c}) = E_{app}(x, \mathbf{p}) + E_{cont}(x, \mathbf{c}) + E_{def}(x, \mathbf{p}, \mathbf{c}) \tag{8.1}$$

where x is the image, $\mathbf{c}$ is the set of contextual part placements and $\mathbf{p} = \{p_0, \cdots, p_K\}$, the root location, scale and component id, as well as the placements of the appearance parts. Assuming a linear model, we define

$$E(\mathbf{p}, \mathbf{c}) = \underbrace{\sum_{i=0}^{P} \mathbf{w}_i^T \cdot \phi(x, p_i)}_{appearance} + \underbrace{\sum_{i=1}^{P} \mathbf{w}_{i,def}^T \cdot \phi(x, p_0, p_i)}_{part\ deformation} +$$
$$+ \underbrace{\sum_{i=1}^{C} \mathbf{w}_{lc}^T \phi(x, \mathbf{c})}_{local\ context} + \underbrace{\mathbf{w}_{gx}^T \phi_{gc}(x)}_{global\ context} + \underbrace{\sum_{i=1}^{C} \mathbf{w}_{i,defc}^T \phi(x, p_0, c_i)}_{context\ deformation}$$

As in [FGM10], we employ a HOG pyramid to compute $\phi(x, p_0)$, and employ double the resolution to compute the part features $\phi(x, p_i)$. For the appearance parts, we use the same quadratic

deformation cost of [FGM10] and learn the anchors from training data. In our context model, we have four local context parts corresponding to top, bottom, left and right side of the root filter. The area of the context box is one third of the area of the root filter. If a context box crosses the boundary of the image, we consider only the area that lies inside the image. Depending on the location of the context box, its dimensions varies. For instance, the height of the top box is $1/3$ of the height of the root filter, and its width is the same as the width of the root filter.

To derive the context features, we first train a pixel-level multi-class context classifier using [CCB12] (details in Section 8.3). We use a threshold common to all classes to transform the predicted scores into binary features for each pixel and each class. For global context, we use a binary feature for each context class, where 1 indicates that at least a 1000 pixels were predicted to be of this class in the image. Note that the global context does not depend location, but on the class, as we learn different weights for the different object detectors. This effectively reduces/increases the score of a particular class depending on global context (i.e., type of scene).

We use very simple features as local contextual features which enable very efficient computation. In particular, we employ counts of pixels of each context class, normalized by the area of the context part's box. As a consequence we can compute the features in constant time by employing a single integral image per context class. Figure 8.5 depicts the graphical model. Note that we have an additional plate with the contextual random variables.

Learning: The model is learned using latent structured SVMs. We utilize as loss function a 0-1 loss on detection [GFM11]. Stochastic gradient descent is employed to optimize the non-convex objective. For initialization, we first train a mixture model as in [GFM11], without context. We then add context parts, initializing the weights to 0, and perform several iterations of learning of the joint model. We add the HOG parts, and train the full model using warm start.

Inference: Our model forms a tree, and thus exact inference is possible using dynamic programming. We start with the leaves, which require computing the score for each root filter, HOG part, and context part. For the context parts, we first compute integral images for each context class, and

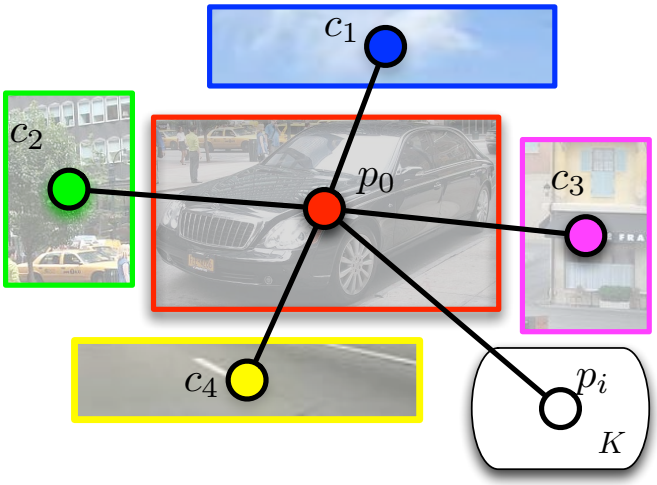


Figure 8.5: The graphical model for our detector. Four context boxes are shown in different colors and correspond to top, bottom, left, and right regions around the root filter.

then score a part in every location and scale of the pyramid. From here on, dynamic programming is agnostic about which part (whether HOG or context) it is using, thus computing the deformations and the final score follows [FGM10].

8.3 Experimental Evaluation

In this section, we explain the details of our experiments. For all of the experiments, we use the `train` portion of PASCAL VOC 2010 detection dataset for training and the `val` subset for evaluation.

Our first experiment is to classify the context classes. As mentioned before, for our experiments, we use 14 classes that have high correlation with the original 20 PASCAL classes and their detection performance is reasonable. So we ignored classes such as cloth and picture for which we obtain almost zero recall. We use the following classes as context in our model: sky, grass, ground, road, building, tree , water, mountain, wall, floor, track (railroad), keyboard, door, and ceiling.

Towards this goal, we first generate UCM superpixels [AMF11]. On average, there are 67 superpixels in each image. We then use the second order pooling method of [CCB12] to learn a

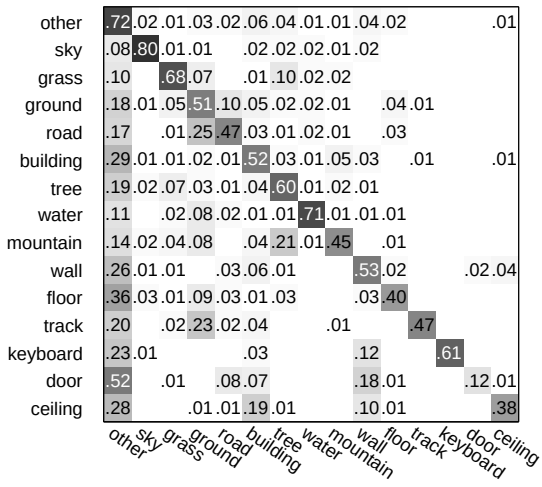


Figure 8.6: Confusion matrix for classification of context classes.

multi-class context classifier. The difference of our method with [CCB12] is that we use UCMs as the input to the classifier instead of CPMC segments. The label for each superpixel is the class with the highest score. We assign it a "background" label if the scores of all of the context classes is below a threshold of 0.35. The resulting confusion matrix is shown in Figure 8.6. This approach is very successful for all classes but door. This is expected as doors are difficult to detect at the superpixel level as they look like walls. Keyboard is also pretty difficult and would be better amenable to approaches that reason at the object level.

In the next experiment, the task is *image annotation*, where for each image we are interested in knowing if a specific category is in the image or not (independent of the number of pixels). For instance, if the image contains sky and we predict at least one pixel of sky, we count that image as a true positive. The average precision for this classification task is shown in Table 8.1. This experiment indicates how reliable global context can be for our model.

	sky	grass	ground	road	building	tree	water	mountain	wall	floor	track	keyboard	door	ceiling	**average**
	Image-level Context Classification														
AP	89.1	80.8	60.7	43.2	67.7	76.7	70.9	38.4	84.7	59.2	66.7	34.3	21.0	33.4	59.05

Table 8.1: Average precision for detecting presence of a context class in an image.

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motorbike	person	plant	sheep	sofa	train	tv	mAP
Dalal **XS**	0.1	8.5	0	0	0	0.1	0	2.2	0	0	0.4	1.5	0.5	0.1	0	0	0	5.7	9.1	1.8	1.5
Dalal+context **XS**	4.2	20.5	0.2	10.8	3.4	6.6	0.9	0.1	0	9.8	0.1	0.3	18.3	18.7	0.4	0.7	9.5	4.7	22.8	12.6	7.23
Dalal **S**	25.3	36.5	1.3	0.6	0.1	20.9	2.7	1.3	1.2	11.7	11.2	3.2	28.7	20.4	5.5	1.5	15.8	21.5	37.7	28.8	13.79
Dalal+context **S**	17.7	45.4	1.7	11.9	32.9	20.6	21.3	2.2	2.7	15.5	18.8	3.4	33.9	30.9	9.8	2.4	38.0	32.9	34.5	21.3	19.9
Dalal **M**	59	42.5	4.6	6.6	23.2	66.8	26.8	11.2	14	20.3	21.5	6.4	45	41.7	15.2	5.7	36.2	24.9	48.1	41.2	28.04
Dalal+context **M**	63.8	47.9	7.1	5.9	19.8	68.9	28.3	11.2	14.7	21.7	23.7	8.1	48.0	42.3	14.7	5.4	42.7	36.8	50.2	44.6	30.3
Dalal L	41.8	65.3	9.1	18.9	34	84.7	55.3	38	17.9	11	16	10.7	31	62.3	28.4	6.2	28.5	9.6	46.1	45.3	33.00
Dalal+context **L**	60.2	72.8	14.1	24.5	29.8	86.7	51.7	37.2	15.1	18.5	25.0	12.9	39.6	64.0	27.1	6.7	27.0	30.0	50.6	45.9	37.0
Dalal **XL**	4.7	45.5	8.9	3.2	33.9	59.3	55.1	42	11.5	6.8	12.9	15.3	23.5	49.1	34.4	7.2	17.8	9.4	31.3	26.2	24.9
Dalal+context **XL**	18.4	52.8	11.2	20.7	30.7	58.6	48.4	49.3	12.0	19.4	7.5	18.6	29.3	37.9	30.9	3.0	17.8	10.7	39.4	24.5	27.0

Table 8.2: Normalized Average Precision for detection of 20 PASCAL categories using [DT05]. The highest gain is observed for the XS class.

We evaluate our context model in junction with two different object detectors. We explore the usefulness of context for detecting 20 PASCAL classes. The first detector we use is [DT05], which represents an object with a global rigid template. To evaluate the effect of context, we perform the evaluation for different object sizes separately. For this purpose, we adopt the convention of [HCD12] that cluster objects into XS (extra small), S (small), M (medium), L (large), and XL (extra large). As shown in Table 8.2, the context model provides a significant boost for detection of XS objects. In the table, we use Normalized Average Precision used by [HCD12] as the evaluation metric, which is different from the common Average Precision (AP) metric used for evaluation of object detectors. Since the number of examples in each size category is different, we report the normalized AP to make the results of different sizes comparable.

For some classes such as person, there is no strong contextual cue as humans can appear in very different contexts. Also, the context model does not help classes such as cat for which we cannot reliably detect the context class. As shown in Figure 8.6, our performance is typically low for detecting indoor classes.

Additionally, we incorporate our context model into DPM [FGM10]. We report Normalized Average Precisions in Table 8.3. Similar to the previous case, context provides a significant im-

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motorbike	person	plant	sheep	sofa	train	tv	mAP
DPM **XS**	0.3	19.2	0	0	0	0	0	2.0	0	0	0.2	0.2	1.8	0.5	0	0	0	6.0	10.8	0.5	2.1
DPM+context **XS**	11.8	23.5	0.3	13.6	5.5	4.6	6.2	2.3	0.2	7.2	1.6	0.4	15.2	22.0	10.5	0.4	10.2	8.5	13.4	17.4	8.7
DPM **S**	38.5	49.6	0.7	1.1	1.5	30.7	15.4	11.8	3.6	5.9	19.0	10.7	31.9	25.4	19.2	2.0	21.2	37.3	37.4	28.9	19.6
DPM+context **S**	37.9	44.9	6.6	10.4	34.7	26.4	35.9	10.2	3.1	19.4	11.4	2.8	37.1	27.3	25.6	0.2	38.7	35.0	37.8	33.3	23.9
DPM **M**	70.9	54.4	5.5	8.2	29.6	74.5	42.6	20.5	18.6	26.1	29.0	17.8	59.5	53.0	30.8	6.9	48.6	48.2	57.4	46.5	37.4
DPM+context **M**	68.2	53.1	6.9	11.3	27.3	71.1	42.4	23.9	16.2	28.2	25.8	9.4	53.4	49.8	28.0	9.9	45.3	45.1	56.7	49.6	36.1
DPM L	75.4	79.8	16.7	35.8	46.9	88.1	67.9	59.5	26.4	26.8	26.9	24.6	49.6	73.1	44.6	12.4	38.4	27.2	56.7	61.2	46.9
DPM+context **L**	70.2	77.1	21.7	42.6	47.6	85.6	64.9	54.1	17.4	22.6	26.0	20.4	45.1	73.4	35.8	11.2	37.9	30.9	62.2	58.6	45.3
DPM **XL**	20.7	57.6	13.6	13.8	45.7	66.0	69.3	74.1	16.8	27.1	18.8	24.2	30.1	66.9	55.7	8.1	19.3	12.9	51.1	48.4	37.0
DPM+context **XL**	20.2	58.0	17.4	17.3	46.9	64.6	61.7	63.6	10.2	34.3	20.7	20.7	29.7	55.3	49.7	9.9	21.3	14.3	51.1	36.7	35.2

Table 8.3: Normalized Avg. Precision for detection of 20 PASCAL categories using DPM and DPM+context. The contextual information provides significant improvement for the XS objects.

provement for XS class. However, since the object appearance model is stronger in DPM, the contextual information is less effective for larger objects. Figure 8.7 shows the pixel-level context labeling and detection results on example images.

8.4 Conclusion

We have presented a novel large-scale dataset that labels each pixel in the PASCAL VOC 2010 detection challenge with one or more context classes. In particular, our labels contain 623 classes divided in a taxonomy. We have also introduced a novel object detector which exploits local and global context by adding contextual parts in a deformable part-based model. Our experiments showed that context is extremely useful when detecting tiny objects, particularly for weak detectors (e.g., rigid templates).

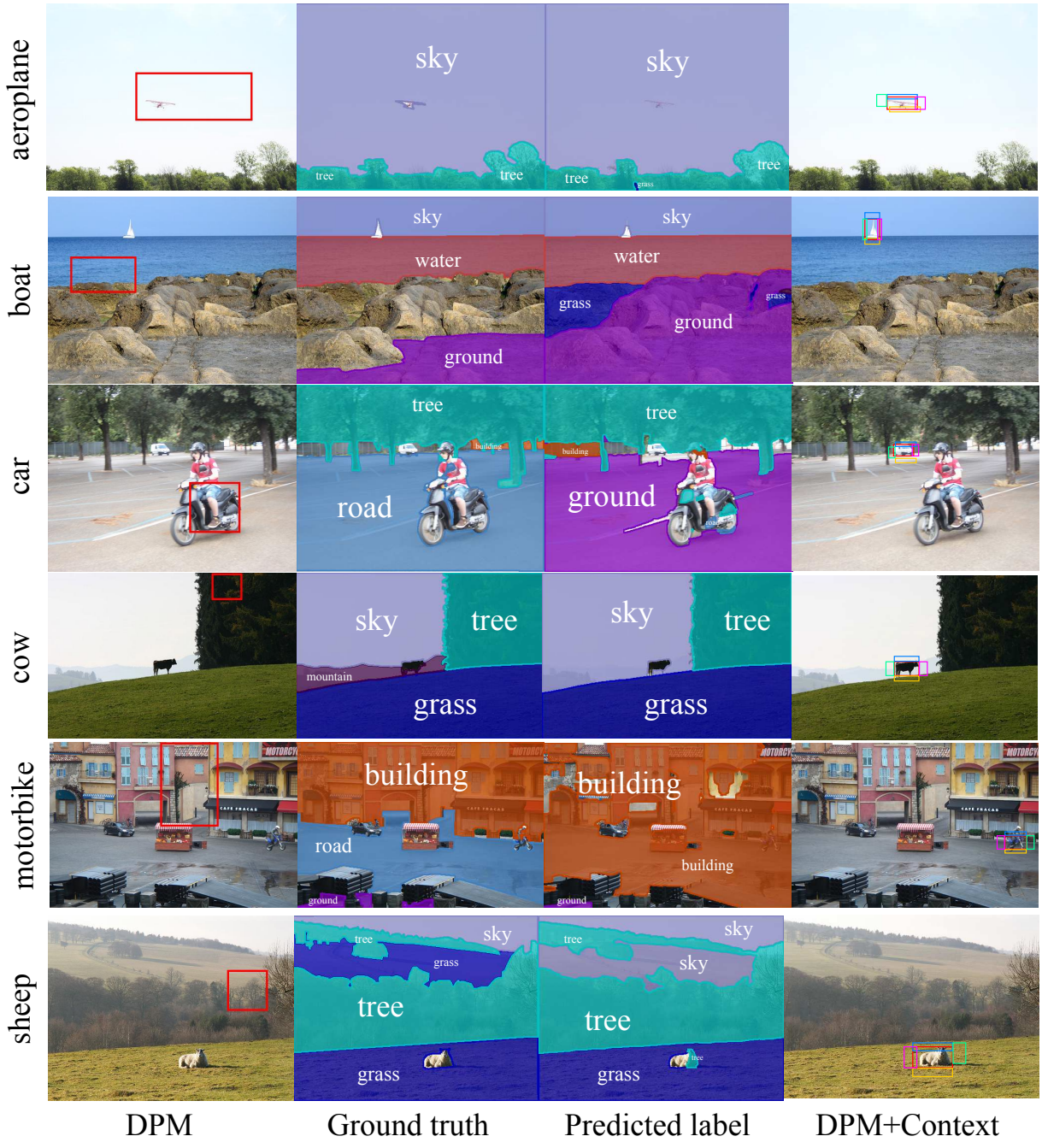


Figure 8.7: Example cases of small objects that are missed by DPM, while they are correctly localized when we incorporate context. In the first column, we show the top detection of DPM. The second column is GT context labeling. Third column is the context prediction result. The last column is the result of our context-aware DPM. Inferred context boxes are also shown with different colors.

CHAPTER 9

Identifying Bottlenecks in Scene Understanding

The main focus of the previous chapters was object detection and segmentation. In this chapter, we tackle the more challenging scene understanding problem that includes detection and segmentation as its sub-tasks. Through a series of hybrid human-machine experiments, we would like to identify the bottleneck in scene understanding methods.

Automatic holistic scene understanding is one of the holy grails of computer vision. Given the lofty challenge it presents, the community has historically studied individual tasks in isolation. This includes tasks such as object detection [FGM10], scene recognition [XHE10], contextual reasoning among objects [RVG07], and pose estimation [YR11]. However, clearly these tasks are related. For example, knowing that the image is a street scene influences where and at what scales we expect to find people. Detecting a microwave in an image can help identify a kitchen scene. Studies have shown that humans can effectively leverage contextual information from the entire scene to recognize objects in low resolution images that can not be recognized in isolation [Tor09]. In fact, different and functionally complementary regions in the brain are known to co-operate to perform scene understanding [PBG11].

Recent works [GGK09, YFU12, HGS08, LKS10], have thus pushed on *holistic* scene understanding models. The advent of general learning and inference techniques for graphical models has provided the community with appropriate tools to allow for joint modeling of various scene understanding tasks. These have led to some of the state-of-the-art approaches.

Our aim is to determine the relative importance of the different recognition tasks in aiding holistic scene understanding. We wish to know, which of the tasks if improved, can boost performance significantly. In other words, to what degree can we expect to improve holistic understanding per-

formance by improving the performance of individual tasks? We argue that understanding which problems to solve is as important as determining how to solve them. Such an understanding can provide valuable insights into which research directions to pursue for further improving the state-of-the-art. We use semantic segmentation, object detection and scene recognition accuracies as proxies for holistic scene understanding performance.

We analyze the most recent and comprehensive holistic scene understanding model of Yao *et al.* [YFU12]. It is a conditional random field (CRF) that models the interplay between a variety of factors such as local super-pixel appearance, object detection, scene recognition, shape analysis, class co-occurrence, and compatibility of classes with scene categories. To gain insights into the relative importance of these different factors or tasks, we isolate each task, and substitute a machine with a human for that task, keeping the rest of the model intact. The resultant improvement in performance of the model, if any, gives us an indication of how much "head room" there is to improve performance by focusing research efforts on that task. Note that human outputs are *not* synonymous with ground truth information, because the tasks are performed in isolation. For instance, humans would not produce ground truth labels when asked to classify a super-pixel in isolation into one of several categories[1]. In fact, because of inherent local ambiguities, the most intelligent machine of the future will likely be unable to do so either. Hence, the use of human subjects in our studies is key, as it gives us a *feasible* point of what can be done.

Our slew of studies reveal several interesting findings. For instance, we found that human classification of *isolated* super-pixels when fed into the model provides a 5% improvement in segmentation accuracy on the MSRC dataset. Hence, research efforts focussed towards the specific task of classifying super-pixels in isolation may prove to be fruitful. Even more intriguing is that the human classification of super-pixels is in fact less accurate than machine classification. However when plugged into the holistic model, human potentials provide a significant boost in performance. This indicates that to improve segmentation performance, instead of attempting to

[1]Of course, ground truth segmentation annotations are themselves generated by humans, but by viewing the whole image and leveraging information from the entire scene. In this study, we are interested in evaluating how each recognition task in *isolation* can help the overall performance.

build super-pixel classifiers that make fewer mistakes, research efforts should be dedicated towards making the right kinds of mistakes. This provides a refreshing new take on the now well studied semantic segmentation task.

Excited by this insight, we conducted a thorough analysis of the human generated super-pixel potentials to identify precisely how they differ from existing machine potentials. Our analysis inspired a rather simple modification of the machine potentials which resulted in a significant increase of 2.4% in the machine accuracy (i.e. no human involvement) over the state-of-the-art on the MSRC dataset.

In addition, through a series of human studies and machine experiments we show that a reliable object shape model is beneficial for semantic segmentation and object detection. We demonstrate that humans are not good at deciphering object shape from state-of-art UCM segment boundaries any more than existing machine approaches.

We also studied how well humans can leverage the contextual information modeled in the CRF. We measure human and machine segmentation performance while progressively increasing the amount of contextual information available. We find that even though humans perform significantly worse than machines when classifying isolated super-pixels, they perform better than machines when both are given access to the contextual information modeled by the CRF.

The key motivation behind holistic scene understanding, going back to the seminal work of Barrow in the seventies [BT78], is that ambiguities in visual information can only be resolved when many visual processes are working collaboratively. A variety of holistic approaches have since been proposed. Many of these works incorporate the various tasks in a sequential fashion, by using the output of one task (e.g., object detection) as features for other tasks (e.g., depth estimation, object segmentation) [HE08, HGS08, LKR09, BBM11, GLA09]. There are fewer efforts on joint reasoning of the various recognition tasks. In [TMF05], contextual information was incorporated into a CRF leading to improved object detection. A hierarchical generative model spanning parts, objects and scenes is learnt in [STF05]. Joint estimation of depth, scene type, and object locations is performed in [LKS10]. Spatial contextual interactions between objects have also been modeled [KH05, RVG07]. Image segmentation and object detection are jointly modeled

in [LSA10, WS08, GGK09] using a CRF. [GBW10] also models global image classification in the CRF. In this work, orthogonal to these advances, we propose the use of human subjects to understand the relative importance of various recognition tasks in aiding holistic scene understanding.

Numerous human-studies have been conducted to understand the human ability to segment an image into meaningful regions or objects. Rivest and Cavanagh [RC96] found that luminance, color, motion and texture cues for contour detections are integrated at a common site in the brain. Fowlkes [Fow05] found that machine performance at detecting boundaries is equivalent to human performance in small gray-scale patches. These and other studies are focused on the problem of unsupervised segmentation, where the task is to identify object boundaries. In contrast, we are interested in holistic scene understanding, including the task of identifying the semantic category of each pixel in the image.

Several works have studied high-level recognition tasks in humans. Fei-Fei *et al.* [FVK02] show that humans can recognize scenes rapidly even while being distracted. Bachmann *et al.*[Bac91] show that humans can reliably recognize faces in 16×16 images, and Oliva *et al.* [OS00] present similar results for scene recognition. Torralba *et al.* [Tor09] show that humans can reliably detect objects in 32×32 images. In contrast, we study human performance at tasks that closely mimic existing holistic computational models for holistic scene understanding in order to identify bottlenecks, and better guide future research efforts.

Human debugging i.e. using human subjects to identify bottlenecks in existing computer vision systems has been recently explored for a number of different applications such as analyzing the relative importance of features, amount of training data and choice of classifiers in image classification [PZ10], of part detection, spatial modeling and non-maximal suppression in person detection [PZ11], of local and global image representations in image classification [Par11], and of low-, mid- and high-level cues in detecting object contours [ZP12]. In this work, we are interested in systematically analyzing the roles played by several high- and mid-level tasks such as grouping, shape analysis, scene recognition, object detection and contextual interactions in *holistic scene understanding*. While similar at the level of exploiting human involvement, the problem, the model, the methodologies of the human studies and machine experiments, as well as the findings

and insights are all novel.

9.1 CRF Model

We analyze the recently introduced CRF model of [YFU12] which achieved state-of-the-art performance on the MSRC dataset by reasoning jointly about a variety of scene components. While the model shares similarities with past work [LRK09, LSA10, GBW10], we choose this model because it provides state-of-the-art performance in holistic scene understanding, and thus forms a great starting point to ask "which components need to be improved to push the state-of-the-art further?". Moreover, it has a simple "plug-and-play" architecture making it feasible to insert humans in the model. Inference is performed via message passing and so it places no restrictions (e.g. submodularity) on the potentials. This allows us to conveniently replace the machine potentials with human responses: after all, we cannot quite require humans to be submodular!

We now briefly review this model (Figure 9.1). We refer the reader to [YFU12] for further technical details. The problem of holistic scene understanding is formulated as that of inference in a CRF. The random field contains variables representing the class labels of image segments at two levels in a segmentation hierarchy: super-pixels and larger segments. To be consistent with [YFU12], we will refer to them as segments and super-segments. The model also has binary variables indicating the correctness of candidate object detection bounding boxes. In addition, a multi-labeled variable represents the scene type and binary variables encode the presence/absence of a class in the scene.

The segments and super-segments reason about the semantic class labels to be assigned to each pixel in the image. The model employs these two segmentation layers for computational efficiency: the super-segments are fewer but more densely connected to other parts of the model. The binary variables corresponding to each candidate bounding box generated by an object detector allow the model to accept or reject these detections. A shape prior is associated with these nodes encouraging segments to take on corresponding class labels. The binary class variables reason about which semantic classes are present in the image. This allows for a natural way to model

Unitary potentials:

- Segment, super-segment, class, object detection, scene

Pairwise potentials:

- Segment – Super-segment
- Super-segment – Class
- Class – Scene
- Detection – Class
- Detection – Segment (shape prior)

Loss:

- Sum of task-specific losses

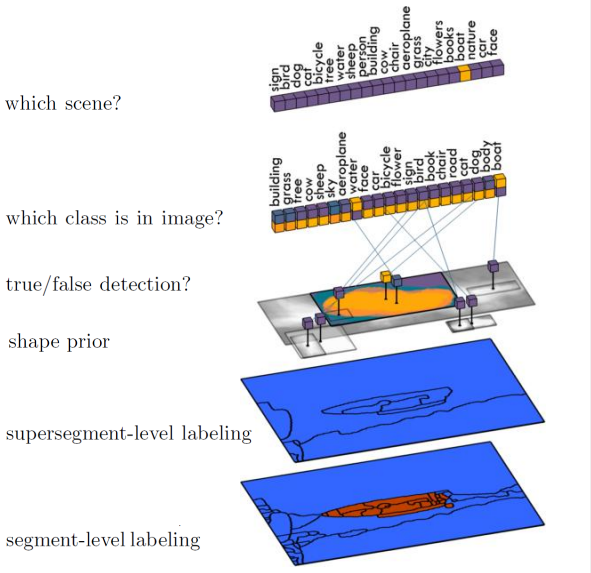


Figure 9.1: Overview of the holistic scene model of [YFU12]. For clarity, not all connections in the model are shown here.

class co-occurrences as well as scene-class affinities. These binary class variables are connected to i) the super-segments via a consistency potential that ensures that the binary variables are turned on if a super-segment takes the corresponding class label ii) binary detector variables via a similar consistency potential iii) the scene variable via a potential that encourages certain classes to be present in certain scene types iv) to each other via a potential that encourages certain classes to co-occur more than others.

More formally, let $x_i \in \{1, \cdots, C\}$ and $y_j \in \{1, \cdots, C\}$ be two random variables representing the class label of the i-th segment and j-th super-segment. We represent candidate detections as binary random variables, $b_i \in \{0, 1\}$, taking value 0 when the detection is a false detection. A part-based mixture model [FGM10] is used to generate candidates. The detector provides us with an object class (c_i), the score (r_i), the location and aspect ratio of the bounding box, as well as the root mixture component ID that has generated the detection (m_i). The latter gives us information about the expected shape of the object. Let $z_k \in \{0, 1\}$ be a random variable which takes value 1 if class k is present in the image, and let $s \in \{1, \ldots, C_l\}$ be a random variable representing the scene type among C_l possible candidates. The parameters corresponding to different potential

semantic labels	scene types	obj. detection classes
building	sign	cow
grass	bird	sheep
tree	dog	aeroplane
cow	cat	face
sheep	bicycle	car
sky	tree	bicycle
aeroplane	water	flower
water	sheep	sign
face	person	bird
car	building	book
bicycle	cow	chair
flower	chair	cat
sign	aeroplane	dog
bird	grass	body
book	city	boat
chair	flowers	
road	books	
cat	boat	
dog	nature	
body	car	
boat	face	

Table 9.1: MSRC-21 dataset information

terms in the model are learnt in a discriminative fashion. Before we provide details about how the various machine potentials are computed, we first discuss the dataset we work with to ground further descriptions.

9.2 Dataset

We use the standard MSRC-21 [SWR09] semantic labeling benchmark, also used by [YFU12], as it contains "stuff" (e.g., *sky, water*) as well as "things" (i.e., shape-defined classes such as *cow, car*). The PASCAL dataset is more challenging in terms of object ("things") detection and segmentation. However, a large portion of its images, especially "stuff", is unlabeled (at the time this work was started. Annotations of Chapter 8 was accomplished afterwards). The SUN dataset [XHE10] is prohibitively large for the scale of human studies involved in our work. The SIFT-flow dataset [LYT09] is dominated by "stuff" with a small proportion of "things" pixels. Camvid [BSF08] is limited to street scenes. The MSRC dataset is widely used, contains stuff, things and a diverse set of scenes, making it the best choice among existing datasets for our study.

We use the more precise ground truth of MSRC provided by Malisiewicz and Efros [MA07] and used in [YFU12], as it offers a more accurate measure of performance. We use the same scene category and object detection annotations as in [YFU12]. Table 9.1 lists this information. As the performance metric we use average per-class recall (average accuracy). Similar trends in our results hold for average per-pixel recall (global accuracy [LSA10]) as well. We use the standard train/test split from [SJC08] to train all machine potentials, described next.

9.3 Machine & Human CRF Potentials

We now describe the machine and human potentials we employed. Section 9.4 presents the results of feeding the human "potentials" into the machine model. Our choices for the machine potentials closely follow those made in [YFU12]. For human potentials, we performed all human studies on Amazon Mechanical Turk. Unless specified otherwise, each task was performed by 10 different subjects. Depending on the task, we paid participants $3-5$ cents for answering 20 questions. The response time was fast, taking 1 to 2 days to perform each experiment. We randomly checked the responses of the workers and excluded those that did not follow the instructions[2]. More than 500 subjects participated in our studies that involved $\sim 300,000$ crowd-sourced tasks, making the results obtained likely to be fairly stable across a different sampling of subjects.

9.3.1 Segments and super-segments

Machine: We utilize UCM [AMF11] to create our segments and super-segments as it returns a small number of segments that tend to respect the true object boundaries well. We use thresholds 0.08 and 0.16 for the segments and super-segments respectively. On average, this results in 65 segments and 19 super-segments per image for the MSRC dataset. We use the output of the modified Textoonboost [SWR09] in [LRK09] to get pixel-wise potentials and average those within the segments and super-segments to get the unary potentials. Following [KKT07], we connect the two levels via a pairwise P^n potential that encourages segments and super-segments to take the same label.

Human: The study involves having human subjects classify segments into one of the semantic categories. We experiment with three different visualizations, which are depicted in Figure 9.2. The first interface (left) shows all the information that the machine exploited when classifying a segment i.e. the image area in which the image features used by the classifier were extracted. Note

[2] As our experiments will demonstrate, the fact that we can train the CRF parameters on responses of human subjects and have it generalize well to human responses to held out test images vouches for the reliability of the collected human responses.

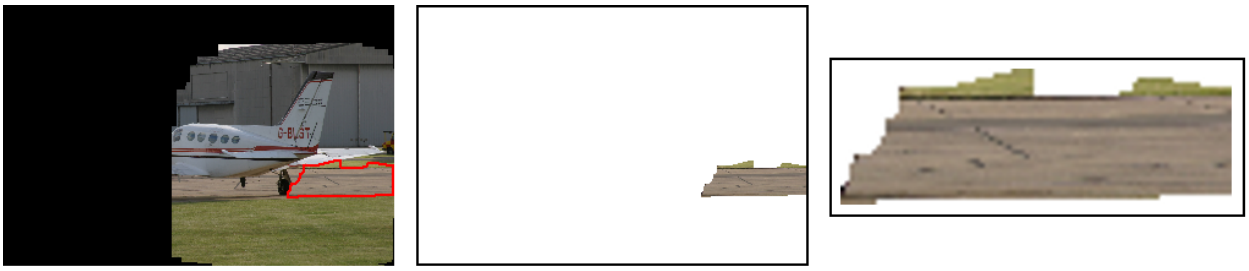


Figure 9.2: Human segment labeling interface. Left panel: Human subjects are asked to classify the segment outlined with a red boundary, where they can see all the information that the machine classifier has access to. Middle panel: Only the segment is shown to the human subjects. Right panel: Location and scale information is discarded when the segment is shown to the subjects.

however, that the machine employed a bag-of-words model to compute the machine potentials and does not exploit the relative location information. The machine classifier, TextonBoost [SWR09] in particular, has access to a large neighborhood (200x200 pixels) around the segment. Clearly, it does not use information only from the pixels in the segment while classifying the segment. However, showing all the information that the machine uses to human subjects would lead to nearly 100% classification accuracy by the subjects, leaving us with little insights to gain. Therefore, we explored two other visualizations with impoverished information. The second interface (middle) only shows the pixels that belong to the segment. The third visualization (right) does the same, but discards the scale and location information of the segment, which is shown at the center of the image and its largest dimension is resized to 240 pixels. We asked the subjects to classify 25 segments selected for each class at random from the set of segments containing more than 500 pixels.

The pixel-wise accuracies obtained via the three visualizations are 99.2%, 79.0%, and 75.6%, respectively. We chose the second interface for the rest of our experiments, which involved having subjects label all segments and super-segments from the MSRC dataset containing more than 500 pixels. This resulted in 10976 segments and 6770 super-segments. They cover 90.2% and 97.7% of all pixels in the dataset[3].

[3]Covering 100% of the pixels in the dataset would involve labeling three times the number of segments, and the resources seemed better utilized in the other human studies.

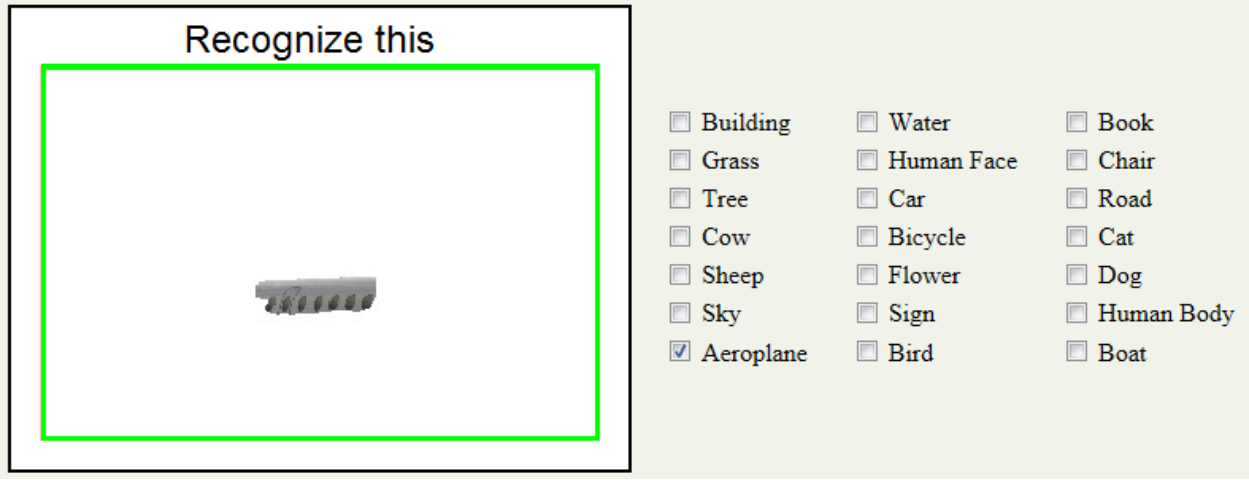


Figure 9.3: Segment labeling interface. We ask the human subjects to choose the category that the segment belongs to. If the subjects are confused among a few categories, they have the option of choosing more than one answer.

We experimented with several interfaces e.g. showing subjects a collection of segments and asking them to click on all the ones likely to belong to a certain category, or allowing a subject to select only one category per segment, etc. before converging to the one that resulted in most consistent responses from subjects (Figure 9.3) where subjects are asked to select all categories that a segment may belong to.

Note that a 200 x 200 window (used by the machine classifier) occupies nearly 60% of the image. If this were shown to the human subjects, it would result in them potentially using holistic scene understanding while classifying the segments. This would contradict our goal of having humans perform individual tasks in isolation. More importantly, a direct comparison between humans and machines is not of interest to us. We are interested in understanding the potential each component in the model holds. To this goal, the discrepancy in information shown to humans and machines is not a concern, as long as humans are not shown *more* information than the machine has access to.

Figure 9.4 shows examples of segmentations obtained by assigning each segment to the class with most human votes. The black regions correspond to either the "void" class (unlabeled regions in the MSRC dataset) or to small segments not being shown to the subjects. Assigning each

Figure 9.4: Human labeling results on isolated segments.

segment to the class with the highest number of human votes achieves an accuracy of 72.2%, as compared to 77.4% for machines[4]. Accuracy for super-segments is 84.3% and 79.6% respectively. As expected, humans perform rather poorly when only local information is available. However, they are better at classifying certain "easy", distinctive, classes or classes they are familiar with e.g., *faces* (see confusion matrix in Figure 9.11(b)).

The C dimensional human unary potential for a (super)segment is proportional to the number of times subjects selected each class, normalized to sum to 1. We set the potentials for the unlabeled (smaller than 500 pixels) (super)segments to be uniform.

9.3.2 Class occurrence and co-occurrence

Machine: We use class-occurrence statistics extracted from training data as a unary potential on z_k. We also employ pairwise potentials between z_i and z_k that capture co-occurrence statistics of pairs of classes. However, for efficiency reasons, instead of utilizing a fully connected graph, we use a tree-structure obtained via the Chow-Liu algorithm [CL68] on the class-class co-occurrence matrix.

Human: To obtain class-occurrence, we showed subjects 50 random images from the MSRC dataset to help them build an intuition for the image collection (not to count the occurrence of

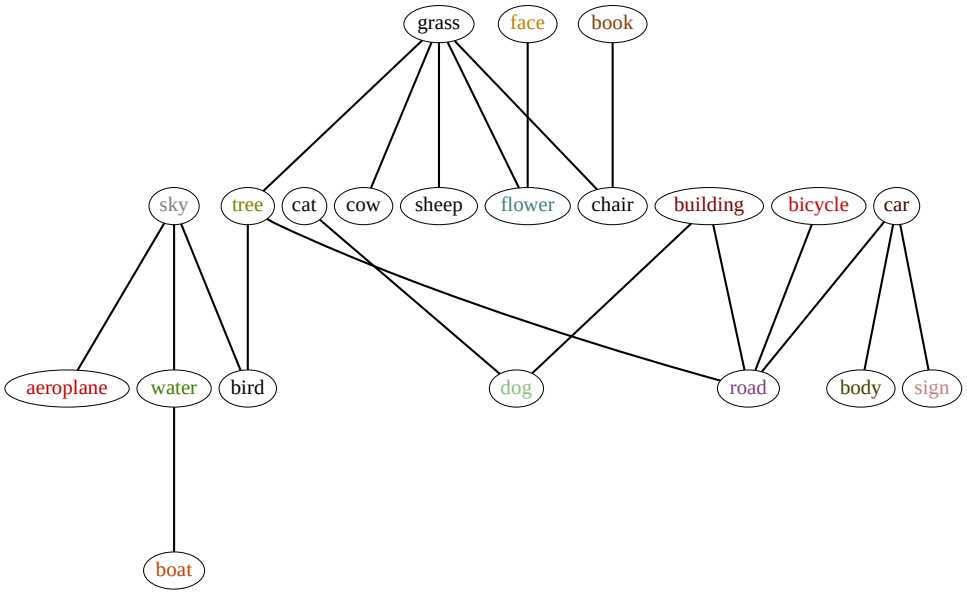
[4]This accuracy is calculated only over segments larger than 500 pixels that were shown to humans. Machine accuracy over all segments is 74.2%.

objects in the images). For all pairs of categories, we then ask subjects which category is more likely to occur in an image from the collection. We build the class unary potentials by counting how often each class was preferred over all other classes. We ask MAP-like questions ("which is more likely") to build an estimate of the marginals ("how likely is this?") because asking subjects to provide scalar values for the likelihood of something is prone to high variance and inconsistencies across subjects.

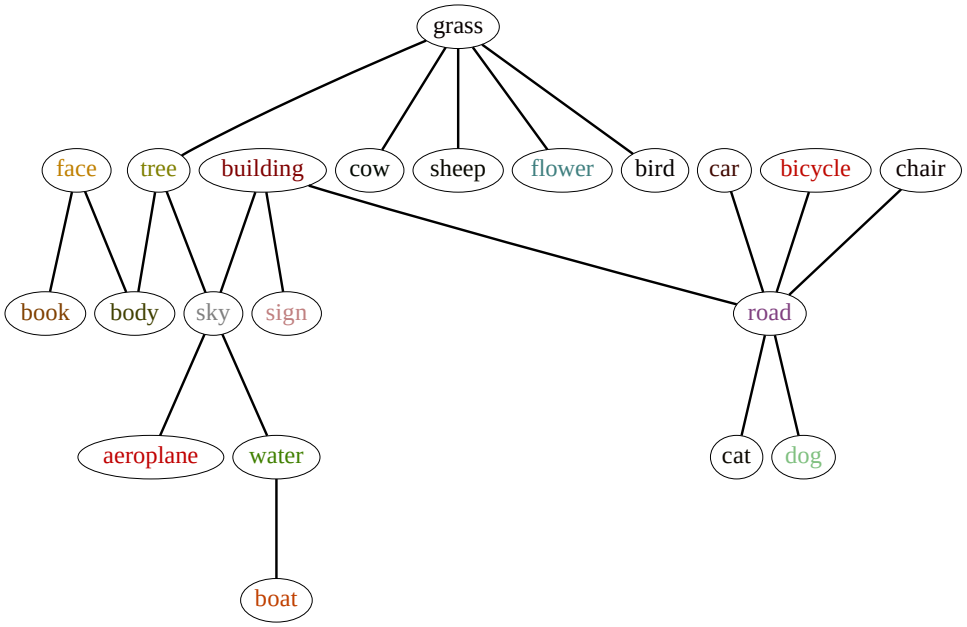
To obtain the human co-occurrence potentials we ask subjects the following question for all triplets of categories $\{z_i, z_j, z_k\}$: "Which scenario is more likely to occur in an image? Observing (z_i and z_j) or (z_i and z_k)?". Note that in this experiment we did not show subjects any images. The obtained statistics thus reflect human perception of class co-occurrences as seen in the visual world in general rather than the MSRC dataset. Given responses to these questions, for every category z_i, we count how often they preferred each category z_j over the other categories. This gives us an estimate of $P(z_j|z_i)$ from humans. We compute $P(z_i)$ from the training images to obtain $P(z_i, z_j)$, which gives us a 21×21 co-occurrence matrix. We use the Chow-Liu algorithm on this matrix, as was used in [YFU12] on the machine class co-occurrence potentials to obtain the tree structure, where the edges connect highly co-occurring nodes. As shown in Figure 9.5, the structure of the human tree is quite similar to the tree obtained from the MSRC training set. For example, in both trees, there are edges between *grass* and categories like *cow*, *sheep*, and *flower*. However, some edges exist in the human tree that are missing in the machine tree e.g., the edge between sky and bird.

9.3.3 Detection

Machine: Detection is incorporated in the model by generating a large set of candidate bounding boxes using the deformable part-based model [FGM10] which has multiple mixture components for each object class. The CRF model reasons about whether a detection is a false or true positive. On average, there are 16 hypotheses per image. A binary variable b_i is used for each detection and it is connected to the binary class variable, z_{c_i}, where c_i is the class of the detector that fired for the $i-$th hypothesis.



(a) Human



(b) Machine

Figure 9.5: Chow-Liu trees for humans and machine. The co-occurring categories are connected by edges. The machine tree is obtained using co-occurrence statistics from labeled training data from the MSRC dataset. The human tree is obtained by asking people (without showing them images) which pairs of categories are more likely to co-occur in a scene, and thus reflects the natural statistics of the world as perceived by humans. Both trees share several similarities.

Human: Since most objects in the MSRC dataset are quite big, it is expected that human object detection would be nearly perfect. As a crude proxy, we showed subjects images inside ground truth object bounding boxes and asked them to recognize the object. Performance was almost perfect at 98.8%. Hence, we use the ground truth object bounding boxes to simulate human responses.

9.3.4 Shape

Machine: Shape potentials are incorporated in the model by connecting the binary detection variables b_i to all segments x_j inside the detection's bounding box. The prior is defined as an average training mask for each detector's mixture component. The values inside the mask represent the confidence that the corresponding pixel has the same label as the detector's class. In particular, for the i-th candidate detection, this information is incorporated in the model by encouraging the x_j segment to take class c_i with strength proportional to the average mask values within the segment.

To evaluate the shape prior in isolation (outside the model), we quantify how well the ground truth mask matches the shape prior mask. The pixel-wise accuracy is normalized across foreground and background. Accuracy by chance would be 50%. The accuracy is computed only using pixels that fall in the bounding box. Table 9.2 (top row) shows these results. The accuracy of the above approach "Detector" is 72.7%. If an oracle were to pick the most accurate of the shapes (across the detector components), the accuracy would be 78.5%[5].

We also experiment with alternative shape priors. We resize the binary object masks in the training images to 10×10 pixels. This produces a 100 dimensional vector for each mask. We use K-Means over these vectors to cluster the masks. We set the number of clusters for each category to be equal to the number of detector's components for that category to be comparable to the detector prior. The shape mask for each cluster is the binary mask that is closest to the cluster center. If an oracle were to pick the most accurate of these K masks, it would achieve an accuracy of 75.3% ("Cluster" in Table 9.2), not better than the detector-based oracle above. If we were to pick the shape mask from the training images (without clustering) that matches the ground truth

[5]We asked humans to look at the average shape masks classify them into the object categories. Human performance at this task was 60%.

	Oracle			Automatic				
	Detector	Training Mask	Cluster	Detector	Dist. Tr.	Naive	**Human**	**GT**
Separate	78.5	88.4	75.3	72.7	78.8	74.3	80.2	93.1
Segmentation (Det.)	77.7	78.4	77.5	77.2	76.8	76.3	77.4	80.2
Segmentation (GT)	80.9	82.3	81.1	80.8	81.6	79.5	80.8	84.5
Object Det. (Det.)	46.8	47.6	47.7	46.8	47.4	46.7	47.2	48.5
Object Det. (GT)	95.8	90.3	95.1	93.9	93.3	94.5	96.4	92.7

Table 9.2: Accuracies of different shape priors inside and outside the model. Average recall and Average Precision is reported in the middle two rows and the bottom two rows, respectively.

segmentation of an object the best, we get an oracle accuracy of 88.4% ("Training Mask"). This gives us a sense of the accuracy one can hope to achieve by transferring shape masks from the training data without learning a generalization.

As another prior, we find the training mask whose shape matches the contours within a bounding box the best. We compute edges in the bounding boxes by thresholding the gPb contours. We compute the distance transform of these edges, and identify the training mask whose boundaries fall in regions closest to the edges. This training mask provides the shape prior for the bounding box. This *automatic* approach ("Dist. Tr.") has an accuracy of 78.8% which is comparable to the *oracle* on detector's masks.

Finally, we also experiment with a Naive approach. We simply encourage all segments that lie fully within the bounding box to take the corresponding class-label. The performance is 74.3%, higher than automatically picking the detector mask using the mixture component. This approach shares similarities with the superpixel straddling cue of [ADF10] which assumes that tight bounding boxes around objects do not have a lot of superpixels straddling the bounding box boundaries.

Note that if we "snap" the ground truth segmentation of an object to the segment boundaries i.e. each segment is turned on/off based on whether most of the pixels in the segment are foreground / background in the ground truth, we get an accuracy of 93.1%. This is the upperbound on the performance given the choice of segments.

Human: We showed 5 subjects the segment boundaries in the ground truth object bounding boxes

Figure 9.6: Human object recognition from image boundaries. We show subjects segments inside the object bounding box and ask them to recognize the category of the object. We show the segments with (left image) and without (right image) context.

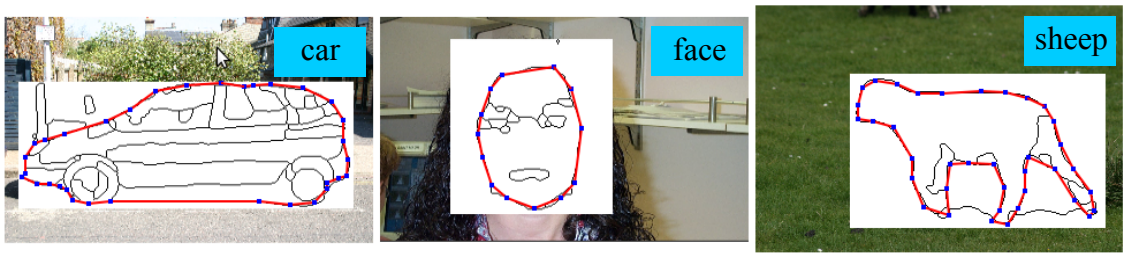


Figure 9.7: Human shape mask labeling interface. Human subjects were asked to draw the object boundaries along the segment contours.

along with its category label and contextual information from the rest of the scene. See Figure 9.7. We showed subject contextual information around the bounding box because without it humans were unable to recognize the object category reliably using only the boundaries of the segments in the box (55% accuracy). With context, classification accuracy was 94%. See Figure 9.6.

Using the interface of [HAB11], subjects were asked to trace a subset of the segment boundaries to match their expected shape of the object. The accuracy of the best of the 5 masks obtained for each object (normalized for foreground and background) was found to be 80.2%. Recall that the best automatic accuracy we obtained with the machine was 78.8% using the distance transform approach, not much worse than the human subjects' accuracy. This shows that humans can not decipher the shape of an object from the UCM segment boundaries much better than an automatic approach. Hence, improved shape analysis simply by "puzzling together" UCM-like segments is unlikely.

Figure 9.8: Human scene classification. Subjects were shown images at multiple resolutions. Subjects were asked to choose one scene category for each image.

9.3.5 Scene and scene-class co-occurrence

Machine: We train a classifier [XHE10] to predict each of the scene types, and use its confidence to form the unitary potential for the scene variable. The scene node connects to each binary class variable z_i via a pairwise potential which is defined based on the co-occurrence statistics of the training data, i.e., likelihood of each class being present for each scene type.

Human: To obtain scene unary, we ask human subjects to classify an image into one of the 21 scene categories used in [YFU12] (see Table 9.1). Images were presented at varying resolutions (i.e., original resolution, smallest dimension rescaled to 32, 24 and 20 pixels) as shown in Figure 9.8. Subjects were allowed to select more than one category when confused, and the potential was computed as the proportion of responses each category got. Human accuracy at scene recognition was 90.4, 89.8, 86.8 and 85.3% for the different resolutions, as compared to the machine accuracy of 81.8%. Note that human performance is not 100% even with full resolution images because the scene categories are semantically ambiguous. Humans clearly outperform the machine at scene recognition, but the question of interest is whether this will translate to improved performance for holistic scene understanding.

Similar to the class-class experiment, to obtain scene-class co-occurrence statistics, subjects were asked which object category is more likely to be present in the scene. We "show" the scene either by naming its category (no visual information), or by showing them the average image for that scene category. Examples are shown in Figure 9.9[6]. The normalized co-occurrence matrix is then used as the pairwise potential.

[6]When asked to look at the average images and recognize the scene category, subjects were 80% accurate.

Figure 9.9: Average scenes for some example scene categories in MSRC.

9.3.6 Ground-truth Potentials

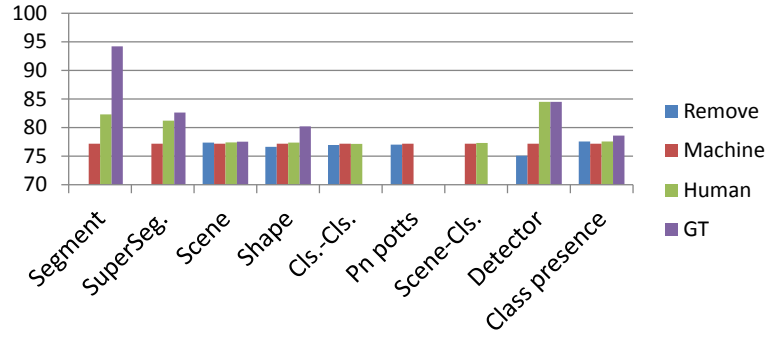
In addition to human potentials (which provide a feasible point), we are also interested in establishing an upper-bound on the effect each subtask can have on segmentation performance by introducing ground truth (GT) potentials into the model. We formed each potential using the dataset annotations. For segments and super-segments we simply set the value of the potential to be 1 for the segment GT label and 0 otherwise, similarly for scene and class unary potentials. For object detection, we used the GT boxes as the candidates and set their detection scores to 1. For the shape prior, we use a binary mask that indicates which pixels inside the GT object bounding box have the object's label.

9.4 Experiments with CRFs

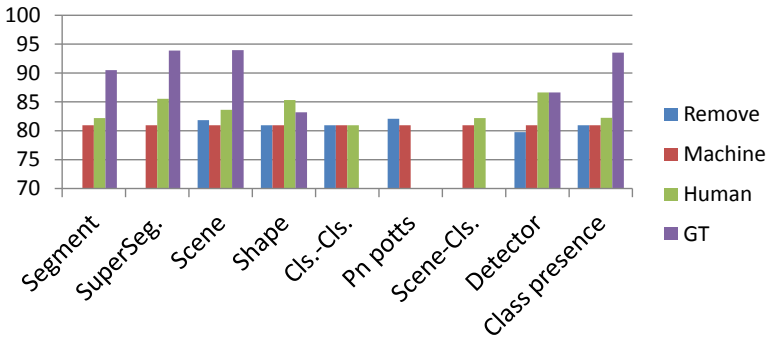
We now describe the results of inserting the human potentials in the CRF model. We also investigated how plugging in GT potentials or discarding certain tasks all together affects performance on the MSRC dataset. For meaningful comparisons, CRF learning and inference is performed every time a potential is replaced, be it with **(i) Human** or **(ii) Machine** or **(iii) GT** or **(iv) Remove**.

A summary of the results for the four different settings is shown in Figure 9.10. Note that in each experiment only a *single* machine potential was replaced, which is indicated in the x axis of the plots. Missing bars for the *remove* setting indicate that removing the corresponding potential would result in the CRF being disconnected, and hence that experiment was not performed. GT is not meaningful for pairwise potentials. The average over all categories is shown on the y axis.

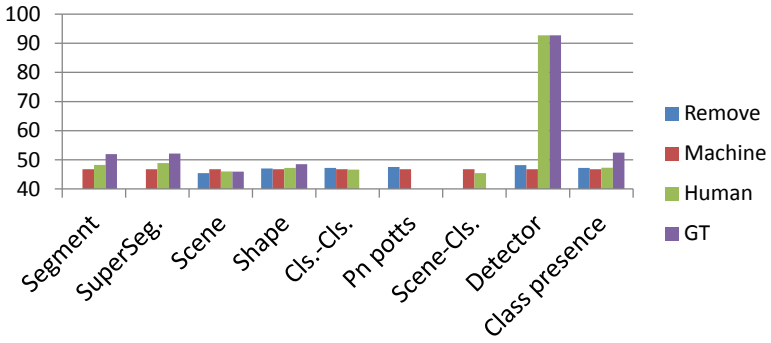
There are several interesting trends. Having GT information for class presence (i.e. knowing



(a) Semantic Segmentation



(b) Scene Classification



(c) Object Detection

Figure 9.10: Impact of each component on machine holistic scene recognition. Here we show the semantic segmentation, object detection, and scene recognition accuracies when a single component of the model is changed (removed, implemented by a machine (default), replaced by a human or replaced with ground truth). The evaluation measures are average per-class recall, Average Precision (AP), and average recall for segmentation, object detection, and scene recognition respectively.

which objects are present in the image) clearly helps scene recognition, but also gives a noticeable boost to object detection and segmentation. This argues in favor of informative classifiers for class presence, which were not used in the current model [YFU12], but is, e.g., done in [GBW10]. Class-class co-occurrence potential and the scene-class potential have negligible impact on the performance of all three tasks. The choice of the scene classifier has little impact on the segmentation but influences detection accuracy. We find that human object detection boosts performance, which is not surprising. GT shape also improves performance, but as discussed earlier, we find that humans are unable to instantiate this potential using the UCM segment boundaries. This makes it unclear what the realizable potential of shape is for the MSRC dataset. One human potential that does improve performance is the unitary segment potential. This is quite striking since human labeling accuracy of segments was substantially worse than machine's (72.2% vs. 77.4%), but incorporating the potential in the model significantly boosts performance (from 77.2% to 82.3%). Intrigued by this, we performed detailed analysis to identify properties of the human potential that are leading to this boost in performance. Resultant insights provided us concrete guidance to improve machine potentials and hence state-of-the-art accuracies.

9.4.1 Analysis of segments

We now describe the various hypotheses we explored including unsuccessful and successful ones to explain the boost provided by human segment potentials.

Scale: We noticed that the machine did not have access to the scale of the segments while humans did. So we added a feature that captured the size of a segment relative to the image and retrained the unary machine potentials. The resultant segmentation accuracy of the CRF was 75.2%, unfortunately worse than the original accuracy at 77.2%.

Over-fitting: The machine segment unaries are trained on the same images as the CRF parameters, potentially leading to over-fitting. Humans obviously do not suffer from such biases. To alleviate any over-fitting in the machine model, we divided the training data into 10 partitions. We trained the machine unaries on 9 parts, and evaluated them on the 10^{th} part, repeating this 10 times. This

gives us machine unaries on the entire training set, which can be used to train the CRF parameters. While the machine unaries may not be exactly calibrated, since the training splits are different by a small fraction of the images, we do not expect this to be a significant issue. The resultant accuracy was 76.5%, again, not an improvement.

Ranking of the correct label: It is clear that the highest ranked label of the human potential is wrong more often than the highest ranked label of the machine potential (hence the lower accuracy of the former outside the model). But we wondered if perhaps even when wrong, the human potential gave a high enough score to the correct label making it revivable when used in the CRF, while the machine was more "blatantly" wrong. We found that among the misclassified segments, the rank of the correct label using human potentials was 4.59 – better than 6.19 (out of 21) by the machine.

Uniform potentials for small segments: Recall that we did not have human subjects label the segments smaller than 500 pixels and assigned a uniform potential to those segments. The machine on the other hand produced a potential for each segment. We suspected that ignoring the small (likely to be misclassified) segments may give the human potential an advantage in the model. So we replaced the machine potentials for small segments with a uniform distribution over the categories. The average accuracy unfortunately dropped to 76.5%. As a follow-up, we also weighted the machine potentials by the size of the corresponding segment. The segmentation accuracy was still 77.1%, similar to the original 77.2%.

Regressing to human potentials: We then attempted to directly regress from the machine potential as well as the segment features (TextonBoost, LBP, SIFT, ColorSIFT, location and scale) to the human potential, with the hope that if for each segment, we can predict the human potential, we may be able to reproduce the high performance. We used Gaussian Process regression with an RBF kernel. The average accuracy in both cases was lower: 75.6% and 76.5%. We also replicated the sparsity of human potentials in the machine potentials, but this did not improve performance by much (77.3%).

Complementarity: To get a deeper understanding as to why human segment potentials significantly increase performance when used in the model, we performed a variety of additional hybrid

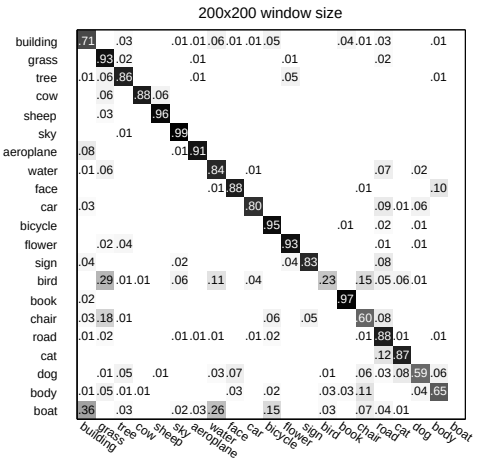
CRF experiments. These included having human (H) or machine (M) potentials for segments (S) or super-segments (SS) or both, with or without the P^n potential in the model. The results are shown in Table 9.3. The last two rows correspond to the case where both human and machine segment potentials are used together at the same level. In this case, using a P^n potential or not has little impact on the accuracy. But when the human and machine potentials are placed at different levels in the model (rows 3 and 4), not having a P^n potential (and thus loosing connection between the two levels) significantly hurts performance. This indicates that even though human potentials are not significantly more accurate than machine potentials, when both human and machine potentials interact, there is a significant boost in performance, demonstrating the complementary nature of the two.

	P^n	without P^n
H S, H SS	78.9	77.2
M S, M SS	77.2	77.0
H S, M SS	82.3	75.3
M S, H SS	81.2	78.2
H S+M S, M SS	80.9	81.3
H S+M S, H SS	82.3	82.8

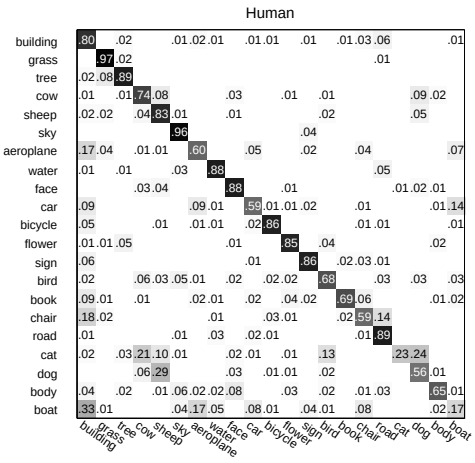
Table 9.3: Human and machine segment potentials are complementary. The last two rows correspond to the case where both human and machine segment potentials are used together at the same level. In this case, using a P^n potential or not has little impact on the accuracy. But when the human and machine potentials are placed at different levels in the model (rows 3 and 4), not having a P^n potential (and thus loosing connection between the two levels) significantly hurts performance. This indicates that even though human potentials are not significantly more accurate than machine potentials, when both human and machine potentials interact, there is a significant boost in performance, demonstrating the complimentary nature of the two.

Therefore, we hypothesized that the types of mistakes that the machine and humans make may be different. Our initial analysis showed that humans are generally better at detecting stuff while machine is better recognizing things (Figure 9.11(d)).

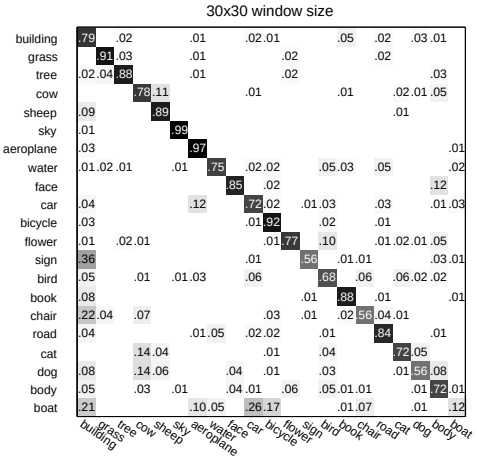
Additionally, we qualitatively analyzed the confusion matrices for both Figure 9.11. We noticed



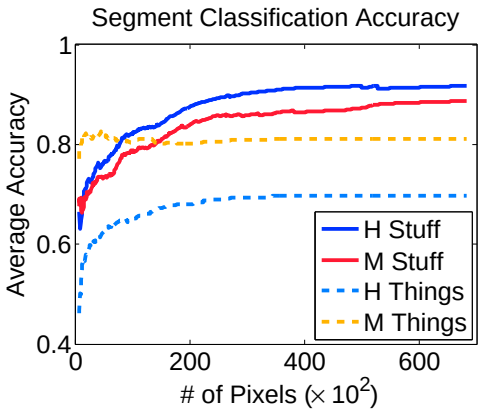
(a) Machine classification with 200x200 windows



(b) Human classification



(c) Machine classification with 30x30 windows



(d)

Figure 9.11: The confusion matrices for segment classification are shown. For machine with large window size (a), there is high confusion between classes appearing in the surrounding area of each other, for example, bird-grass, car-road, etc. The types of mistakes are different for humans (b). They confused objects that look similar, for instance, there is confusion between cat-cow or boat-aeroplane. When we reduce the window size for machine (c), the mistakes become more similar to the human mistakes. Combining the small-window machine potentials with the large-window machine potentials results in a significant improvement in segmentation accuracy, resulting in state-of-the-art performance on the MSRC dataset. (d) Humans (H) and machines (M) with 200×200 window have different performance for recognizing stuff and things segments. Humans are generally better at recognizing stuff, while machines are better at things recognition. Larger segments are generally easier to recognize.

that the machine confuses categories that spatially surround each other *e.g.*bird and grass or car and road. This was also observed in [SWR09]. This is understandable because Textonboost uses a large (200 $\times$ 200) window surrounding a pixel to generate its feature descriptor. On the other hand, human mistakes are between visually similar categories *e.g.*car and boat.[7] Hence, we trained Textonboost with smaller windows. We re-computed the segment unaries and plugged them into the model in addition to the original unaries that used large windows. The average accuracy using window sizes of 10, 20, 30 and 40 are shown in Table 9.4. This improvement of 2.4% over state-of-the-art is quite significant for this dataset[8] Notice that the improvement provided by the entire CRF model over the original machine unaries *alone* was 3% (from 74.2% to 77.2%). While a fairly straightforward change in the training of machine unaries lead to this improvement in performance, we note that the insight to do so was provided by our use of humans to "debug" the state-of-the-art model.

Textonboost Window Size	[YFU12] (200×200)	10×10	20×20	30×30	40×40
Average per-class Recall	77.2	77.9	78.5	79.6	79.6

Table 9.4: Segmentation accuracy obtained by resizing Textonboost window size. Decreasing the window size makes the machine errors similar to human errors in the task of isolated segment classification.

9.4.2 Analysis of Shape Prior

In section 9.3, we explored various types of shape priors outside the machine model. We now evaluate the impact of the various shape potentials when used in the full model. The second two rows in Table 9.2 provide the semantic segmentation accuracy, while bottom two rows correspond

[7] One consequence of this is that the mistakes made within a super-segment are consistent for machines but variable for humans. Specifically, on average machine assigns different labels to 4.9% of segments, while humans assign different labels to 12% of the segments within a super-segment. The consistent mistakes may be harder for other components in the CRF to fix.

[8] Adding a new unary potential simply by incorporating a different set of features and kernels than Textonboost (such as color, SIFT and self-similarity with intersection kernel) provides only a small boost at best (77.9%).

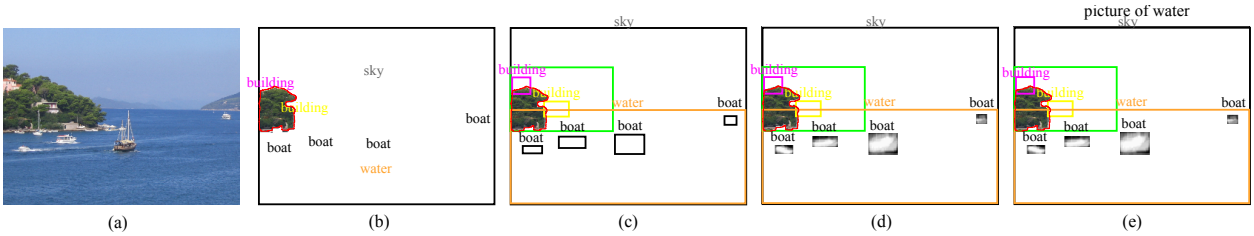


Figure 9.12: (b):(e) Human-study interfaces for (a) with increasingly more contextual information.

to object detection accuracy. The bottom row in both pairs of rows corresponds to using the ground truth bounding boxes around an object, while the top rows correspond to using an object detector to automatically determine the object bounding boxes.

The improvement in performance we obtained over the detector induced prior used in [YFU12] via our distance transform over contour detection approach outside the model (Table 9.2 top row) did not translate into an improvement in the performance of the overall model. Further analysis into this revealed that while the normalized binary segmentation accuracy for our approach was better, the unnormalized accuracy was slightly worse, which corresponds better to the metric the model is trained to optimize. While GT shape can provide significant improvement in conjunction with GT bounding boxes for objects, we find that human subjects were not able to realize this potential in terms of segmentation accuracy. Interestingly, for object detection, using human shape on ground truth detection outperforms using ground truth shape on ground truth detection.

9.5 Analyzing the pipeline

In this section, we analyze the holistic scene understanding CRF model itself. First, we investigate whether the components used in the model are beneficial for humans. Second, we estimate the potential that the model as a whole holds for all the three tasks.

9.5.1 Contextual Image Labeling

To study if humans benefit from contextual information provided by the model in order to recognize local regions, we design an experiment where, in addition to the segment pixels, we show subjects

progressively more information from the model: presence / absence of each class, object bounding boxes, shape prior masks and scene type. We selected 50 random segments of any size from each category and asked the subjects to classify each segment. Fig. 9.12 shows an example for each of the interfaces used for this study. From left to right (b – e), the contextual information available to the subjects is increased.

The results are shown in Table 9.5. We see that human performance in the task of image labeling significantly increases when presented with the same contextual information used by holistic machine models. We also show accuracies of the machine model when given access to precisely the same information as shown to humans. We find that humans can effectively leverage the available information. Even though human performance is worse than machines when viewing segments in isolation, they outperform the machine when given access to the contextual information modeled by the CRF. This is especially true for "things". Access to context seem to confuse human responses for "stuff".

	build.	grass	tree	cow	sheep	sky	aeropl.	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**avg.**	**avg. stuff**	**avg. things**
	Humans																							
Segm. (S)	48	82	38	20	14	**64**	18	60	50	22	12	58	44	8	20	16	36	0	6	22	6	30.7	54.7	21.1
S+label	**78**	**84**	**54**	86	82	58	64	62	72	50	72	90	92	88	90	90	**52**	92	82	58	70	74.6	**64.7**	78.5
S+box (B)	58	72	38	**100**	98	44	**100**	54	82	96	**94**	**100**	**96**	**96**	**98**	96	46	**100**	**98**	92	98	83.6	52.0	96.3
S+B+msk	58	72	44	**100**	**100**	42	**100**	54	98	**98**	**94**	**100**	**96**	94	**98**	**98**	36	**100**	**98**	**96**	**100**	84.6	51.0	**98.0**
Full info.	68	78	48	98	**100**	50	**100**	**66**	**100**	**98**	**94**	**100**	**96**	94	**98**	**98**	46	**100**	**98**	90	**100**	**86.7**	59.3	97.6
	Machine model																							
Segm. (S)	82	86	93	74	94	96	84	88	96	70	90	88	80	6	97	30	92	95	51	34	0	72.7	89.5	65.9
S+label	83	87	93	76	92	97	87	92	96	73	100	98	80	42	97	54	93	96	70	33	0	78.2	90.8	72.9
S+box (B)	83	87	93	86	96	97	90	92	96	82	100	100	87	44	100	61	93	96	71	35	55	82.9	90.8	79.9
S+B+msk	84	87	93	86	98	97	90	92	**97**	82	100	100	87	44	100	61	93	96	71	42	75	84.3	**91.0**	**81.9**
Full info.	**84**	**87**	**93**	**86**	**98**	**97**	**90**	**92**	96	**82**	**100**	**100**	**87**	**44**	**100**	**61**	**93**	**96**	**71**	**42**	**75**	**84.4**	**91.0**	**81.9**

Table 9.5: Human segmentation accuracies with increasing information from the model

9.5.2 Journey to Perfection

To analyze the potential that the model holds to reach perfection we conduct experiments using different combinations of machine, human, and ground truth potentials. Fig. 9.13(a) provides a journey on the segmentation task from the machine's current 77% performance to perfection. We find that incorporating human (H) segment (S) potentials improves performance by 5-6%. Incorporating ground truth (GT) detection (Det) provides an improvement of 4-7%. Adding in GT for remaining tasks (except super-segments (SS)) further improves performance by 2-5%. These combined bring us to 92%. GT segments plugged into the model perform at 94.2%, which outside the model are at 94.5% (the upper bound on the performance of the model since it makes segment-level decisions). This shows that as far as segmentation goes the model itself is sufficient and all the required tasks are being modeled. This analysis also provides concrete guidance for future research efforts: designing representations complementary to the ones used in the model, perhaps by mimicking human responses, has potential for significant improvement. And of course, improving on all the tasks would lead to more effective holistic scene understanding.

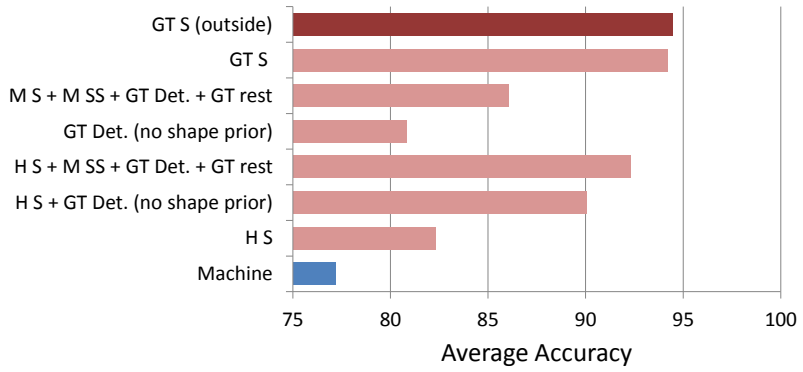
Fig. 9.13(c) shows the effect of each component on the object detection task. The machine's average precision is 46.8. Including the ground truth information for class presence (CP) improves the average precision to 52.5 (5.7 improvement in AP). Incorporating human segment (S) and super-segment (SS) potentials provides an additional improvement of 1.1 AP, which leads to 53.6 AP. Ground truth shape information also provides improvement for the object detection task, and increases the AP to 54.3. Including the ground truth scene has negligible impact on performance. If we replace the human segment and super-segment potentials by their ground truth counterparts, the average precision decreases to 54.0. Hence, ground truth class-presence (knowing whether a certain category exists in the image or not similar to the image classification task in PASCAL) is the single component that provides the biggest boost in performance. Obviously, using ground truth information for object detection has a significant effect on the performance, where it improves the AP to 93.9. Hence, given the scope of this model, the burden of improving the detection performance from 54.3 to 93.9 lies on the detector itself. Enhancing the model with additional cues such as a rough 3D layout of the scene, etc. that directly influence likely locations and scales

of detections may be important to aid the detector.

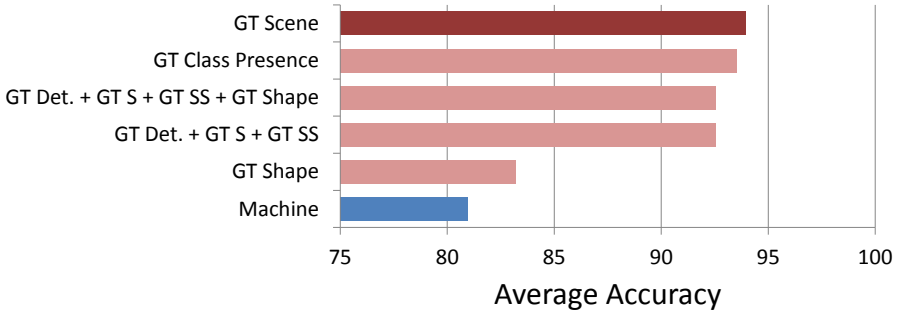
The journey for scene recognition is shown in Fig. 9.13(b). The machine performance for scene recognition is 81.0%. Using ground truth shape potential improves the performance by 2.2%. Using ground truth detection (Det), segment (S) and super-segment (SS) potentials instead provides 9.3% boost in accuracy. Adding ground truth shape to this combination does not change the accuracy. Using ground truth class-presence single-handedly takes the performance to 93.5% (an improvement of 12.5%, while ground truth scene information is at 94.0%). This is because the scene categories in this dataset as defined in Table 9.1 are object-category centric, and hence knowing whether a certain object-category is present in an image or not provides strong cues about the scene category.

9.6 Conclusion

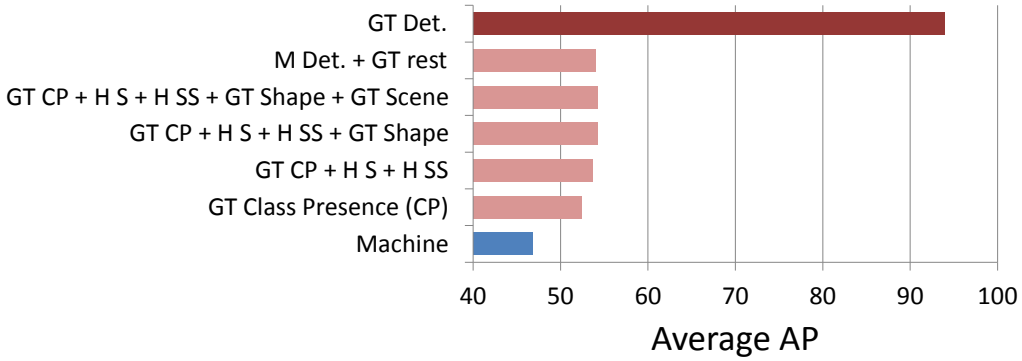
Researchers have developed sophisticated machinery for holistic scene understanding. Insights into which aspects of these models are crucial, especially for further improving state-of-the-art performance is valuable. We gather these insights by analyzing a state-of-the-art CRF model for holistic scene understanding on the MSRC dataset. Our analysis hinges on the use of human subjects to produce the different potentials in the model. Comparing performance of various human-machine hybrid models allows us to identify the components of the model that still have "head room" for improving performance. One of our findings was that human responses to local segments in isolation, while being less accurate than machines', provide complementary information that the CRF model can effectively exploit. We explored various avenues to precisely characterize this complementary nature, which resulted in a novel machine potential that significantly improves accuracy over the state-of-art. We also investigated different shape priors for the model, and it turned out that human subjects can not decipher the object shape from super-pixel boundaries any better than machines. In addition, we showed that humans can effectively leverage the contextual information incorporated in the machine model. We expect even more insightful findings if this model is studied on larger and more challenging datasets.



(a) Semantic Segmentation



(b) Scene Classification



(c) Object Detection

Figure 9.13: Journey to perfection for segmentation, scene classification, and object detection.

CHAPTER 10

Conclusion

We proposed a series of novel methods to tackle the challenging task of scene understanding. The scene understanding problem can be decomposed into several different sub-problems. Our focus in this thesis was on three highly correlated sub-problems namely object detection, segmentation, and contextual reasoning. We proposed variants of flat and hierarchical compositional models for object detection. Our goal was to design object models such that learning and inference can be performed efficiently for a large number of categories. We also analyzed the complexity of compositional models. We believe there is no such analysis for other object models in the computer vision literature.

Another challenge in object detection is to model deformable objects. We proposed novel approaches for modeling highly flexible objects. First, we proposed a method that augmented the reliably detectable part of an object [1] with irregular shaped patches to better capture deformations. Furthermore, we proposed another method that combined segments and object templates. The idea was that a fixed template is not ideal for capturing deformations, and deformations can be better captured with the segments. However, segmentation methods are not perfect and sometimes they under or over segment objects. Our method was able to recover from these mistakes and automatically learned which model (segments or template) is more reliable for a particular category. We achieved the state-of-the-art performance using these approaches. These methods infer bounding box or silhouettes for objects. We also proposed a method to provide richer descriptions for objects in terms of semantic parts. We not only showed significant gain over the state-of-the-art (our method mentioned previously) but also parsed the objects into their semantic parts.

[1] For example, head of a cat can be detected more reliably than its body since it is more rigid.

We additionally showed that contextual reasoning plays an important role for object detection and scene understanding. We proposed a method that employed contextual information and provided significant boost for detection of small objects, where there are not strong appearance cues. Finally, we proposed a series of hybrid human-machine experiments to find the bottleneck in scene understanding methods. We replaced different components of a state-of-the-art scene understanding model by humans and our goal was to find out which task, if improved, can boost the segmentation performance significantly. We modified our model according to the results of our human studies, which improved the performance significantly.

Below, we discuss a number of future directions:

Scaling up detectors: One of the main limitations of current object detectors is that they are able to handle only a limited number of object categories. Moreover, these methods are far from meeting real-time requirements. One direction to pursue is to develop better models for representing objects so learning and inference can be performed much faster. We have had an initial step towards that direction in chapters 2–4 and proposed a compositional learning approach. These methods are still in early experimental stages, but they have shown to be promising.

Learning with active observers: One approach to resolve ambiguities in learning object models or inference using them is to incorporate an active observer (humans, robots, etc.) in the loop. The learning procedure can produce feedback for the active observer based on its current belief about the structure of the scene. So it can ask the observer to take actions such that learning becomes 'easier'. For instance, sometimes reasoning from a single viewpoint is not possible. However, the inference method can request the observer to move to certain locations so it can boost or weaken its hypothesis about an entity in the scene seen from a particular viewpoint before.

Learning interpretable models: Most of the state-of-the-art object detection methods are formulated within a discriminative framework and so learn parts that are most effective for discriminating the object from the background clutter. Hence, the learned parts usually do not correspond to meaningful (semantic) structures, and the models are not *interpretable*. In contrast, we should develop methods that provide richer descriptions (e.g., semantic parts or localized attributes) for objects since other tasks such as pose estimation or activity recognition rely on these types of

representation.

Beyond 2D appearance: Using only 2D appearance cues makes the object detection task unnecessarily difficult. For instance, reasoning about occlusion according to 2D cues is almost impossible in some cases. Other cues such as camera motion or 3D structure of the scene potentially make the task much easier. Nevertheless, reasoning about 3D information is computationally expensive and also modeling non-rigid parts of a scene is not trivial. A hybrid 2D-3D model seems a good choice for this application, where the method can be adaptive and choose to use additional 3D information or rely completely on 2D cues.

REFERENCES

[ADF10] B. Alexe, T. Deselaers, and V. Ferrari. "What is an object?" In *CVPR*, 2010. 136

[AHG12] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. "Semantic Segmentation using Regions and Parts." In *CVPR*, 2012. 76

[AHT12] B. Alexe, N. Heess, Y. W. Teh, and V. Ferrari. "Searching for objects driven by context." In *NIPS*, 2012. 108

[AL12] H. Azizpour and I. Laptev. "Object detection using strongly-supervised deformable part models." In *ECCV*, 2012. 77, 92, 93, 99, 100, 101

[AMF11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour Detection and Hierarchical Image Segmentation." In *PAMI*, 2011. 117, 129

[AT08] N. Ahuja and S. Todorovic. "Connected Segmentation Tree A Joint Representation of Region Layout and Hierarchy." In *CVPR*, 2008. 8

[AW03] N. J. Adams and C. K. I. Williams. "Dynamic Trees for Image Modelling." *Image and Vision Computing*, **20**(10):865–877, 2003. 22

[Bac91] T. Bachmann. "Identification of spatially quantized tachistoscopic images of faces: How many pixels does it take to carry identity?" *Europ. J. of Cogn. Psych.*, 1991. 125

[BBM11] T. Brox, L. Bourdev, S. Maji, and J. Malik. "Object Segmentation by Alignment of Poselet Activations to Image Contours." In *CVPR*, 2011. 76, 124

[BD07] F. Brendan and D. Delbert. "Clustering by Passing Messages Between Data Points." *Science*, **315**(972–976), 2007. 51

[Bie87] I. Biederman. "Recognition-by-components: a theory of human image understanding." *Psychological Review*, **94**(2):115–147, April 1987. 21

[BM09] L. Bourdev and J. Malik. "Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations." In *ICCV*, 2009. 58, 93, 100, 101

[BMB10] L. Bourdev, S. Maji, T. Brox, and J. Malik. "Detecting People Using Mutually Consistent Poselet Activations." In *ECCV*, 2010. 77, 78

[BMR82] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. "Scene perception: Detecting and judging objects undergoing relational violations." *Cognitive Psychology*, 1982. 107

[BPB11] S. Branson, P. Perona, and S. Belongie. "Strong Supervision From Weak Annotation: Interactive Training of Deformable Part Models." In *ICCV*, 2011. 93

[BSF08] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. "Segmentation and recognition using structure from motion point clouds." In *ECCV*, 2008. 109, 128

[BSI08] O. Boiman, E. Shechtman, and M. Irani. "In defense of Nearest-Neighbor based image classification." In *CVPR*, 2008. 14

[BT78] H. Barrow and J. Tenenbaum. "Recovering intrinsic scene characteristics from images." *In Comp. Vision Systems*, 1978. 124

[BU02] E. Borenstein and S. Ullman. "Class-specific, top-down segmentation." *ECCV 2002*, pp. 639–641, 2002. 22

[BYV11] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. "Kernelized structural svm learning for supervised object segmentation." In *CVPR*, 2011. 78

[BZM07] A. Bosch, A. Zisserman, and X. Munoz. "Image classifcation using random forests and ferns." In *ICCV*, 2007. 43, 44

[CCB12] J. Carreira, R. Caseiroa, J. Batista, and C. Sminchisescu. "Semantic Segmentation with Second-Order Pooling." In *ECCV*, 2012. 76, 85, 87, 116, 117, 118

[CFB04] P. Carbonetto, N. de Freitas, and K. Barnard. "A Statistical Model for General Contextual Object Recognition." In *ECCV*, pp. 350–362, 2004. 108

[CFH05] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. "Spatial priors for part-based recognition using statistical models." In *CVPR*, 2005. 46

[CH06] D. J. Crandall and D. P. Huttenlocher. "Weakly Supervised Learning of Part-Based Spatial Models for Visual Object Recognition." In *ECCV*, pp. I: 16–29, 2006. 92

[CH07] D. Crandall and D. Huttenlocher. "Composite Models of Objects and Scenes for Category Recognition." In *CVPR*, 2007. 42, 50, 58

[CL68] C. K. Chow and C. N Liu. "Approximating discrete probability distributions with dependence trees." *IEEE Transactions on Information Theory*, **14**(3):462467, 1968. 132

[CLS12] J. Carreira, Fuxin Li, and Cristian Sminchisescu. "Object Recognition by Sequential Figure-Ground Ranking." *IJCV*, **98**(3):243–262, 2012. 4, 76, 86, 87, 101, 109

[CLT10] M. Choi, J. Lim, A. Torralba, and A. S. Willsky. "Exploiting Hierarchical Context on a Large Database of Object Categories." In *CVPR*, 2010. 108

[CSH12] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan. "Hierarchical Matching with Side Information for Image Classification." In *CVPR*, 2012. 86, 87

[CZY09] Y. Chen, L. Zhu, A. Yuille, and H. Zhang. "Unsupervised Learning of Probabilistic Object Models (POMs) for Object Classification, Segmentation and Recognition using Knowledge Propagation." *PAMI*, **31**(10), 2009. 53

[CZY10] Y. Chen, L. Zhu, and A. Yuille. "Active mask hierarchies for object detection." In *ECCV*, 2010. 58, 64, 71, 72, 77

[DBB08] P. Dollár, B. Babenko, S. Belongie, P. Perona, and Z. Tu. "Multiple Component Learning for Object Detection." In *ECCV*, 2008. 92

[DH12] Q. Dai and D. Hoiem. "Learning to Localize Detected Objects." In *CVPR*, 2012. 78

[DHH09] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. "An Empirical Study of Context in Object Detection." In *CVPR*, 2009. 108

[DRF09] C. Desai, D. Ramanan, and C. Fowlkes. "Discriminative Models for Multi-Class Object Layout." In *ICCV*, 2009. 77

[DT05] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." In *CVPR*, 2005. 42, 43, 77, 85, 86, 90, 119

[DZR12] J. DiCarlo, D. Zoccolan, and N. Rust. "How Does the Brain Solve Visual Object Recognition?" *Neuron*, **73**(3):415–434, February 2012. 41

[EKS96] M. Ester, H. Kriegel, J. Sander, and X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996. 51

[FBL08] S. Fidler, M. Boben, and A. Leonardis. "Similarity-based cross-layered hierarchical representation for object categorization." In *CVPR*, 2008. 9

[FCH08] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. "LIBLINEAR: A library for large linear classification." *JMLR*, 2008. 97

[FEH10] A. Farhadi, I. Endres, and D. Hoiem. "Attribute-Centric Recognition for Cross-category Generalization." In *CVPR*, 2010. 65

[FFJ08] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. "Groups of Adjacent Contour Segments for Object Detection." *PAMI*, **30**(1):36–51, 2008. 53, 54, 56

[FGM10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. "Object Detection with Discriminatively Trained Part Based Models." *PAMI*, 2010. 3, 8, 42, 44, 54, 56, 58, 68, 71, 72, 73, 76, 77, 79, 80, 84, 85, 86, 90, 91, 92, 93, 96, 99, 101, 102, 115, 116, 117, 119, 122, 127, 133

[FH05] P. Felzenszwalb and D. Huttenlocher. "Pictorial Structures for Object Recognition." *IJCV*, **61**(1), 2005. 58, 92

[FJS09] V. Ferrari, F. Jurie, and C. Schmid. "From Images to Shape Models for Object Detection." *IJCV*, **87**(3), 2009. 53, 54, 56

[FL07] S. Fidler and A. Leonardis. "Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts." In *CVPR*, 2007. 58

[Fow05] C. C. Fowlkes. "Measuring the Ecological Validity of Grouping and Figure-Ground Cues." *Thesis*, 2005. 125

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. "Object Class Recognition by Unsupervised Scale-Invariant Learning." In *CVPR*, 2003. 8, 49

[Fuk88] K. Fukushima. "Neocognitron - a Hierarchical Neural Network Capable of Visual-Pattern Recognition." *Neural Networks*, **1**(2):119–130, 1988. 21

[FVK02] L. Fei-Fei, R. VanRullen, C. Koch, and P. Perona. "Rapid natural scene categorization in the near absence of attention." *PNAS*, 2002. 125

[GAL12] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik. "Multi-Component Models for Object Detection." In *ECCV*, 2012. 86, 93, 100, 101

[GBW10] J. Gonfaus, X. Boix, J. van de Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. "Harmony Potentials for Joint Classification and Segmetation." In *CVPR*, 2010. 59, 76, 109, 125, 126, 141

[GFM11] R. Girshick, P. Felzenszwalb, and D. McAllester. "Object Detection with Grammar Models." In *NIPS*, 2011. 59, 84, 93, 116

[GGK09] S. Gould, T. Gao, and D. Koller. "Region-based Segmentation and Object Detection." In *NIPS*, 2009. 60, 122, 125

[GLA09] C. Gu, J.J. Lim, P. Arbelaez, and J. Malik. "Recognition using Regions." In *CVPR*, 2009. 78, 124

[GN09] P. Gehler and S. Nowozin. "On Feature Combination for Multiclass Object Classification." In *ICCV*, 2009. 63

[GPC02] S. Geman, D. F. Potter, and Z. Chi. "Composition systems." *Quarterly of Applied Mathematics*, **60**(4):707–736, 2002. 22

[HAB11] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. "Semantic Contours from Inverse Detectors." In *ICCV*, 2011. 73, 108, 137

[HCD12] D. Hoiem, Y. Chodpathumwan, and Q. Dai. "Diagnosing Error in Object Detectors." In *ECCV*, 2012. 102, 119

[HE08] D. Hoiem, A. A. Efros, , and M. Hebert. "Closing the loop on scene interpretation." In *CVPR*, 2008. 124

[HEH05] D. Hoiem, A. A. Efros, and M. Hebert. "Geometric Context from a Single Image." In *ICCV*, 2005. 108

[HGS08] G. Heitz, S. Gould, A. Saxena, and D. Koller. "Cascaded Classification Models: Combining Models for Holistic Scene Understanding." In *NIPS*, 2008. 76, 122, 124

[HGW74] H. Hock, G. P. Gordon, and R. Whitehurst. "Contextual relations: The influence of familiarity, physical plausibility, and belongingness." In *Perception & Psychophysics*, 1974. 107

[HK08] G. Heitz and D. Koller. "Learning spatial context: Using stuff to find things." In *ECCV*, 2008. 108

[HOT06] G. E. Hinton, S. Osindero, and Y. Teh. "A fast learning algorithm for deep belief nets." *Neural Computation*, **18**:1527–54, 2006. 22

[HZ10] W. Hu and S. C. Zhu. "Learning a probabilistic model mixing 3D and 2D primitives for view invariant Object Recognition." In *CVPR*, 2010. 55, 56

[JKG08] P. Jain, B. Kulis, and K. Grauman. "Fast image search for learned metrics." In *CVPR*, 2008. 14

[JKL09] K. Jarrett, K. Kavukcuoglu, and Y. Lecun. "What is the Best Multi-Stage Architecture for Object Recognition?" In *ICCV*, 2009. 14

[KDH10] L. Karlinsky, M. Dinerstein, D. Harari, and S. Ullman. "The chains model for detecting parts by their context." In *CVPR*, 2010. 92

[Ker87] D. Kersten. "Predictability and redundancy of natural images." *JOSA A*, **4**(12):2395–2400, 1987. 21

[KH04] S. Kumar and M. Hebert. "Multiclass Discriminative Fields for Parts-Based Object Detection." In *Snowbird Learning Workshop*, 2004. 92

[KH05] S. Kumar and M. Hebert. "A Hierarchical Field Framework for Unified Context-Based Classification." In *ICCV*, 2005. 124

[KK11] P. Krähenbühl and V. Koltun. "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials." In *NIPS*, 2011. 76, 78

[KKT07] P. Kohli, M. Pawan Kumar, and Philip H. S. Torr. "P^3 and Beyond: Solving Energies with Higher Order Cliquess." In *CVPR*, 2007. 129

[Kol06] V. Kolmogorov. "Convergent tree-reweighted message passing for energy minimization." *PAMI*, 2006. 68

[KTZ05] M. P. Kumar, P. H. S. Torr, and A. Zisserman. "OBJ CUT." In *CVPR*, 2005. 59

[KY11] I. Kokkinos and A. Yuille. "Inference and Learning with Hierarchical Shape Models." *IJCV*, **93**(2):201–225, June 2011. 22

[LB95] Y. LeCun and Y. Bengio. "Convolutional Networks for Images, Speech, and Time-Series." In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995. 22

[Len98] P. Lennie. "Single units and visual cortical organization." *Perception*, **27**:889–935, 1998. 39

[LG10] Y. J. Lee and K. Grauman. "Object-Graphs for Context-Aware Category Discovery." In *CVPR*, 2010. 108

[LGR09] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In *ICML*, pp. 609–616, 2009. 9, 14, 17, 19

[LJ08] D. Larlus and F. Jurie. "Combining Appearance Models and Markov Random Fields for Category Level Object Segmentation." In *CVPR*, 2008. 59

[LKR09] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. "Image Segmentation with A Bounding Box Prior." In *ICCV*, 2009. 76, 124

[LKS10] C. Li, A. Kowdle, A. Saxena, and T. Chen. "Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models." In *NIPS*, 2010. 109, 122, 124

[LLS08] B. Leibe, A. Leonardis, and B. Schiele. "Robust Object Detection with Interleaved Categorization and Segmentation." *IJCV*, **77**(1):259–289, 2008. 8, 18

[LM03] T. S. Lee and D. B. Mumford. "Hierarchical Bayesian inference in the visual cortex." *Journal of the Optical Society of America A, Optics, Image Science, and Vision*, **20**(7):1434–1448, July 2003. 23

[LRK09] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. "Associative hierarchical CRFs for object class image segmentation." In *ICCV*, 2009. 59, 126, 129

[LRK10] L. Ladicky, C. Russell, P. Kohli, and P. H.S. Torr. "Graph Cut based Inference with Co-occurrence Statistics." In *ECCV*, 2010. 76, 109

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems." *Journal of the Royal Statistical Society.*, **50**(2):157–224, 1988. 11

[LSA10] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. H.S. Torr. "What, Where & How Many? Combining Object Detectors and CRFs." In *ECCV*, 2010. 60, 78, 125, 126, 128

[LYT09] C. Liu, J. Yuen, and A. Torralba. "Nonparametric scene parsing: label transfer via dense scene alignment." In *CVPR*, 2009. 128

[LYT11] C. Liu, J. Yuen, and A. Torralba. "SIFT Flow: Dense Correspondence across Scenes and its Applications." *TPAMI*, **33**(5):978–994, 2011. 109

[MA07] T. Malisiewicz and A. A.Efros. "Improving Spatial Support for Objects via Multiple Segmentations." In *BMVC*, 2007. 128

[MO12] A. Monroy and B. Ommer. "Beyond Bounding-Boxes: Learning Object Shape by Model-Driven Grouping." In *ECCV*, 2012. 78

[MYP11] M. Maire, S. X. Yu, and P. Perona. "Object Detection and Segmentation from Joint Embedding of Parts and Pixels." In *ICCV*, 2011. 78

[OB06] B. Ommer and J. M. Buhmann. "Learning Compositional Categorization Models." In *ECCV*, 2006. 14

[OS00] A. Oliva and P. G. Schyns. "Diagnostic colors mediate scene recognition." *Cognitive Psychology*, 2000. 125

[Par11] D. Parikh. "Recognizing Jumbled Images: The Role of Local and Global Information in Image Classification." In *CVPR*, 2011. 125

[PBG11] S. Park, T. F. Brady, M. R. Greene, and A. Oliva. "Disentangling scene content from its spatial boundary: Complementary roles for the PPA and LOC in representing real-world scenes." *Journal of Neuroscience*, 2011. 122

[PCI08] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases." In *CVPR*, 2008. 61

[PD10] H. Poon and P. Domingos. "Sum-Product Networks: A New Deep Architecture." In *Uncertainty in Artificial Intelligence (UAI)*, 2010. 22

[PRF10] D. Park, D. Ramanan, and C. Fowlkes. "Multiresolution models for object detection." In *ECCV*, 2010. 93, 102, 108

[PVG11] M. Pedersoli, A. Vedaldi, and J. Gonzlez. "A coarse-to-fine approach for fast deformable object detection." In *CVPR*, 2011. 78

[PVJ11] O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. "The Truth About Cats and Dogs." In *ICCV*, 2011. 59, 78, 100, 101

[PZ10] D. Parikh and C. L. Zitnick. "The Role of Features, Algorithms and Data in Visual Recognition." In *CVPR*, 2010. 125

[PZ11] D. Parikh and C.L. Zitnick. "Finding the weakest link in person detectors." In *CVPR*, 2011. 125

[PZC11] D. Parikh, C. L. Zitnick, and T. Chen. "Exploring Tiny Images: The Roles of Appearance and Contextual Information for Machine and Human Object Recognition." *PAMI*, 2011. 107

[RC96] J. Rivest and P. Cabanagh. "Localizing contours defined by more than one attribute." *Vision Research*, 1996. 125

[RP99] M. Riesenhuber and T. Poggio. "Hierarchical Models of Object Recognition in Cortex." *Nature Neuroscience*, **2**:1019–1025, 1999. 22

[RTM08] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. "LabelMe: a database and web-based tool for image annotation." *IJCV*, **77**(1-3):157–173, 2008. 110

[RVG07] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. "Objects in Context." In *ICCV*, 2007. 109, 122, 124

[SCH11] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. "Contextualizing Object Detection and Classification." In *CVPR*, 2011. 71, 72

[SFR09] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele. "Discriminative structure learning of hierarchical representations for object detection." In *CVPR*, 2009. 9

[SGS10] M. Stark, M. Goesele, and B. Schiele. "Back to the Future: Learning Shape Models from 3D CAD Data." In *BMVC*, 2010. 55

[SJC08] J. Shotton, M. Johnson, and R. Cipolla. "Semantic Texton Forests for Image Categorization and Segmentation." In *CVPR*, 2008. 128

[SKH12] N. Silberman, P. Kohli, D. Hoiem, and R. Fergus. "Indoor Segmentation and Support Inference from RGBD Images." In *ECCV*, 2012. 109

[SNH10] V. Singh, R. Nevatia, and C. Huang. "Efficient Inference with Multiple Heterogeneous Part Detectors for Human Pose Estimation." In *ECCV*, 2010. 92

[SRS10] P. Schnitzspan, S. Roth, and B. Schiele. "Automatic Discovery of Meaningful Object Parts with Latent CRFs." In *CVPR*, 2010. 58, 59, 92

[SS11] M. Sun and S. Savarese. "Articulated Part-based Model for Joint Object Detection and Pose Estimation." In *ICCV*, 2011. 92, 93, 99

[SSF09] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. "Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories." In *ICCV*, 2009. 8, 53, 55, 56

[STF05] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. "Learning Hierarchical Models of Scenes, Objects, and Parts." In *ICCV*, 2005. 76, 124

[STT10] B. Sapp, A. Toshev, and B. Taskar. "Cascaded Models for Articulated Pose Estimation." In *ECCV*, 2010. 92

[SUG11] K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. "Segmentation As Selective Search for Object Recognition." In *ICCV*, 2011. 86

[SWB07] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. "Robust object recognition with cortex-like mechanisms." *PAMI*, **29**:411–426, 2007. 22

[SWR09] J. Shotton, J. Winn, C. Rother, and A. Criminisi. "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Appearance, Shape and Context." *IJCV*, **81**(1), 2009. 109, 128, 129, 130, 145

[SZS10] P. Srinivasan, Q. Zhu, and J. Shi. "Many-to-one Contour Matching for Describing and Discriminating Object Shape." In *CVPR*, 2010. 54, 56, 78

[TF10] D. Tran and D. Forsyth. "Improved Human Parsing with a Full Relational Model." In *ECCV*, 2010. 92

[TFM96] S. Thorpe, D. Fize, and C. Marlot. "Speed of processing in the human visual system." *Nature*, **381**(6582):520–2, 1996. 21

[TMF03] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. "Context-based vision system for place and object recognition." In *CVPR*, 2003. 108

[TMF05] A. Torralba, K. P. Murphy, and W. Freeman. "Contextual Models for Object Detection Using Boosted Random Fields." In *NIPS*, pp. 1401–1408, 2005. 124

[TMF10] A. Torralba, K. Murphy, and W. T. Freeman. "Using the forest to see the trees: object recognition in context." *Comm. of the ACM*, **53**(3):107–114, 2010. 108

[Tor09] A. Torrabla. "How many pixels make an image?" *Visual Neuroscience*, 2009. 107, 122, 125

[TTD10] A. Toshev, B. Taskar, and K. Daniilidis. "Object Detection via Boundary Structure Segmentation." In *CVPR*, 2010. 55

[Tu08] Z. Tu. "Auto-context and Its Application to High-level Vision Tasks." In *CVPR*, 2008. 109

[Val00] L. G. Valiant. *Circuits of the Mind*. 2000. 22

[VF08] A. Vedaldi and B. Fulkerson. "VLFeat: An Open and Portable Library of Computer Vision Algs." `http://www.vlfeat.org/`, 2008. 45

[VGV09] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. "Multiple Kernels for Object Detection." In *ICCV*, 2009. 71, 72

[VK10] S. Vijayanarasimhan and A. Kapoor. "Visual recognition and detection under bounded computational resources." In *CVPR*, 2010. 54, 56

[WS06] J. Winn and J. Shotton. "The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects." In *CVPR*, 2006. 60

[WS08] C. Wojek and B. Schiele. "A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes." In *ECCV*, volume 4, pp. 733–747, 2008. 125

[WSG09] Y. N. Wu, Z. Z. Si, H. F. Gong, and S. C. Zhu. "Learning Active Basis Model for Object Detection and Recognition." *IJCV*, 2009. 8

[WYY10] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. "Locality-constrained Linear Coding for Image Classification." In *CVPR*, 2010. 14, 15, 19

[WZF06] G. Wang, Y. Zhang, and L. Fei-fei. "Using dependent regions for object categorization in a generative framework." In *CVPR*, 2006. 14

[XHE10] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. "SUN Database: Large-scale Scene Recognition from Abbey to Zoo." In *CVPR*, 2010. 109, 122, 128, 138

[YFU12] J. Yao, S. Fidler, and R. Urtasun. "Describing the Scene as a Whole: Joint Object Detection, Scene Classification and Semantic Segmentation." In *CVPR*, 2012. 76, 78, 109, 122, 123, 126, 127, 128, 129, 133, 138, 141, 145, 146

[YR11] Y. Yang and D. Ramanan. "Articulated Pose Estimation using Flexible Mixtures of Parts." In *CVPR*, 2011. 72, 78, 92, 122

[YZH10] Y. Yu, J. Zhang, Y. Huang, S. Zheng, W. Ren, C. Wang, K. Huang, and T. Tan. "Object Detection by Context and Boosted HOG-LBP." In *ECCV workshop on PASCAL*, 2010. 86

[ZBM06] H. Zhang, A. C. Berg, M. Maire, and J. Malik. "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition." In *CVPR*, 2006. 14, 15, 19

[ZCT10] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille. "Part and Appearance Sharing: Recursive Compositional Models for Multi-View Multi-Object Detection." In *CVPR*, 2010. 22, 26, 27, 38, 39, 40

[ZCY09] L. Zhu, Y. Chen, and A. Yuille. "Unsupervised Learning of Probabilistic Grammar-Markov Models for Object Categories." *PAMI*, **31**(1), 2009. 50

[ZCY10] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. "Latent Hierarchical Structural Learning for Object Detection." In *CVPR*, 2010. 9, 43, 58, 77, 86

[ZKT10] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks." *CVPR*, pp. 2528–2535, 2010. 22

[ZLH08] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. "Unsupervised Structure Learning: Hierarchical Recursive Composition, Suspicious Coincidence and Competitive Exclusion." In *ECCV*, 2008. 9, 11, 12, 22, 26, 40, 92

[ZML07] J. Zhang, M. Marszaek, S. Lazebnik, and C. Schmid. "Local Features and Kernels for Classification of Texture and Object Categories." *IJCV*, **73**(2), 2007. 8, 14

[ZP12] C. L. Zitnick and D. Parikh. "The Role of Image Understanding in Contour Detection." In *CVPR*, 2012. 125