Segmentation Features, Visibility Modeling and Shared Parts for Object Detection

by

Patrick Ott

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.



UNIVERSITY OF LEEDS
The University of Leeds
School of Computing

February 2012

The candidate confirms that the work submitted is his own and that the appropriate credit has been given where reference has been made to the work of others.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgment.

Abstract

This thesis investigates the problem of object localization in still images and is separated into three individual parts.

The first part proposes a new set of feature descriptors, motivated by the problem of pedestrian detection. Sliding window classifiers, notably using the Histogram-of-Gradient (HOG) features proposed by Dalal & Triggs are the state-of-the-art for this task, and we base our method on this approach. We propose a novel feature extraction scheme which computes implicit 'soft segmentations' of image regions into foreground/background. The method yields stronger object/background edges than gray-scale gradient alone, suppresses textural and shading variations, and captures local coherence of object appearance. The main contributions of this part are: (i) incorporation of segmentation cues into object detection; (ii) integration with classifier learning *c.f.* a post-processing filter and (iii) high computational efficiency.

The second part of the thesis considers deformable part-based models (DPM) as proposed by Felzenszwalb *et al*. These models have demonstrated state-of-the-art results in object localization and offer a high degree of learnt invariance by utilizing viewpoint-dependent mixture components and movable parts in each mixture component. One might hope to increase the accuracy of the DPM by increasing the number of mixture components and parts to give a more faithful model, but limited training data prevents this from being effective. We propose an extension to the DPM which allows for *sharing* of object part models among multiple mixture components as well as object classes. This results in more compact models and allows training examples to be shared by multiple components, ameliorating the effect of a limited size training set. We (i) reformulate the DPM to incorporate part sharing, and (ii) propose a novel energy function allowing for coupled training of mixture components and object classes.

An 'elephant in the room' for most current methods is the lack of explicit modeling of partial visibility due to occlusion by other objects or truncation by the image boundary. In the third part of this thesis, we propose a method which explicitly models partial visibility by treating it as a latent variable. As a second contribution we propose a novel non-maximum suppression scheme which takes into account partial visibility of objects while, in contrast to other methods, providing a globally optimal solution. Our method gives more detailed scene interpretations, in that we are able to identify the visible parts of an object.

We evaluate all methods on the PASCAL VOC 2010 dataset. In addition, we report state-of-the-art results on the INRIAPerson pedestrian detection dataset for the first part, considerably exceeding those of the original HOG detector.

Acknowledgements

I would like to thank my supervisor Mark Everingham for his help in the past four years. Without his guidance and support this thesis would have not been possible and I am truly grateful for the long meetings, the much needed pushes and everything I have learned from him. I would also like to thank David Hogg for all the support he provided and Jiri Matas for letting me intern at the Center for Machine Perception in Prague.

I had a great time in Leeds, mostly because I was surrounded by great people. Krishna Sridhar, whose support has been invaluable, Sam Johnson, Thomas Müller, Tracy Petrie and Constantine Zakkaroff are just five of them and they deserve my special thanks for so many things they have done in the past years. Many thanks to the technical support staff of Leeds University, especially Graham Hardman, John Hodrien and Dureid El-Moghraby. I would like to thank my friends in Prague, especially James Pritts for all the fruitful discussions over good Thai food, Jan Sochman and Michal Perdoc for making settling in so easy.

Mojej priateľke Zuzke, ďakujem, že si tu bola pre mňa, keď som Ťa najviac potreboval. Ďakujem, že si mi poskytla útočisko v posledných mesiacoch, aby som mohol dopísať svoju dizertačnú prácu a ďakujem, že si ma udržala pri rozume.

Besonderer Dank gebührt meinen Eltern für soviele Dinge die hier nicht aufzählbar sind. Ihnen ist diese Dissertation gewidmet. Ich danke meinen Großeltern, meinem Bruder Thomas und all meinen Freunden in Deutschland, im Besonderen Axel Gäbler und Stefan Koch, die immer ein offenes Ohr für meine großen und kleinen Probleme hatten.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

P. Ott and M. Everingham, "Shared parts for deformable part-based models", *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011*

P. Ott and M. Everingham,"Implicit Color Segmentation Features for Pedestrian and Object Detection", *In Proceedings of the IEEE International Conference on Computer Vision, 2009*

Es irrt der Mensch, so lang er strebt.

Man errs, till he has ceased to strive.

Goethe, Faust I (Der Herr / The Lord)

Gewidmet meinen Eltern

⋆

To my parents

Contents

1 Introduction **1**

1.1 Challenges . . . 2
1.2 Datasets . . . 6
1.2.1 The PASCAL VOC Challenge . . . 6
1.2.2 INRIAPerson Dataset . . . 9
1.3 Sliding Window Detectors . . . 12
1.3.1 Overview . . . 13
1.3.2 Feature Extraction . . . 13
1.3.3 Classification . . . 14
1.3.4 Efficiency Considerations . . . 15
1.3.5 Contextual Modeling . . . 16
1.4 Thesis Overview and Contributions . . . 17

2 Implicit Color Segmentation Features **21**

2.1 Introduction . . . 21
2.1.1 Chapter Contributions . . . 21
2.2 Related Work . . . 23
2.2.1 Pioneering Work . . . 23
2.2.2 Histograms of Oriented Gradients . . . 24
2.2.3 Feature Development and Engineering . . . 24
2.2.4 Deep Learning . . . 25
2.2.5 Pedestrian Detection by Pose Estimation . . . 26
2.2.6 Caltech Pedestrian Dataset . . . 27
2.2.7 Motivation . . . 27
2.3 Histogram of Oriented Gradients . . . 29
2.3.1 The HOG Descriptor . . . 29
2.3.2 Dimensionality Reduction . . . 31
2.3.3 Gradient Computation . . . 33

2.3.4 Classification . . . 33
2.4 Color Histograms of Oriented Gradient . . . 33
2.4.1 Soft Segmentation by Fisher Discriminant . . . 34
2.4.2 Semi-local Segmentation . . . 37
2.4.3 Color Normalization . . . 38
2.4.4 Discussion of CHOG . . . 40
2.5 Classification . . . 43
2.5.1 Support Vector Machines . . . 43
2.5.2 Kernel Support Vector Machines . . . 45
2.5.3 Inhomogeneous Quadratic Kernel . . . 47
2.5.4 Bootstrapping . . . 47
2.6 Empirical Results on INRIAPerson Dataset . . . 48
2.6.1 Implementation Details . . . 48
2.6.2 Experimental Evaluation . . . 50
2.7 Empirical Results on PASCAL VOC 2010 . . . 54
2.7.1 Implementation Details . . . 54
2.7.2 Quantitative Results for PASCAL VOC 2010 . . . 58
2.7.3 Qualitative Results for PASCAL VOC 2010 . . . 61
2.8 Conclusions . . . 66

3 Shared Parts for DPM **71**
3.1 Introduction . . . 71
3.1.1 Chapter Contributions . . . 73
3.2 Related Work . . . 75
3.2.1 Deformable Part-based Models . . . 75
3.2.2 Sharing Parts in Boosted Classifiers . . . 76
3.2.3 Conditional Random Field Approaches . . . 77
3.2.4 Implicit Shape Models . . . 78
3.2.5 Generative Approaches . . . 79
3.3 Deformable Part-based Model . . . 81
3.3.1 Feature Descriptor . . . 81
3.3.2 Parts . . . 83
3.3.3 Inference of Part Positions . . . 84
3.3.4 Window Responses . . . 84
3.3.5 Mixture Model . . . 85
3.4 Sharing Parts . . . 86

3.4.1 Motivation . . . 86
3.4.2 Shared Parts . . . 86
3.5 Learning . . . 88
3.5.1 Energy Function . . . 89
3.5.2 Optimization . . . 91
3.5.3 Convergence Guarantees . . . 96
3.6 Empirical Results . . . 97
3.6.1 Implementation Details . . . 97
3.6.2 Quantitative Results . . . 102
3.6.3 Qualitative Results . . . 112
3.7 Conclusions . . . 118

4 Detection of Partially Visible Objects 123
4.1 Introduction . . . 123
4.1.1 Chapter Contributions . . . 124
4.2 Related Work . . . 126
4.2.1 Approaches to Occlusion Modeling . . . 127
4.2.2 Face Detection . . . 129
4.2.3 Motivation . . . 131
4.3 Modeling Partial Visibility . . . 132
4.3.1 Partial Visibility as a Latent Variable . . . 132
4.3.2 Window Responses . . . 134
4.3.3 Inference . . . 135
4.3.4 Mixture Model . . . 135
4.4 Learning . . . 136
4.4.1 Energy Function . . . 137
4.4.2 Optimization . . . 137
4.5 Fusion and Non-Maximum Suppression . . . 138
4.5.1 Previous Work . . . 139
4.5.2 Preparation & Notation . . . 141
4.5.3 Overlap Measure . . . 143
4.5.4 Energy Maximization for NMS . . . 143
4.5.5 Optimization by Branch-and-Bound . . . 144
4.6 Emprical Results . . . 145
4.6.1 Implementation Details . . . 146
4.6.2 Quantitative Results . . . 147

4.6.3 Qualitative Results . . . 152
4.7 Conclusions . . . 156

5 Conclusions and Future Work **158**
5.1 Conclusions . . . 159
5.1.1 Implicit Color Segmentation Features . . . 159
5.1.2 Shared Parts for Deformable Part-based Models . . . 160
5.1.3 Detection of Partially visible Objects . . . 161
5.2 Future Work . . . 162
5.2.1 Implicit Color Segmentation Features . . . 162
5.2.2 Shared Parts for Deformable Part-based Models . . . 163
5.2.3 Detection of Partially Visible Objects . . . 165
5.3 Outlook and Final Thoughts . . . 167

A Additional Results **170**
A.1 Implicit Color Segmentation Features . . . 170
A.2 Shared Parts for Deformable Part-based Models . . . 173
A.3 Detection of Partially Visible Objects . . . 173

Bibliography **180**

List of Figures

1.1 Example images of the PASCAL VOC 2010 [30] dataset 3
1.2 Annotation provided by the PASCAL VOC challenge 7
1.3 Bounding Box Overlap and Average Precision 8
1.4 Sample of positive training images from the INRIA dataset 10
1.5 Example DET curve 11
1.6 Pipeline of a general sliding window detector 12

2.1 Color as a discriminative indicator 22
2.2 The 'verification by segmentation' method of Ramanan [105] 28
2.3 Histograms of Oriented Gradient feature descriptors by Dalal & Triggs [23] 30
2.4 Detection pipeline for HOG features 32
2.5 Extraction scheme for CHOG features 34
2.6 Local segmentation model used by CHOG 36
2.7 Potential reference points for CHOG descriptors 37
2.8 Color normalization of CHOG descriptors 39
2.9 Comparison of HOG & CHOG feature descriptors 41
2.10 Additional modeling capacity of CHOG features 42
2.11 Performance evaluation on INRIAPerson I 51
2.12 Performance evaluation on INRIAPerson II 52
2.13 Visualization of the positive weights of a pedestrian detector 53
2.14 Car and bicycle detectors with average bounding box 56
2.15 Precision/recall curves for selected PASCAL VOC2010 classes 62
2.16 Visualization of the positive weights of a car detector 63
2.17 Visualization of the positive weights of a bicycle detector 64
2.18 Visualization of the positive weights of a bus detector 65
2.19 Example detections for HOG and HOG+CHOG 67
2.20 Example detections for HOG+CHOG with linear and nonlinear SVM . . 68

3.1 The DPM training "dilemma" 72

3.2 Comparison of DPMs with and without part sharing I 74
3.3 Computing detection scores in the DPM of Felzenszwalb *et al*. [38] . . . 82
3.4 Part extraction and spatial configuration 83
3.5 Comparison of DPMs with and without part sharing II 87
3.6 DPM Optimization Scheme 92
3.7 Initial sliding window 'motorbike' detector with average bounding box . . 99
3.8 Precision/recall curves for selected PASCAL VOC 2010 classes 105
3.9 Results of part-sharing for PASCAL VOC 2010 I 107
3.10 Results of part-sharing for PASCAL VOC 2010 II 109
3.11 Results of part sharing for PASCAL VOC 2010 III 111
3.12 Results of part sharing for jointly trained PASCAL VOC 2010 classes . . 112
3.13 Car detector with 6 shared parts and 3 mixture components. 114
3.14 Car detector with 12 shared parts and 3 mixture components. 114
3.15 Motorbike detector with 12 shared parts and 3 mixture components. . . . 115
3.16 Horse detector with 12 shared parts and 3 mixture components. 116
3.17 Part locations for 'car' detections. 117
3.18 Part locations for 'motorbike' detections. 119
3.19 Part locations for 'horse' detections. 120

4.1 Examples of partial visibility of the car class 124
4.2 Pipeline of proposed detection framework 125
4.3 Patterns of occlusion considered by Lin *et al*. [81] 129
4.4 An example detection with indicated partial visibility 130
4.5 Modeling of partial visibility 132
4.6 The imposed Ising model used for partial visibility modeling 134
4.7 Occlusion-aware NMS *c.f*. regular NMS 138
4.8 Comparison of regular detector to visibility-aware detectors 140
4.9 Matrices for the non-maximum suppression scheme 142
4.10 Precision/recall curves for selected PASCAL VOC 2010 classes 150
4.11 Example detections for the baseline and partial visibility modeling 153
4.12 Additional detections of the proposed NMS scheme 155

A.1 Remaining Precision/recall curves for HOG+CHOG experiments I 171
A.2 Remaining Precision/recall curves for HOG+CHOG experiments II . . . 172
A.3 Remaining Precision/recall curves for HOG+CHOG experiments classes III 173
A.4 Remaining Precision/recall curves for shared parts experiments I 174
A.5 Remaining Precision/recall curves for shared parts experiments II 175

A.6 Remaining Precision/recall curves for shared parts experiments III 176
A.7 Remaining Precision/recall curves for visibility modeling experiments I . 177
A.8 Remaining Precision/recall curves for visibility modeling experiments II . 178
A.9 Remaining Precision/recall curves for visibility modeling experiments III 179

List of Tables

2.1 Height of the bounding box used for feature extraction 55
2.2 AP results for PASCAL VOC 2010 (CHOG features) 59

3.1 AP results for PASCAL VOC 2010 (DPM with shared parts) 103

4.1 AP results for PASCAL VOC 2010 (partial visibility modeling) I 149
4.2 AP results for PASCAL VOC 2010 (partial visibility modeling) II 151
4.3 Maximum recall results for PASCAL VOC 2010 (partial visibility modeling) 152

List of Acronyms

AP - Average Precision

CCCP - Concave-Convex Programming

CHOG - Color Histograms of Oriented Gradient

CRF - Conditional Random Field

DPM - Deformable Part-based Model

HOG - Histograms of Oriented Gradient

L-BFGS - Limited-Memory Broyden-Fletcher-Goldfarb-Shanno

MRF - Markov Random Field

NMS - Non Maximum Suppression

PASCAL - Pattern Analysis, Statistical Modelling and Computational Learning

SVM - Support Vector Machine

VOC - Visual Object Classes

List of Important Notation

Vectors

$\boldsymbol{w}$ - Weight vector, usually modeling appearance

$\boldsymbol{x}$ - Feature vector

$\boldsymbol{v}$ - Visibility flags

u - Unary bias term

$\boldsymbol{z}$ - Position of a part

$\boldsymbol{a}$ - Anchor position of a part

$\boldsymbol{s}$ - Spatial prior of a part

Matrices

$\boldsymbol{\beta}$ - Combination weights in a DPM with shared parts

$\boldsymbol{H}$ - Relates visible blocks of a detection to a scene

$\boldsymbol{G}$ - Relates score of a detection to covered blocks in a scene

Sets

$\mathcal{T}$ - Training dataset

$\mathcal{D}$ - Detections

$\mathcal{I}$ - Scene interpretation

$\mathcal{W}$ - Appearance weights

$\mathcal{S}$ - Spatial priors

$\mathcal{U}$ - Unary bias terms

$\mathcal{M}$ - Latent mixture assignments

$\mathcal{V}$ - Latent visibility flags

$\mathcal{P}$ - Latent part placements

$\mathcal{Z}$ - Latent part placements per example

$\mathcal{O}$ - Model variables

$\mathcal{L}$ - Latent variables

Functions

$K(\cdot)$ - Kernel (SVM)

$E(\cdot)$ - Energy function

$R(\cdot)$ - Regularization function

$L(\cdot)$ - Loss function

$F(\cdot)$ - Energy function for NMS

$\vartheta(\cdot)$ - Part response

$f(\cdot)$ - Classification score

$g(\cdot)$ - Mixture component

$h(\cdot)$ - Detector

CHAPTER 1

INTRODUCTION

This thesis investigates the task of object detection in still images, also referred to as object localization. The task can be described as enabling a computer to locate all instances of a specific object class (*e.g.* 'car', 'bicycle', 'sheep' or 'person') in a given image. This involves the learning of a statistical model using a representation of object appearance, which captures salient properties of the object classes in question.

Object detectors have many areas of application and form an important component in the current research on higher-level scene understanding of images. The ability to detect the class, presence and location of objects in an image supports more detailed interpretations of the scene *e.g.* understanding the interactions between objects or people and objects. This leads to the ability to analyze and understand activities in images and videos [24, 122]. Object detection therefore also directly contributes to the field of robotics in helping robots to understand and interpret the world around them [88]. In computer vision, object detectors are often also used for tracking, *i.e.* the process of locating and following objects in a video scene, by initializing the tracker with the help of an object detector [68] or by using 'tracking-by-detection' approaches [4, 28]. In the research area of human pose estimation, which deals with the determination of a configuration of a person in an image or video, object detectors are actively used to detect individual body

parts [104, 5, 65, 66]. Object detection is also employed in the industrial sector *e.g.* in 'smart cars' [51] to detect people and ensure pedestrian safety or in digital cameras to detect faces [138] and perform automatic camera calibration. Furthermore, object detectors can be employed to perform image and video search by, for example, presenting an image containing a specific object to the computer and letting it retrieve images containing similar objects from a large database [119, 118] or assisting the user in identifying the object.

In the following section we describe the challenges of object detection in natural images. One dataset in which these challenges are particularly evident is the PASCAL VOC 2010 dataset [30] which is introduced in Section 1.2 and used for experimental evaluation throughout this thesis. In Section 1.3 we explain sliding window detectors, a method which we use as a basis for the work presented in this thesis. The chapter closes with an overview of the thesis and a summary of our contributions.

1.1 Challenges

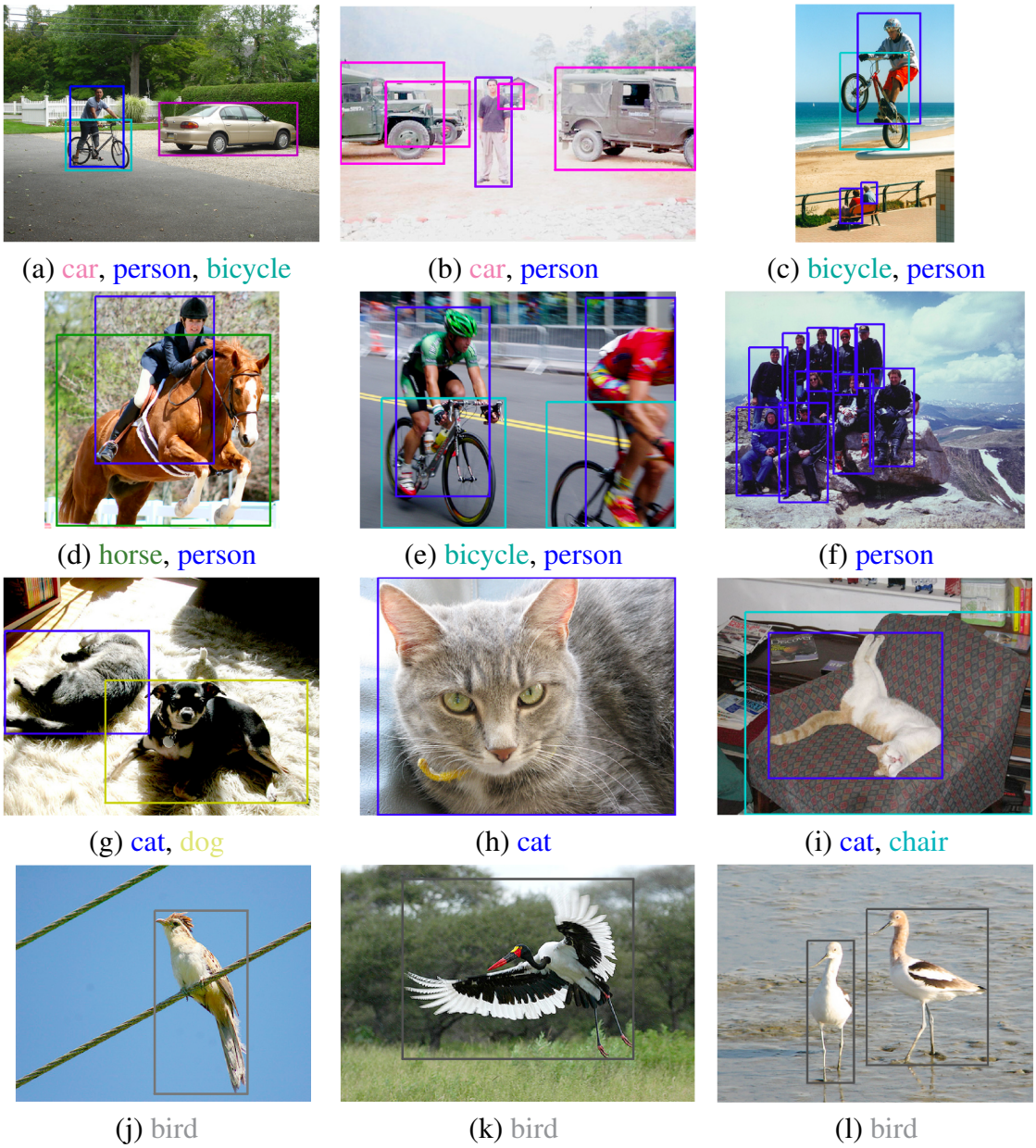
Object detection remains an active area of research due to the many challenges that need to be addressed. In this section we provide a summary of the main challenges and briefly outline our contributions in addressing some of these. We illustrate each of these challenges using examples of natural images in Figure 1.1. These images are a subset of the images provided by the PASCAL VOC 2010 dataset [30], which will be introduced in detail later.

Intra-class Variation

A significant challenge in object detection arises due to intra-class variation in appearance of an object class. Most object classes exhibit large intra-class variation due to three main reasons: (i) pose/structure; (ii) viewpoint and (iii) object surface characteristics.

Examples of wide variations in pose are often found in images containing people, thus introducing wide intra-class variance for the 'person' class. This is illustrated in Figure 1.1 (a)–(f) in which people appear in many different poses. The poses of animals can also be highly articulated, leading to drastic changes in appearance. Figure 1.1 (g)–(i) shows example images of the 'bird' and 'cat' classes.

Rigid object classes also tend to exhibit differences in appearances due to structural variation as shown by the different types of cars in Figure 1.1 (a, b). Different viewpoints



(a) car, person, bicycle (b) car, person (c) bicycle, person

(d) horse, person (e) bicycle, person (f) person

(g) cat, dog (h) cat (i) cat, chair

(j) bird (k) bird (l) bird

Figure 1.1: Example images of the PASCAL VOC 2010 [30] dataset. Bounding box colors correspond to the different object classes. Large viewpoint changes for all object classes are commonplace and objects are often significantly occluded, either by objects of a similar class (b, f) or another class (c, e). In addition, objects are very often truncated by the image boundary (b, e, h). Examples (g)-(i) show different animal object classes and we observe a large variety of poses and a significant difference in intra-class appearance.

also contribute to substantial intra-class variance. For example, Figure 1.1 (h) and (i) illustrate how different viewpoints can affect the appearance of the 'cat' object class. Figure 1.1 (a, c, d) shows instances of the 'bicycle' class captured from different viewpoints.

Object detectors require the ability to deal with this wide range of poses and viewpoints. To address this problem we adopt the approach of Deformable Part-based Models (DPM) [38] in Chapter 3. DPMs break up the appearance of an object class into 'sub-classes' which correspond to different viewpoints (*e.g*. front facing and side-view cars) or differences in pose/structure of an object class *e.g*. different configurations of a person. In each of these sub-classes the DPM uses a flexible part-based model to capture small variation in appearance. However, traditional DPMs [38] find it hard to cope with limited training data per sub class and model complexity as the number of sub-classes increases. In this thesis we propose an efficient way of handling this problem by sharing components of the model among sub-classes and even across object classes [97].

Even for similar pose and viewpoint, variability in *instance-specific* characteristics *e.g*. clothing of people, fur of animals or the paint of cars further contribute to intra-class variance. This can be observed in Figure 1.1 (d, e) in which people appear in a similar pose but with substantial differences in clothing. For animals, different subspecies can cause further complications in the effective modeling of the variance of appearance within each class. Figure 1.1 (j, k, l) shows examples of three different kinds of birds, illustrating the large variation of instance-specific appearance for that object class.

In Chapter 2 we propose a method of incorporating instance-specific object characteristics to enhance the object representation [96]. This results in a more efficient modeling of the variances induced by object-specific characteristics.

Inter-class Similarity in Appearance

Despite obvious differences in appearance for some object classes (*e.g*. 'car' compared to 'bird') many object classes share similar characteristics. For example, both motorbikes and bicycles can be considered as examples of a more general 'two-wheeler' class and therefore often appear somewhat similar. This is also true for object classes such as 'cat' and 'dog', which can appear similar or even nearly indistinguishable to an observer – see Figure 1.1 (g).

For this reason, throughout the thesis, we adopt the general scheme of discriminative learning which aims to model characteristics that discriminate object classes despite these inter-class similarities. To further aid discriminative learning we also adopt the general strategy of bootstrapping which selects samples from other classes and the background

that are particularly hard to discriminate.

Partial Visibility

Partial visibility of object instances poses another significant challenge for the detection of objects in natural images because areas which may hold valuable cues for recognition might be hidden from the detector. Partial visibility manifests itself in the form of occlusion of an object class by instances of the same object class (Figure 1.1 (b, f)) or other object classes – see *e.g.* Figure 1.1 (d, e). In addition, the truncation of an object instance by the image boundary introduces another form of partial visibility as shown in Figure 1.1 (b, e, h).

In Chapter 4 we propose a novel approach which explicitly tackles the problem of partial visibility of an object instance by inferring which parts of an object may be occluded. We propose a mechanism in the detection process which compensates for the absence of potential recognition cues in the regions that are inferred as being occluded.

Other challenges

Additional more global challenges in the localization of objects in natural images exist. For example, objects usually exhibit variation in scale and this is especially challenging since potentially useful depth information (such as the ordering of objects) has been suppressed in the image formation process. In addition, object instances at different scales offer a different degree of detail in appearance, *e.g.* the instances of the 'bicycle' class in Figure 1.1 (e) are much more detailed in their appearance than the instance in Figure 1.1 (a). To detect objects at different scales, we use sliding window detectors throughout this thesis. Such detectors densely scan the image for object instances at multiple scales.

The illumination of an image scene can significantly influence the appearance of objects. Different light sources, such as natural (sun) light or artificial light, can have a great impact on the apparent color of an object instance. For example, the cat in Figure 1.1 (g) appears in intensive natural light which causes a light-colored appearance while the cat itself has an original gray color. In contrast, the cat in Figure 1.1 (i) appears in artificial light and depicts its natural color. Object detectors therefore need to be robust to the effect of scene illumination on object appearance. Throughout this thesis we build on an image representation which incorporates multiple levels of illumination invariance [23] while Chapter 2 specifically focusses on further improving this image representation by

implicitly modeling illumination amongst other image characteristics.

1.2 Datasets

Throughout this thesis the PASCAL VOC 2010 dataset [30] is used for experimental evaluation. We have already shown example images from this dataset in Figure 1.1 and explained the challenges that arise from object detection in natural images. Chapter 2, which is particularly motivated by the task of pedestrian detection, is additionally evaluated on the INRIAPerson dataset [23]. This section provides key statistics for both datasets and explains the individual evaluation protocols. We chose these datasets as representing the state-of-the-art at the time of carrying out the work presented in this thesis. An overview of other popular datasets is provided by Torralba & Efros [126], who also give interesting insights into dataset bias and additional information on the generality of available datasets today.

1.2.1 The PASCAL VOC Challenge

The PASCAL Visual Object Classes (VOC) challenge [32] is a benchmark in visual object category recognition. The challenge is organized annually and started in the year 2005 [34]. To evaluate object detection, the PASCAL VOC challenge provides the "object detection competition" (`comp3`) which asks the participants of the challenge to learn an object detector from a subset of annotated images – the training dataset. To evaluate the detector the participant has to predict bounding boxes of object instances in the test dataset.

While the annotation for training data is provided, annotation for the test data is not made public. Evaluation of detectors is performed by an online evaluation server which only allows a limited number of submissions per week. This rigorous evaluation process ensures that optimization of specific algorithms on the test subset is not possible.

Annotation for the `comp3` competition is provided in the form of bounding boxes for each ground truth object – examples are shown in Figure 1.2. Additional annotation is available in the form of a categorical viewpoint label (facing left, frontal, ...), whether or not it is truncated by the image boundary or appears occluded in the image. An additional 'difficult' flag indicates whether an object instance can be considered difficult to detect. Difficult object instances are often strongly occluded/truncated or hard to recognize even for a human observer (see *e.g.* example in the top of Figure 1.2) and are ignored during

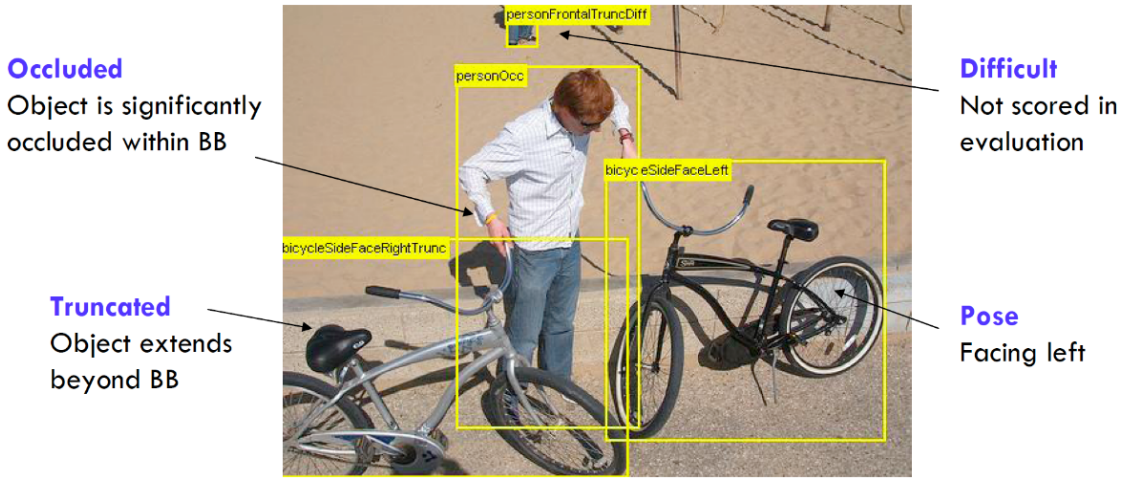


Figure 1.2: Annotation provided by the PASCAL VOC challenge [32]. Each object instance is enclosed by a bounding box (shown in yellow). In addition the annotation provides the pose of an object and whether or not it is occluded or truncated by the image boundary. An additional 'difficult' flag indicates whether an object is difficult to detect. Objects flagged in this way are ignored during evaluation.

evaluation, which is discussed below.

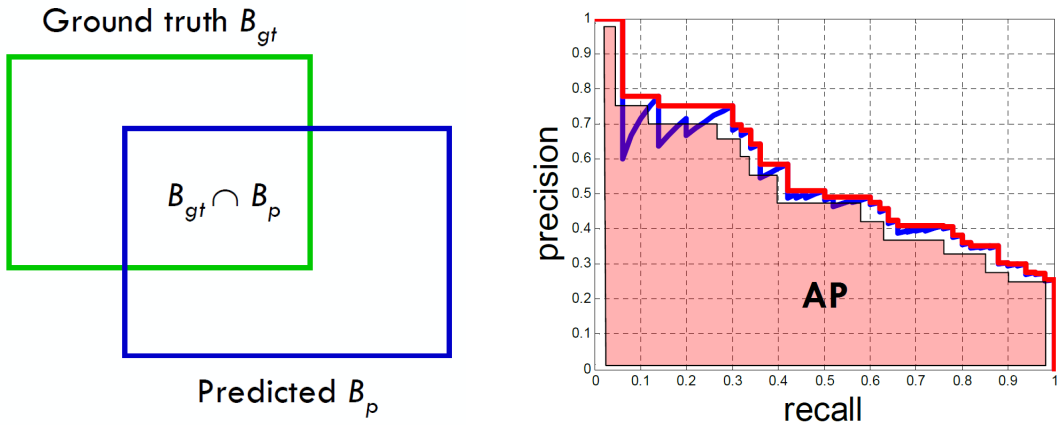
We consider the PASCAL VOC 2010 dataset one of the most challenging object detection datasets available to date. It strikes a good balance between the available training data, number of object classes and accuracy of the provided annotation. The large amount of available training data per class is ideal to train the complex object detection models presented in this thesis.

Key Statistics

As previously mentioned the PASCAL VOC 2010 [30] dataset features 20 object classes separated into the following categories:

- Person: 'person'
- Animal: 'bird', 'cat', 'cow', 'dog', 'horse', 'sheep'
- Vehicle: 'aeroplane', 'bicycle', 'boat', 'bus', 'car', 'motorbike', 'train'
- Indoor: 'bottle', 'chair', 'dining table', 'potted plant', 'sofa', 'tv/monitor'

In total the dataset consists of $21,738$ images. These are split into training (`train`), validation (`val`), and test (`test`) subsets. The combination of the training and validation



(a) Calculation of Bounding Box Overlap (b) Calculation of Average Precision

Figure 1.3: Average Precision for precision/recall curves and the overlap of bounding boxes used by the PASCAL VOC evaluation measure. (a) A prediction is considered a true positive if the predicted bounding box B_p overlaps at least 50% with the ground truth bounding box B_{gt}. (b) Precision/recall curves are interpolated in such a way that they are monotonically non-increasing. The area under the curve then serves as the 'Average Precision', a single descriptive number to summarize the quality of the detector.

subsets is called `trainval`. The dataset is equally split between `trainval` and `test`. For `trainval` $10,103$ training images with $23,374$ annotated objects are available. Further statistics are provided by Everingham *et al*. [30].

Evaluation Measure of the VOC Challenge

Evaluation in the `comp3` competition is performed as a set of single-class detection tasks, *i.e*. the applicability of a provided detector to the task of object detection is evaluated for each object class *separately*. The evaluation measure employed by the PASCAL VOC challenge is the precision/recall curve [132]. This curve is computed from detections that are ranked by a confidence score which has to be provided by the detection methods entering the challenge. Recall is defined as "the proportion of all positive examples ranked above a given rank" while precision is defined as "the proportion of all examples above that rank which are from the positive class" [32]. Detections are considered true positive if their predicted bounding box B_p overlaps at least 50% with the ground-truth bounding box B_{gt}:

$$ov\left(B_p, B_{gt}\right) = \frac{area\left(B_p \cap B_{gt}\right)}{area\left(B_p \cup B_{gt}\right)} \geq 50\% \tag{1.1}$$

where $B_p \cap B_{gt}$ is the intersection of the predicted and the ground-truth bounding box and $B_p \cup B_{gt}$ denotes their union – this is also visualized in Figure 1.3 (b). It is important to note that all additional detections to the highest ranking detection satisfying the overlap constraint of Equation 1.1 are considered false positives and hence a method to fuse neighboring detections belonging to one and the same ground-truth object is required to yield good precision/recall curves. These methods are referred to as non-maximum suppression (NMS). We will return to the issue of NMS in Chapter 4 of this thesis.

Average Precision

To summarize the results provided by a precision/recall curve into a single descriptive number the PASCAL VOC challenge computes Average Precision (AP) as the area under an interpolated precision/recall curve – Figure 1.3 (b) provides a visual description. The interpolation of the curve ensures that precision is monotonically non-increasing, which causes the AP value to be robust w.r.t. commonly appearing sawtooth shapes in precision/recall curves. To obtain high AP a detector must achieve both high recall *and* high precision.

1.2.2 INRIAPerson Dataset

Chapter 2 of this thesis is particularly motivated by the task of pedestrian detection, which can be seen as a specialized object detection task. To evaluate the applicability of the methods presented in Chapter 2 to the task of pedestrian detection we additionally perform experiments on the INRIAPerson dataset introduced by Dalal & Triggs [23, 22].

The INRIAPerson dataset differs from the PASCAL VOC datasets [32] in that it casts the pedestrian detection problem as a forced-choice binary classification, *i.e.* does this image show a pedestrian or not? For that purpose the INRIAPerson dataset provides cropped positive training and test images centered on the pedestrian. Negative training and test data is provided in the form of uncropped images which do not contain pedestrians. At test time these images are scanned for pedestrians by moving a window of fixed size over the image and analyzing the image content beneath that window, essentially creating a large set of cropped negative images.

Key Statistics

The INRIAPerson dataset contains $2,416$ cropped positive training images ($1,208$ truly unique ones and their left-right reflections, centered on humans) and a set of $1,218$ un-



Figure 1.4: Sample of positive training images from the INRIA dataset [23]. The subset of images illustrates the wide variety of poses, different lighting conditions and image quality. All people are standing upright but still in significantly different poses. Some appear on a bike while others appear standing still and a few walking or even jumping. Poor lighting conditions sometimes make it hard to tell the actual pose of the pedestrian and indeed to locate the pedestrian at all.

cropped negative training images. For testing, another $1,128$ cropped positive images (564 unique ones with their left-right reflections, again centered on the humans) and 453 uncropped negative images are provided. The negative images contain many windows (over $10,000$ per image) which are each evaluated separately for the presence of a pedestrian. As a result the amount of negative training and test data significantly outnumbers the amount of positive training and test data. It should also be noted that uncropped positive images with bounding box annotation are available, *i.e.* the performance of detectors can also be evaluated using the PASCAL VOC evaluation methodology (Section 1.2.1).

A selection of positive training examples is presented in Figure 1.4. While the 'person' class of the PASCAL VOC 2010 dataset [30] offers a wide range of poses, the INRIA-Person dataset offers mainly upright standing pedestrians as they are commonly found in street scenes. However, even with that restriction in mind, poses still vary significantly – see Figure 1.4.

Evaluation Methodology

To evaluate the performance of detectors on the INRIAPerson dataset, Dalal & Triggs [23] use Detection Error Tradeoff (DET) curves [87]. Such curves plot the miss rate on the

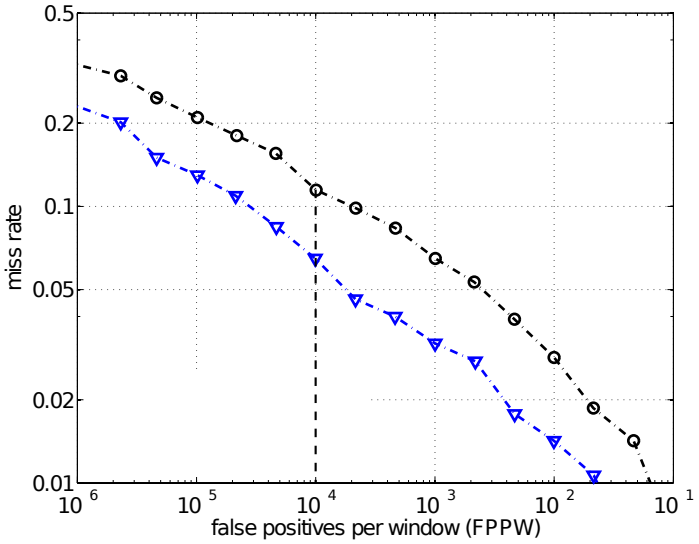
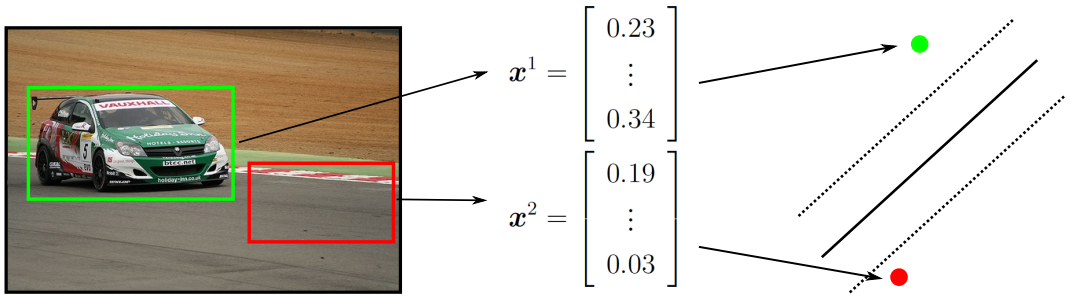


Figure 1.5: For experiments on the INRIAPerson dataset [23] Detection Error Tradeoff (DET) curves are used. These curves plot false positives per window (FPPW) against miss rate on a log-log scale. In the figure two DET curves are shown. Curves closer to the south-west corner of the plot (*i.e.* the blue curve) are considered better. A value of 10^{-4} FPPW and the corresponding miss rate is used as a reference point by many researchers in the field to compare different detectors and is shown in the plot.

y-axis against false positives per window (FPPW) on the x-axis using a log-log scale – an example DET curve is shown in Figure 1.5. Miss rate is defined as

$$missrate = \frac{FalseNegatives}{TruePositives + FalseNegatives} \tag{1.2}$$

DET curves are plotted on a log-log scale to better distinguish the performance at low false positive rates. Dalal & Triggs [23] focus on a certain range of false positive values from 10^{-1} to 10^{-6}. It is common for researchers to report miss rate at a FPPW rate of 10^{-4} as a reasonable summary of performance. For the pedestrian detectors presented by Dalal & Triggs [23], this FPPW rate roughly corresponds an error rate of 0.8 false positive detections per 640×480 image. The focus on small FPPW values sets DET curves apart from the commonly used and closely related ROC (Receiver Operating Characteristic) curves [121], which plot false positive rate against true positive rate. In case of the INRIAPerson dataset the amount of negative test data clearly outnumbers the amount of positive test data as mentioned earlier and hence a focus on small FPPW rates is required. In addition most negative training windows are easy to classify, leading to a very low miss



(a) Image with windows

(b) Feature vectors $\boldsymbol{x}^1, \boldsymbol{x}^2$

(c) Classification in the feature space of $\boldsymbol{x}$

Figure 1.6: Pipeline of a sliding window detector. (a) The image is densely scanned at various scales by a window of a fixed size; two windows are visualized. The green window contains the object class of interest (car), the red one does not. (b) Features are extracted from the underlying image content and represented by the feature vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$. (c) In feature space a classifier is used to obtain a confidence that the window depicts an instance of the object class. In the visualization we use a Support Vector Machine [133].

rate for FPPW values larger than 10^{-1} – see Figure 1.5.

1.3 Sliding Window Detectors

In this section we describe sliding window detectors, which form a key component of the methods proposed in the following chapters. Our choice of this particular detection scheme is motivated by its simplicity, in that it allows the application and extension of a conventional machine learning paradigm, and by the success of the approach demonstrated by many existing methods.

In the following we discuss the main idea and motivation behind the sliding window detector, including aspects of image representation and classification. We additionally briefly discuss methods for improving computational efficiency (Section 1.3.4) and schemes for incorporating contextual information (Section 1.3.5). Alternative approaches to sliding window object detection will be discussed throughout the remainder of this thesis and individually evaluated in the scope of the respective chapter.

1.3.1 Overview

Sliding window detectors essentially turn a 'whole-image' object/non-object classifier into an object detector by using exhaustive search over possible locations and scales of the object. The classifier assigns confidence to an extracted sub-image – the *window* – which either contains an instance of the object class or not. The sliding window detection scheme is visualized in Figure 1.6 and can be broadly separated into two stages:

- *Step 1, Feature extraction*: In the first stage a window of fixed size is moved over an image, usually using a small stride of a few pixels to ensure a dense scan. Two example windows are shown in Figure 1.6 (a). For each window, a feature descriptor of the underlying image content is extracted and represented by a vector $\boldsymbol{x}$ – see Figure 1.6 (b).

  Once the image is completely scanned, we resize it by a fixed scale factor and repeat the scanning process to detect objects of multiple scales. This ensures that the window size is fixed and consequently the feature vectors of all windows have the same length and lie in the same feature space.

- *Step 2, Classification*: In the second stage the feature vector for each window is passed to a classifier, which assigns a score. This score can be interpreted as the classifier's confidence that the window (represented by the features $\boldsymbol{x}$) contains the object class of interest – see Figure 1.6 (c). The classifier is usually discriminative [133, 10] because of the good performance of such approaches, while in general generative classifiers may also be used [113].

In the following we consider further details with respect to these two stages. Specifically, two questions are addressed: (i) which features should be used to represent the image content and (ii) which classifier is the best choice?

1.3.2 Feature Extraction

An obvious choice of feature representation would be to represent the content of a window using the raw pixel values of the underlying image area. However, such a representation does not explicitly capture the salient properties of the objects which may be present in the window. Moreover, a classifier trained using such a feature representation may not generalize well as it can easily over-fit, *i.e.* model properties specific to the training dataset.

In order to generalize well, feature extraction schemes should ideally encode *invariance* into the feature descriptors so that they may capture salient properties representative of an object class despite changes in *e.g*. illumination or small translations. Chapter 2 deals with the development of a new feature descriptor which introduces additional invariant properties over existing work. We consider related work on feature descriptors and recent developments in feature engineering in Section 2.2.

1.3.3 Classification

As mentioned earlier, we adopt a discriminative approach towards classification by using Support Vector Machines (SVM) [10, 133] in this thesis. SVMs are our preferred choice because they implement maximum margin learning in a principled way [21]. This form of learning aims to maximize the 'gap' between positive and negative training examples. In addition, SVMs provide a natural way of incorporating kernels that enable learning of non-linear discriminants which improve classification performance in an optimal manner [116]. We will further discuss the topic of SVMs in Chapter 2. Another popular choice of classifiers are boosted ones, *e.g*. AdaBoost [46], which accumulate scores of a set of weak classifiers to form a strong classifier. Weak classifiers often correspond to the evaluation of a single feature or a small set of features.

We use a fully supervised learning scheme [10], which means that training data (feature vectors) is provided with labels (object/non-object). These feature vectors and labels are derived from a set of bounding boxes which are provided by the training data *e.g*. the PASCAL VOC 2010 dataset [30]. Semi-supervised approaches, which consider partially provided training labels, are also an active area of research in computer vision – see *e.g*. Fergus *et al*. [40, 43]. The discovery of object classes given unlabeled training data is also a well studied research field – Tuytelaars *et al*. [130] provide a recent overview of the area.

At training time, initial positive and negative training data is extracted. Feature vectors of positive training data are extracted at the location of the object of interest as provided by the training dataset. Negative training examples are usually extracted by sampling feature vectors of random windows which do not contain the object. It is usually not possible to store all negative training examples *i.e*. the feature vectors of all windows which do not contain the object class, because modern datasets [32, 23] contain many millions of such windows. We use the 'bootstrapping' method [95] to collect salient negative windows which are difficult to recognize, and encourage learning stronger detectors. During bootstrapping the learned detector is executed on the training images and high-scoring

false positives are extracted. These false positives are added to the training dataset and the classifier is retrained, usually yielding a detector which generalizes better on unseen data. Bootstrapping is further explained in consecutive chapters. Note that throughout the thesis we refer to the learning stage of an object detector as 'training time' while the evaluation of an object detector is referred to as 'test time'.

1.3.4 Efficiency Considerations

As mentioned earlier, in a sliding window detection scheme the classifier is applied to every window in the image. Commonly an image contains many windows at multiple scales (more than $10,000$) and consequently the speed of evaluating each window is commonly the key determinant of the execution speed of sliding window detectors. For this reason, complex feature extraction and classification schemes are usually not applicable to sliding window detectors. As a result previous work on sliding window detectors has often used image representations which can be computed in reasonable time in combination with linear SVMs due to their relatively low computational complexity. More complex classification schemes, such as kernel SVMs [116], are generally more expensive to evaluate, as will later be explained in Section 2.5.2. In Chapter 2 we consider the question of how to engineer an image representation which offers better detection accuracy in a sliding window framework while maintaining reasonable computational complexity.

Some previous work has focused on enabling more complex classifiers to be used with sliding window detectors by employing a cascade architecture [134, 148, 37] motivated by the face detection framework of Viola & Jones [138]. In a cascade architecture a sequence of classifiers is defined in order of increasing complexity and corresponding increase in computational expense. Positive windows should be accepted by all classifiers in the sequence while negative windows which are 'simple', *e.g.* uniform image regions, can be rapidly rejected by classifiers earlier in the cascade. Difficult negative windows, *e.g.* highly textured regions, are rejected by the more complex classifiers later in the cascade, at greater computational expense. The early rejection of abundant simple negative windows greatly reduces the average evaluation time per window. A related approach is the coarse-to-fine search of Pedersoli *et al.* [102], which uses fast coarse object detectors to prune the search space for object detectors which use a finer image representation but take longer to evaluate. In the context of a cascade architecture the coarse object detectors can be thought of as classifiers at the early stages of the cascade while the detectors using a fine image representation can be thought of as complex classifiers which are evaluated later in the cascade.

Lampert *et al*. [74] propose an efficient search of the image by using an upper bound on the classification score which can be achieved by any window within an image region. The upper bound is used to iteratively discard larger regions of the image, so that the search for an object can be increasingly focused on smaller regions that potentially contain the objects of interest. To that end they propose a branch and bound procedure to efficiently explore the search space of all image regions.

Alternatives to cascade architectures or the approach of Lampert *et al*. [74] are heuristic methods which find promising regions of the image where more complex classifiers can be applied. This can help to reduce false positive responses to clutter, and furthermore allows the use of expensive classifiers. Examples include the method of Chum & Zisserman [17] or region-based methods [18]. Chum & Zisserman [17] propose a method that identifies regions of interest containing configurations of interest points [55, 67] such that they are similar to equivalent configurations for a set of exemplar images corresponding to an object class. By restricting the sliding window detectors only to these regions of interest the detection process can be made more computationally efficient. Region-based methods *e.g*. Chunhui *et al*. [18] first create initial hypothesis of the positions of objects in an image scene based on matched regions from training examples. These hypothesis are then used as an input to a more expensive verification classifier in a second stage.

1.3.5 Contextual Modeling

A common approach to improving object detectors, which is also adopted in this thesis, is to advance their feature extraction and classification schemes. Another approach, however, is to leverage contextual information in the scene. The motivation behind such approaches is that sliding window detectors by themselves are somewhat limited: they only consider a small window of the image at a time while discarding all other image information. Other parts of the image may, however, hold important information regarding the detection of object instances. Some work [125, 60] has proposed the use of contextual probabilistic priors for object location, scale and type. Such approaches take the entire image representation into account to provide a better idea of what kind of an object is to be expected in the scene and where it may be located.

Other approaches to contextual modeling have specifically focused on leveraging information provided by the co-occurrence of object classes in a scene [49, 25]. For example, cars may often appear together with motorbikes, and people are often seen riding motorbikes, but horses and motorbikes co-occur less often. Leveraging such information can be helpful in that detections of object classes which often appear together can re-

inforce each other while potential false-positive detections from the background can be removed.

Specific to the recognition of persons in an image, it has also been proposed to contextually model the actions these persons are performing in the image [144, 110]. The focus on a single action can constrain the range of appearance of a particular object class *e.g*. a person riding a horse exhibits less variation in pose than a person performing any possible action. Approaches like these can potentially help with the challenging modeling of intra-class variability of the person object class.

Harzallah *et al*. [56] incorporate contextual information into the detection scheme by using image classification *i.e*. the decision of whether an image contains an object instance (independent of location). They report that an object detector can be improved by combining it with a whole image classifier which can recognize scenes likely to contain the object *e.g*. street scenes which usually contain cars. This is especially helpful in the detection of occluded or truncated object instances which miss important key points for recognition by an object detector but can be sufficiently recognized by leveraging the contextual information provided by the whole image classifier. Inversely, object detectors can also support the image classification task by *e.g*. recognizing small objects in a non-standard context.

Ladicky *et al*. [72] combine object detection with image segmentation, *i.e*. the pixel-precise labeling of the image with object labels, in a conditional random field (CRF) [73]. Combining these methods results in a superior scene understanding (*c.f*. using each method separately) by detecting all objects in the scene as well as providing the location and pixel-precise spatial extent of *each* object. In contrast to that, object detectors are only able to report on the location of an object instance while image segmentation methods provide a pixel-precise labeling of the scene but do not distinguish between individual object instances.

1.4 Thesis Overview and Contributions

The development of powerful image descriptors as well as the adoption of methods from the machine learning community has yielded a steady improvement in recognition accuracy, as recorded by the PASCAL VOC challenge [32]. This thesis continues this line of work by contributing to both steps of the sliding window detection scheme: (i) feature extraction and (ii) classification.

The main contributions of this work are covered in Chapters 2, 3 and 4 of this the-

sis. Each of these chapters also reviews related work and reports experimental results to demonstrate the applicability of the proposed approaches. The thesis closes with concluding remarks and future work in Chapter 5. In the following we highlight the novel contributions of this thesis.

Implicit Color Segmentation Features

Chapter 2 proposes an extension to the popular Histogram of Oriented Gradient (HOG) features, in which local histograms of gradient orientation are computed in localized 'cells' of an image. Adjacent cells are aggregated and normalized to give 'blocks' with greater invariance to local lighting and spatial deformation.

By themselves HOG features do not take advantage of potentially useful instance-specific characteristics, such as the color of an object instance. As a result HOG features do not have the capability to leverage instance-specific color information.

We propose a new feature descriptor (CHOG) [96] in Chapter 2, which incorporates an implicit color-based segmentation of a *specific* object with respect to its background into the feature extraction process. In contrast to HOG features, these 'segmentation features' have the ability to exploit potential instance-specific foreground/background relationships in an image window. We experimentally demonstrate that the proposed feature descriptor offers superior performance over the previous state-of-the-art [23].

Shared Parts for Deformable Part-based Models

Chapter 3 introduces an extension to Deformable Part-based Models (DPM) [38], which represent the current state-of-the-art in object detection by describing an object as a collection of parts that can move around subject to spatial constraints. In addition, the DPM uses mixture models [63] to capture large variability of appearance of an object class.

A challenge of such models is the efficient usage of available training data: in the original scheme proposed by Felzenszwalb *et al.* [38], increasing the number of parts and mixture components leads to a decrease in available training data per mixture component and an increase in number of parameters.

In Chapter 3 we extend the formulation of such models in a way that allows for a more efficient use of limited training data by sharing parts among all components of the mixture model [97]. Through our experiments we demonstrate that the proposed models offer superior detection performance over models that do not share parts, while also reducing the number of parameters and consequently allowing for faster evaluation at test time.

Detection of Partially Visible Objects

In Chapter 4 we augment the sliding window framework with the ability to handle partial visibility of an object instance. Sliding window detectors generally consider the entirety of an object instance is visible. This is problematic as the detector expects to find salient features corresponding to the object of interest in the occluded region. The absence of these features in the occluded region can lower the overall score of the detection and thus cause the object to be left undetected.

In order to address this problem we propose to treat visibility of an object class as a latent (hidden) variable. The inference of this latent variable allows to predict the visible regions of an object instance. We introduce a mechanism that uses this prediction to compensate for the decrease in the detection score caused by occluded regions of an object.

In addition we also propose a novel non-maximum-suppression (NMS) scheme which takes partial visibility of an object instance into account. Our method incorporates the inference about the visible parts of the object into the NMS scheme and as a consequence allows detection of objects which occlude each other to large degree.

We experimentally demonstrate that the proposed method for partial visibility modeling yields stronger qualitative results compared to common object detection schemes in that we are able to report on the visible extent of an object instance. In addition the proposed method improves detection accuracy over a model which does not have the ability to model partial visibility of an object class.

CHAPTER 2

IMPLICIT COLOR SEGMENTATION FEATURES

2.1 Introduction

One of the most successful recent methods for object detection is that proposed by Dalal & Triggs [23]. They propose Histogram of Oriented Gradients (HOG), which combines an image descriptor capturing local gradient orientation with a linear support vector machine (SVM) classifier in a sliding window framework. The method proposed in this chapter was originally motivated for the task of pedestrian detection [96] and hence we provide experiments on the INRIAPerson dataset [23] as well as the PASCAL VOC 2010 dataset [30].

2.1.1 Chapter Contributions

In this chapter we extend the method of Dalal & Triggs by proposing a novel feature extraction scheme which incorporates segmentation cues in the feature descriptor. Figure 2.1 motivates the work undertaken in this chapter. While some objects exhibit specific colors



(a) sheep (b) car (c) car segmentation

Figure 2.1: Color can be useful for discriminating an object class from the background or other object classes, *e.g*. sheep are often white (a). Most object classes of the PASCAL VOC 2010 [30] appear however in many different colors *e.g*. the 'car' object class (b). The ability to know the color of an object class can potentially help to yield more discriminative features *e.g*. by computing a segmentation of the image (c) which in return can be used to extract stronger shape features of the object class.

such as sheep, which are usually white, others *e.g*. cars do not. Therefore in the former case color can be used a class-specific indicator while in the latter case instance-specific knowledge regarding the color of the object must be exploited. By being able to identify foreground and background (Figure 2.1 (c)), stronger discriminative features (*e.g*. shape) can be extracted and used for classification in a sliding window detector.

The main idea behind our approach is to compute *window-specific* features which adapt to local image characteristics – color of foreground (object) and background. This differs from conventional feature extraction methods *e.g*. HOG [23] which compute fixed descriptors independent of the local image statistics. By using such local estimates of foreground and background statistics, a 'soft segmentation' of an image region is computed. Gradients in this segmentation image amplify edges between object and background, while reducing clutter from variation in texture and shading.

A key contribution of this chapter is that the soft segmentation is obtained in a computationally efficient manner by local linear projections of the original image, such that the approach is suitable for use in a sliding window scheme.

While exploiting color information, the proposed method does not assume any *class*-level color model, so it is applicable to object categories *e.g*. people/vehicles which vary

greatly in color. Compared to other methods which have exploited color-based segmentation to aid detection [117, 105], our method incorporates segmentation cues directly in the feature extraction and learning process, rather than as a separate post-processing step.

Chapter Outline

Section 2.2 introduces related work. The HOG descriptor, on which we base our method, is reviewed in Section 2.3. Section 2.4 describes the proposed approach. We report experimental results in Sections 2.6 and 2.7. Conclusions are offered in Section 2.8.

2.2 Related Work

In this section we first present related work on the pedestrian detection task. Further related work on the more general object detection task will be discussed in Chapter 3. Specifically this section focuses on Histograms of Oriented Gradient (HOG) features and extensions thereof because our method is also based on HOG features. In addition, we present recent developments in feature development and engineering. A very detailed overview of state-of-the-art approaches to pedestrian detection is provided by Dollar *et al*. [27]. The second part of this section introduces two specific pieces of work that motivate the research in this chapter.

2.2.1 Pioneering Work

Most, and indeed the most successful, approaches to pedestrian detection can be considered sliding window classifiers or template matchers. We have explained sliding window detectors in Section 1.3 and briefly summarize: a classifier is applied to every window of an image over multiple scales, yielding a positive (pedestrian) or negative (non-pedestrian) output for each window.

The pioneering work on chamfer matching by Gavrila & Philomin [52] deals with the task of developing a real-time pedestrian detection system to be used in a vehicle. Chamfer matching can be viewed as sliding window classification, using a threshold on distance to a hierarchy of edge templates as the classifier.

Early work by Papageorgiou and Poggio [100] uses a wavelet representation of a window and an SVM classifier for the task of pedestrian detection. Their work is also motivated by the specific application scenario as a ‘driver assistance system’ in a car.

Viola *et al*. [139] applied AdaBoost learning [46] and Haar-like features, essentially building on previous experience in face detection [138], and adding features computed over several frames to capture dynamic cues in videos. Haar-like features, as used by Viola & Jones [138] and introduced by Papageorgiou *et al*. [99], measure the difference between the sum of grayscale values in two rectangular regions of the window. As there are many possible combinations of such rectangular regions, even for small windows, Viola & Jones use a cascade architecture [46] (Section 1.3.4) to select the best set of descriptors during learning of the classifier.

2.2.2 Histograms of Oriented Gradients

The Histogram of Oriented Gradients (HOG) feature descriptor proposed by Dalal & Triggs [23] has proven particularly successful, achieving perfect performance on the MIT database used in earlier work [100], and this has sparked new interest in the pedestrian detection problem. We briefly discuss the method here and fully review it in Section 2.3 since we build upon it. Within a window the HOG descriptor computes local histograms of gradient orientation in a set of square 'cells'. These cells are laid out on a regular grid with adjacent cells aggregated and normalized to give 'blocks' with greater invariance to local lighting and spatial deformation.

HOG features are closely related to SIFT (Scale-Invariant Feature Transform) descriptors [83], which also compute local histograms of gradient orientation in a set of square cells but do not further organize them into a block structure. While Dalal & Triggs compute HOG features in a dense arrangement for an entire window, SIFT features were originally proposed as an image patch representation for matching scale-invariant keypoints.

The computational efficiency of the Dalal & Triggs detector has been improved by Zhu *et al*. [148] by applying the cascade architecture popularized in Computer Vision by Viola & Jones [138] for the task of face detection – see also Section 1.3.4.

2.2.3 Feature Development and Engineering

Recent work on feature engineering has focused on combining HOG features with other types of features. Wang *et al*. [140] train global pedestrian detectors by combining gradient (HOG) and texture features (Local Binary Patterns, LBP) [91]. In addition to combining these feature channels their method considers occlusion of the object class, computing a patch-wise response map for a global detector by 'splitting' the linear classifier coeffi-

cients into patches and segmenting this response map using a mean shift algorithm [20] to infer occluded regions. Their method yields a significant improvement in detection accuracy on the INRIAPerson dataset [23].

Schwartz *et al*. [115] combine gradient, texture and color features into a sliding window framework, showing that multiple feature channels can increase detection accuracy. They use Partial Least Squares (PLS) analysis [143] to reduce the dimensionality of their feature descriptors. The low dimensionality of the final feature vector allows them to classify each window using a non-linear SVM, yielding a significant performance improvement compared to a linear SVM.

Tuzel *et al*. [131] have reported improved results using a method which describes an image window in terms of the covariance matrices of features based on intensity derivatives. These covariance matrices lie on a Riemannian manifold and Tuzel *et al*. propose an adapted LogitBoost [47] classifier to perform classification on these manifolds. Both Tuzel *et al*. [131] and Schwartz *et al*. [115] recognize the need to combine advanced feature descriptors with a non-linear classifier to achieve good detection performance.

Vedaldi & Zisserman [136] suggest approximating the feature maps of the additive kernels of an SVM classifier to achieve good performance along with execution speed comparable to that of linear classifiers. Kernel SVMs, which are explained in Section 2.5.2, use feature maps to 'lift' a window's feature descriptor to a high dimensional space, in which classification is implicitly performed with the help of a kernel function. Computing the kernel function can be slow and hence the method proposed by Vedaldi & Zisserman derives approximate finite-dimensional feature maps which can be used in conjunction with a linear SVM and often have indistinguishable performance from regular kernel SVMs on standard datasets [35, 23, 90].

In this chapter we also explore the possibility of non-linear classifiers and combine the proposed feature descriptors with an inhomogeneous polynomial kernel [116].

2.2.4 Deep Learning

An alternative to engineering feature descriptors by hand is to automatically learn them. Convolutional neural networks [76] adopt such an approach by convolving the window's content with a set of *learned* filters. These filters often correspond to edglet-like features and encode a certain degree of learned spatial invariance. The output of the convolution of the window with one filter can be used as an input to another convolution, hence giving rise to an architecture comparable to a neural network [9]. While convolutional neural

networks have originally been introduced using a supervised training paradigm (*i.e*. the classification of a window into object/non-object) by employing the back-propagation algorithm [108], unsupervised approaches have recently also been used [58, 106]. Such approaches focus on learning a feature representation for a set of windows independent of a specific task.

Convolutional neural networks are part of the research area of Deep Learning [57, 8], which considers the learning of 'deep architectures' for modeling high-dimensional structured data. Such data is often encountered in Computer Vision tasks such as object detection or action recognition [75] but also, for example, in the processing of audio streams [120, 77]. Deep learning approaches for object detection can be seen as sliding window detectors with a learned feature extraction scheme. While this is theoretically appealing, hand-crafted feature extraction schemes still provide better detection accuracy in practice *e.g*. in pedestrian detection benchmarks [69].

2.2.5 Pedestrian Detection by Pose Estimation

In addition to enriching or extending the feature vector per window, various authors have proposed to improve pedestrian detection by combining it with efficient pose estimation techniques.

Work by Tran & Forsyth [128] employs structured learning to incorporate detailed 2D pose estimation in the detection process, estimating pose for every image window (positive or negative). The proposed model combines pose-specific image descriptors with a global window descriptor and uses an SVM to reach a classification decision.

Lin & Davis [82] have recently proposed a scheme which extracts instance-specific features based on template matching. Edge templates are used to build a part-template tree, which can be parsed to infer the pose of a hypothesized pedestrian. Provided with this pose the method of Lin & Davis extracts only features covered by the pedestrian and passes them to an SVM classifier.

Okada & Soatto [92] proposed to divide the pedestrian class into disjoint sub-classes by pose, and train individual classifiers for each. They report modest improvements on the detection problem, and apply the method to improve regression-based pose estimation.

Felzenszwalb *et al*. [38] also estimate the position of parts, but use semi-automatically learned parts rather than manually defined limbs. This method gives very promising results on the challenging PASCAL VOC databases [32] and has more recently shown promising results on the INRIAPerson dataset [53]. In Chapter 3 of this thesis we pro-

pose extensions to the work of Felzenszwalb *et al*. [38].

2.2.6 Caltech Pedestrian Dataset

Dollár *et al*. [26] presented a pedestrian detection benchmark comparing multiple detectors on the new Caltech Pedestrian dataset. The new dataset is significantly larger than the INRIAPerson dataset [23], which is used for evaluation purposes in this chapter, and additionally suggests a new per-image performance evaluation measure. Compared to the per-window measure popularized by Dalal & Triggs [23], their measure is more robust w.r.t. possible artifacts at the border of an image but requires an additional non-maximum suppression (NMS) procedure to produce a final set of detection per image.

In this chapter we will use both per-window [23] and per-image [32] performance evaluation on the INRIAPerson dataset to test the proposed novel feature descriptor.

2.2.7 Motivation

The proposed method is inspired by two pieces of work which incorporate segmentation cues for object detection and tracking.

Using Segmentation to Verify Object Hypotheses

Ramanan [105] proposed a scheme for 'verification' of detections from a sliding window detector by segmenting the window and verifying if the segmentation resembles the object class of interest *e.g*. a pedestrian. An overview of the method is shown in Figure 2.2.

Initially all windows in an image are evaluated using a regular Dalal & Triggs detector and only windows that score above a specific threshold are analyzed further. These windows are then segmented using a weak prior on object shape learned from unsegmented training images, representing foreground/background color distributions using a color histogram, and applying a GraphCut segmentation method [13]. A linear classifier is applied to the resulting binary segmentation mask to verify the hypothesized detections. The method can be seen as a 'post-processing' filter on the detections – training of the sliding window detector is performed independently of the segmentation process.

The proposed method is suboptimal as detection and segmentation are performed separately. It is also computationally demanding because GraphCut [13] has to be performed for many windows.

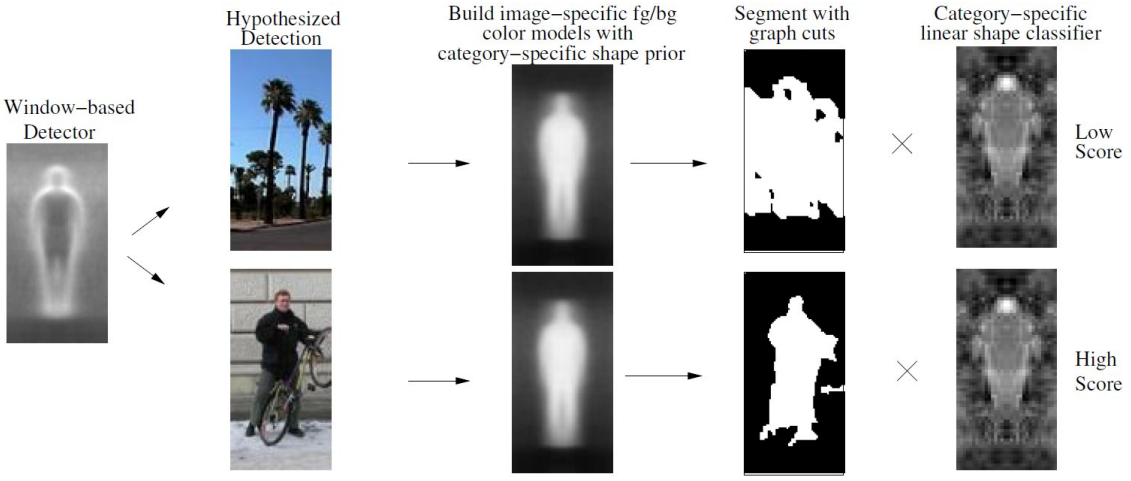


Figure 2.2: The 'verification by segmentation' method of Ramanan [105]. As a first step a regular Dalal & Triggs detector is applied to every window, yielding hypothesized detections. Segmentations based on a weak initial prior are then computed for those windows scoring above a specific threshold. The resulting segmentation masks are passed to a catergory-specific linear classifier which derives a score and hence verifies the object hypothesis. Figure reproduced with permission from Ramanan [105].

We use the idea of incorporating instance-specific appearance into the detection process, but integrate this into the classifier learning. Our method is highly efficient, allowing us to create segmentations at the window level without the use of an initial filtering detector.

On-Line Selection of Discriminative Tracking Features

Collins *et al*. [19] propose a method for extracting discriminative features for tracking which are adapted to the surroundings of the object to be tracked in each frame.

The approach is particularly elegant in its simplicity: given the bounding box of the object and a larger bounding box capturing the background appearance in its immediate neighborhood, the method picks between a set of pre-defined linear transformations of the RGB color space, choosing the transformation which maximizes discrimination between the object and background region in the current frame. This transformation is then applied to pixels in the next frame (assuming some coherence across frames), obtaining better foreground/background discrimination with which to drive the tracker.

We exploit this idea of using different transformations of the color space to 'pull out' stronger foreground/background features, applying the approach to image windows.

2.3 Histogram of Oriented Gradients

In this section we review the HOG descriptor and detection scheme proposed by Dalal & Triggs [23].

2.3.1 The HOG Descriptor

The feature extraction scheme is visualized in Figure 2.3. HOG descriptors are extracted for every window separately – see Figure 2.3 step (a). In the original scheme of Dalal & Triggs [23] the image is first gamma- and color-normalized (b). In our implementation we skip this step as we have not found any noticeable increase in detection accuracy from it.

Accumulation of Gradient Responses

For each pixel within the detection window the image gradient is computed – this is visualized in step (c) of Figure 2.3. In Section 2.3.3 we explain how these gradients are computed and further note why the method suggested by Dalal & Triggs can be problematic.

In the next step (d) gradient responses are accumulated into bins over (i) orientation, and (ii) spatial regions ('cells'). An individual cell and the grid of cells is shown in Figure 2.3 (d). Within each cell, a histogram of gradient orientation is computed by quantizing the gradient orientation and casting votes of the magnitude into histogram bins. These votes are usually bi-linearly interpolated *i.e*. the magnitude is shared between neighboring bins. Although many schemes for dividing the image into cells *e.g*. square cells or log-polar schemes have been investigated [23], using simple square cells is shown to be effective as well as computationally efficient [148].

The intuition to binning (quantizing) orientation and spatial position is to introduce some invariance to local image deformation – object parts can shift around in each cell to some degree without changing the HOG descriptor significantly.

Blocks and Contrast Normalization

A second stage of spatial accumulation (step (e) in Figure 2.3) groups contiguous ranges of cells into 'blocks'. The descriptor for each block is then independently normalized to have constant norm by dividing each histogram bin in the block by the ℓ_2-norm of

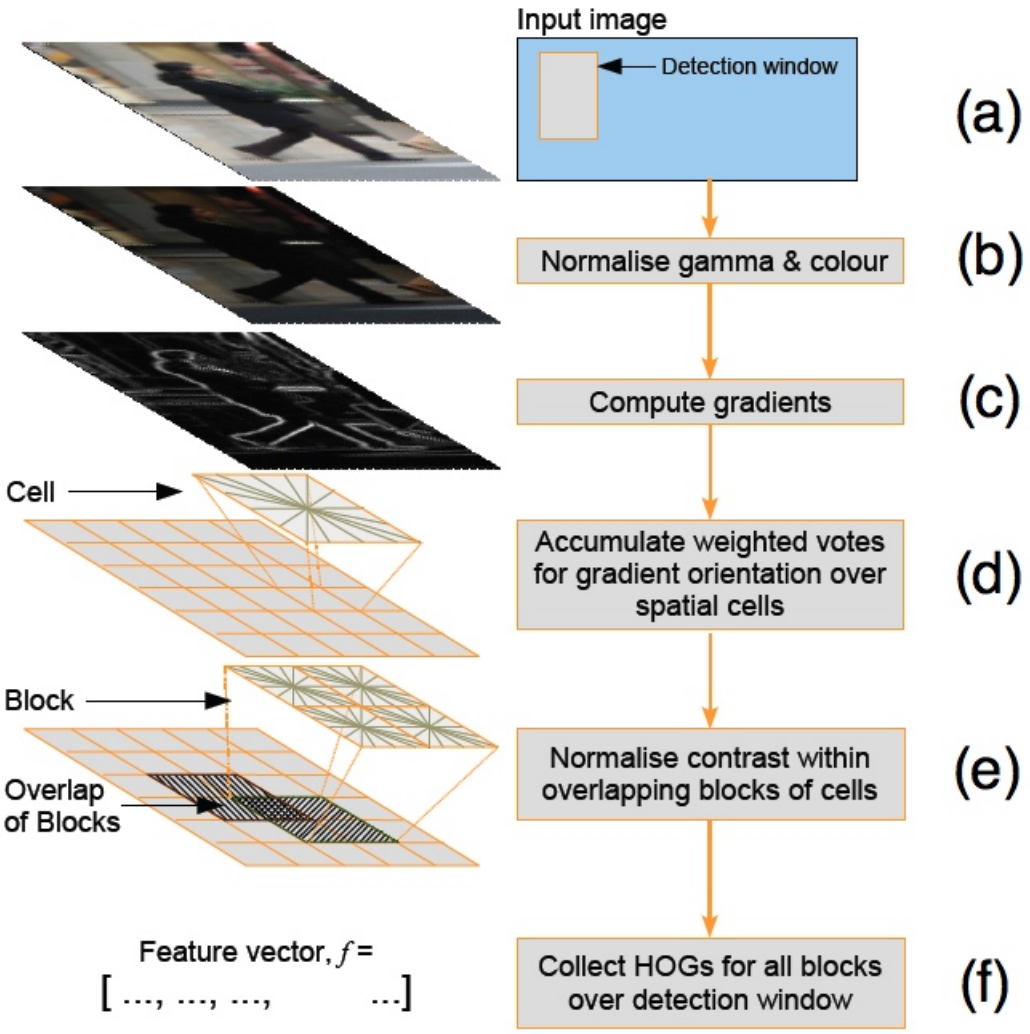


Figure 2.3: Histograms of Oriented Gradient (HOG) feature descriptors as proposed by Dalal & Triggs [23]. For a given input window (a) color normalization is applied (b) and the gradient of the image is computed (c). Gradient responses are then pooled w.r.t. orientation and spatial region (d), giving cells (orange grid) with histograms of oriented gradient. As a final step these cells are grouped to form blocks which are then normalized to give contrast invariance (e). Figure reproduced with permission from [22].

all bins in the block . The blocks typically overlap by one or more cells such that each cell is represented multiple times (with different normalizations) in the final descriptor formed by concatenating all blocks. The block normalization scheme gives some contrast invariance over a larger scale than that of the individual cells.

Dalal & Triggs [23] investigated various schemes of cell size, shape and spacing, block arrangement, orientation binning and normalization. For the task of pedestrian detection in the INRIAPerson dataset [23], image windows are 128×64 pixels. A reasonable scheme is to use blocks of 2×2 cells overlapping by one cell in each direction, square cells of width 6–8 pixels, 9 orientation bins (discarding gradient sign) and ℓ_2 normalization with a constant term added to avoid amplifying noise in regions with very low variance.

2.3.2 Dimensionality Reduction

HOG features are usually very high-dimensional: a 2×2 block of cells with each cell using 9 orientation bins consists of $4 \times 9 = 36$ components. Since blocks usually overlap in the window, the number of features per window can grow quickly. In our scheme a window of size 128×64 pixels consists of 15×7 blocks (cell size of 8 pixels) and hence gives feature vectors with $15 \times 7 \times 4 \times 9 = 3,780$ components.

Inspired by Felzenszwalb *et al*. [38], we propose a block-based dimensionality reduction method motivated by Principal Component Analysis (PCA) [101, 10]. For each block the corresponding gradient orientation bins of all cells belonging to this block are accumulated, essentially leading to 9 orientations bins per block. In addition, all orientation bins within a cell are also accumulated, providing a per-cell 'energy'. This leads to a $9 + 4 = 13$-dimensional feature vector per HOG block – a significant reduction from the original 36-dimensional feature vector. For a window of size 128×64 the number of components reduces from $3,780$ to $1,365$. We have established experimentally that the loss in detection accuracy is insignificant when using the reduced feature vector compared to the 'full' vector per block. However, because the per-block dimensionality is much lower, convolution in the sliding window detector can be executed faster with the reduced feature set.

We use this dimensionality reduction technique *throughout the thesis* for all experiments on the PASCAL VOC 2010 dataset [30] and also apply it to the novel feature descriptor proposed in this chapter.

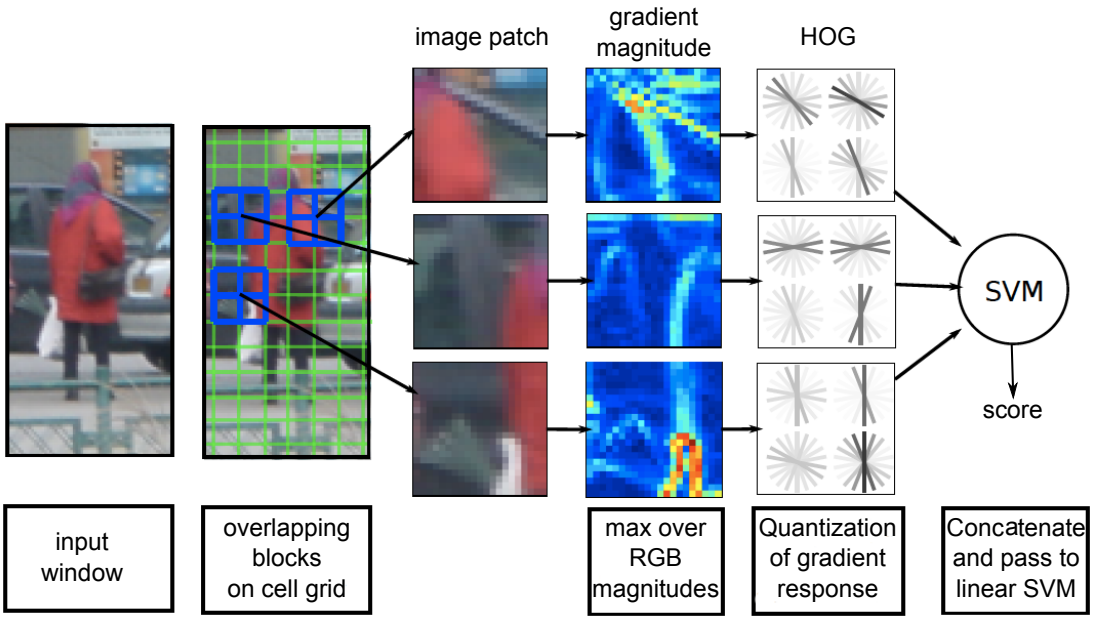


Figure 2.4: Detection pipeline for HOG features. Features are extracted on a cell grid (green) and then independently normalized for overlapping blocks (blue). Image gradients are computed by picking the gradient from the color channel with the maximum magnitude. This can potentially increase noise as visualized *e.g.* the edge of the car is dominant in the maginitude of the top block resulting in HOG features that do not precisely represent the outline of the pedestrian. Once HOG features are computed, they are passed to a linear SVM which gives a confidence score of the entire window.

2.3.3 Gradient Computation

The HOG scheme can be applied to gray-scale images since it makes use of the image gradient alone. However, when applying the descriptor to color images, Dalal & Triggs' implementation computes the gradient at a pixel in each of the red, green and blue channels, and selects the response with greatest magnitude [23].

This gradient extraction scheme can potentially increase the sensitivity to edges in color images which correspond to changes in color with no associated change in luminance. However, because the decision is made independently at every pixel it can also have the effect of increasing noise – an example is shown in Figure 2.4 in which the background pixels contribute a disproportionate amount of magnitude to the gradient representation.

The gradient extraction scheme does not enforce any 'coherence' in the edges, *i.e.* as gradient responses are collected from different color channels, edges might not be represented coherently in these channels and consequently in the final feature descriptor.

In addition the method is agnostic to dark/light *vs.* light/dark transitions because the unsigned gradient is typically used, and again cannot capture the likelihood that nearby edges are likely to be of the same sign. Our proposed method (Section 2.4) addresses some of these limitations.

2.3.4 Classification

Dalal & Triggs [23] paired the HOG descriptor with a Support Vector Machine (SVM) [10, 133] classifier in the sliding window framework and investigated the use of linear and nonlinear kernels [116]. Competitive results were obtained using a linear SVM which is appealing because of its computational efficiency. We explain SVMs and kernels in Section 2.5.

2.4 Color Histograms of Oriented Gradient

This section describes the proposed feature extraction scheme. The essence of the scheme is to extract a descriptor which captures the shape of the foreground object (if any) in a given window. The extraction scheme consists of two main steps: (i) soft segmentation into foreground/background; (ii) describing edges in the soft segmentation using a HOG descriptor. Figure 2.5 visualizes the proposed feature extraction scheme and gives a more

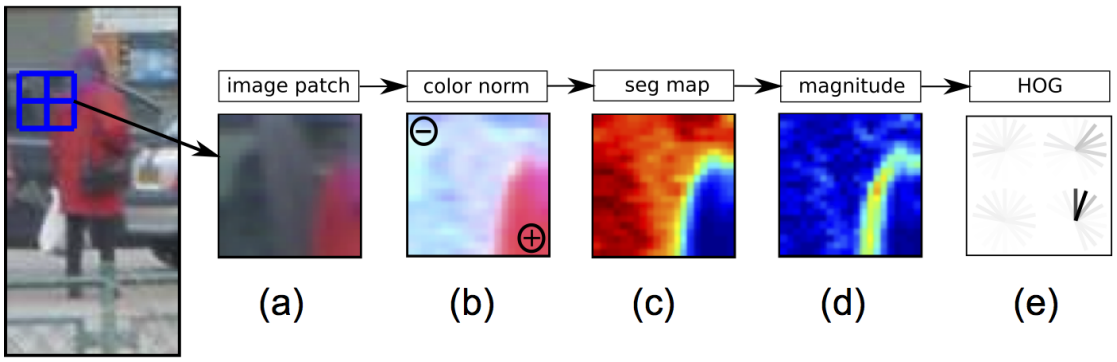


Figure 2.5: Extraction scheme for CHOG features. For a given input window (left) the image region (a) is color-normalized (b) (Equation 2.9) and a segmentation image (c) is derived given the two seed points marked as '+' and '-' in (b) using the method described in Section 2.4.1. In (d) we derive the gradients (and magnitudes) following Equation 2.8. In the final step (e) we compute HOG features with signed gradients on the segmentation image which yields the CHOG features.

detailed overview.

We refer in the following to our features as *'CHOG'*, short for *Color Histograms of Oriented Gradient*, to emphasize the extension of the HOG descriptor with color information. Figure 2.9 compares HOG and CHOG features.

2.4.1 Soft Segmentation by Fisher Discriminant

Given an image region we can think of segmenting pixels in the region into foreground and background. We reason that such a segmentation *i.e.* the object silhouette gives strong cues to recognition [105].

Let us assume that we are given a sample of RGB pixel values $\{\boldsymbol{p}^i\}$ where $i \in \mathcal{C}_1$ denotes background pixels and $i \in \mathcal{C}_2$ denotes foreground pixels. Samples are taken from a particular image window, so the foreground distribution captures instance-specific properties (this person is wearing blue trousers) rather than class-level properties (sheep are white) – see Figure 2.6 (a).

We seek a linear projection of the pixel values which maximizes the separation between foreground and background. The projection is given by

$$s(\boldsymbol{p}) = \boldsymbol{w} \cdot \boldsymbol{p} \tag{2.1}$$

Assuming that the distributions of foreground and background pixels are Gaussian, the Fisher criterion [10] gives a suitable means for choosing $\boldsymbol{w}$:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{\Sigma}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{\Sigma}_W \boldsymbol{w}} \tag{2.2}$$

where $\boldsymbol{\Sigma}_B$ is the *between-class* covariance matrix, given by

$$\boldsymbol{\Sigma}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1) \cdot (\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{2.3}$$

where $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are the sample means of the background and foreground classes respectively, and $\boldsymbol{\Sigma}_W$ is the total *within-class* covariance matrix, given by

$$\boldsymbol{\Sigma}_W = \sum_{i \in \mathcal{C}_1} (\boldsymbol{p}_i - \boldsymbol{m}_1) \cdot (\boldsymbol{p}_i - \boldsymbol{m}_1) + \sum_{i \in \mathcal{C}_2} (\boldsymbol{p}_i - \boldsymbol{m}_2) \cdot (\boldsymbol{p}_i - \boldsymbol{m}_2) \tag{2.4}$$

The value $\hat{\boldsymbol{w}}$ maximizing $J(\boldsymbol{w})$ can readily be shown to be:

$$\hat{\boldsymbol{w}} \propto \boldsymbol{\Sigma}_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{2.5}$$

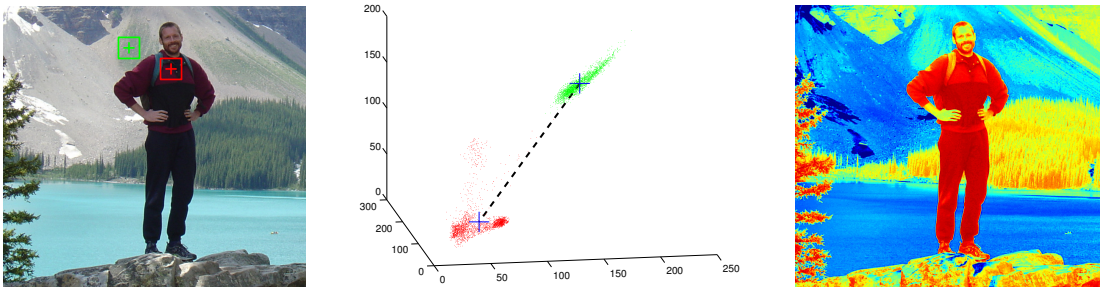
This value of $\hat{\boldsymbol{w}}$ can also be thought of as a classifier, learning to distinguish between pixels in the foreground and pixels in the background.

Soft Segmentation

We denote the original image as I and an individual pixel by $\boldsymbol{p}$. Projecting the original image pixels by

$$s = \hat{\boldsymbol{w}} \cdot \boldsymbol{p} \tag{2.6}$$

gives an image S which can be considered a 'soft segmentation' (Figure 2.6) – large values indicate high probability of foreground, and small values high probability of background. Gradients in this image give strong evidence for edges between foreground and background.



(a) Input Image and seed regions $\mathcal{C}_1$, $\mathcal{C}_2$

(b) Foreground and background color clouds and means

(c) Segmentation map

Figure 2.6: Local segmentation model used by our method. (a) Two seed points are selected, one in the foreground (red) and one in the background (green). Our approach computes the mean color values in close proximity of the points (rectangles). (b) Blue crosses show the means of each point clouds in RGB color space. The difference of the two means yields a projection from foreground to background (black dashed line). (c) This projection is applied to every pixel in the image. In the resulting segmentation map high values (warm colors) indicate foreground and low values (cold colors) indicate background.

A Simplified Model

Equation 2.5 could directly be applied to obtain soft segmentations, but is somewhat computationally expensive because of the need to compute and invert the covariance matrix $\mathbf{\Sigma}_W$. If we assume that the covariance of both foreground and background distributions is isotropic *i.e.* $\mathbf{\Sigma}_W$ is proportional to the identity matrix, then the optimal $\boldsymbol{w}$ simplifies to the difference between the class means:

$$\hat{\boldsymbol{w}} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{2.7}$$

In the following we use this simplified model, and justify its use in Section 2.4.2. Figure 2.6 shows an example segmentation using the proposed (simplified) method.

Projection of Gradients

Our aim is to compute HOG features from the soft segmentation. This requires computation of gradient orientation and magnitude at each pixel. Since computation of the gradient is a linear operation, the gradient of the segmentation image can simply be computed by linear transformation of the original RGB gradients. Given the original image I

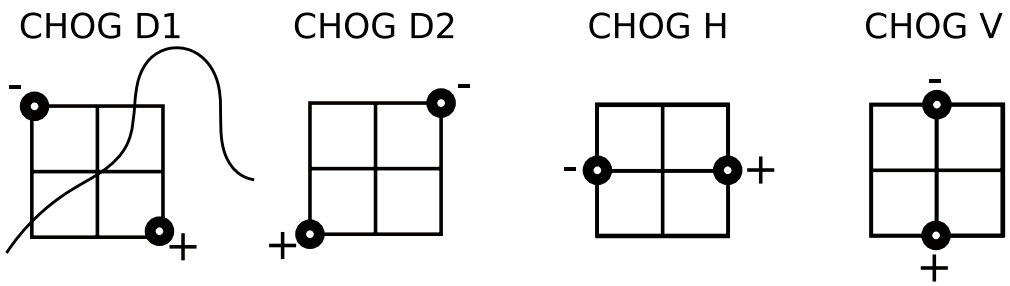


Figure 2.7: Potential reference points for a single block. The 'foreground' color distribution is estimated from pixels in the neighborhood of the +ve point, and 'background' from the -ve point. The resulting projection is used for all pixels within the block. Different combinations of reference points are useful in the description of different parts of the object. The first set of reference points (CHOG D1) is potentially useful for the shoulder region of a pedestrian as shown in the sketch.

we have

$$\frac{\partial}{\partial x} S = \boldsymbol{w} \cdot \left\langle \frac{\partial}{\partial x} I_R, \frac{\partial}{\partial x} I_G, \frac{\partial}{\partial x} I_B \right\rangle \tag{2.8}$$

and similarly for the gradient in the y-direction.

The relevance of this identity is that the gradient of the image under any projection can be computed without needing to apply the convolution operator to the *projected* image. This means that the gradient of any projected image can be computed from the gradient of the original RGB image. Since, as described below, a large number of projections are used for every image window, this enables the feature extraction to be performed in a computationally efficient manner. We refer to the CHOG features as using 'implicit' segmentation since the gradients of the soft segmentation used to construct the descriptor are computed without explicitly computing the segmentation.

2.4.2 Semi-local Segmentation

Thus far we have assumed that we are given a set of sample pixels labeled as foreground or background. Such samples could be obtained by using an average person mask [105] to be applied to each window. However, for negative windows this can result in 'hallucinating' pedestrian regions. Additionally, the Gaussian assumption made in the Fisher discriminant will be a poor model when there is significant variation in color within *e.g.* the background – Ramanan [105] uses a color histogram, which prevents applying the

method to every window because of computational expense.

Semi-local Projections

We use a number of *semi-local* projections rather than a single global sample of foreground and background [105]. For all HOG blocks, a set of reference pairs of image points chosen at training time are defined as positive (potential foreground) and negative (potential background).

For a given window the means of the 'foreground' and 'background' distributions are estimated from pixels in the neighborhood of these reference points. Since a HOG scheme is used to describe the soft segmentations, we base the selection of reference points on HOG blocks. Figure 2.7 shows the selected potential reference points for a block: four different combinations (horizontal, vertical and two diagonal combinations) are used to fully cover the range of possible appearance of object edges.

The two diagonal combinations of reference points can model the shoulder region of a pedestrian as shown by the pedestrian outline underneath CHOG D1 in Figure 2.7. The horizontal reference points are ideal to model the legs of a pedestrian. Additional examples of blocks with reference points are shown in Figure 2.9.

For each block in the window *and* each combination of reference points, the discriminant projection is computed using Equation 2.7 while the gradients of the soft segmentation within the block are computed using Equation 2.8. These gradients are represented in the descriptor by a conventional HOG block *i.e*. histograms over orientation in the 2×2 cells of the block.

Using semi-local projections in this way has two advantages over attempting a global segmentation of the window: (i) the Gaussian assumption enforced by the Fisher discriminant [10] is likely to be well-satisfied given the local color measurements, enabling use of the simple linear projection scheme for implicit segmentation and gradient computation; (ii) by choosing projections at training time the method can also capture selected salient internal gradients, *e.g*. between shirt and trousers, which is lost in the object silhouette. Section 2.4.4 offers further discussion.

2.4.3 Color Normalization

When using RGB pixels to form the soft segmentation, a significant component of the discriminant projections (Equation 2.7) is difference in intensity. We found that results were improved by using intensity-invariant features, which give complementary cues to

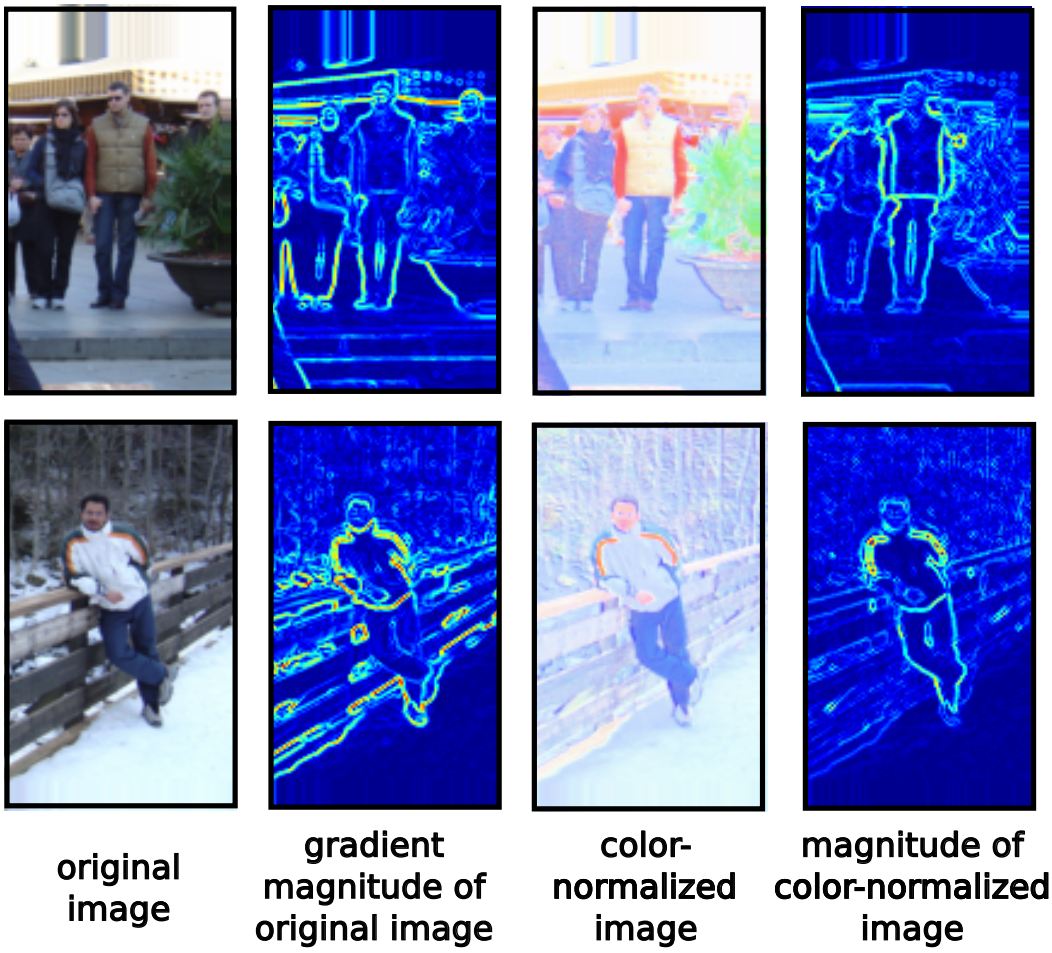


Figure 2.8: Color normalization used by the proposed scheme. Note how the color normalization is able to suppress background clutter due to intensity invariance and 'pull out' the different gradients of the pedestrian. To compute gradient magnitudes for this visualization the maximum over individual RGB gradient magnitudes was used as suggested by Dalal & Triggs [23].

the intensity gradients used by the HOG descriptor. Prior to computation of projections *and* gradients, the RGB image pixels $\boldsymbol{p} = \langle p_r, p_g, p_b \rangle$ are normalized thus:

$$\boldsymbol{p}' = \frac{\boldsymbol{p}}{\max(p_r, p_g, p_b) + \epsilon} \tag{2.9}$$

where ϵ is a small constant, avoiding unstable results for pixels of low intensity.

Figure 2.8 shows the effect of color normalization on two example images. Note how the gradients of the color normalized image 'pull out' additional edges which are not prominent in the gradients of the original image. The gradient cues are very often complementary *e.g.* the trousers of the second person (lower example in Figure 2.8) are not visible in the gradients of the original image but appear strongly in the gradients of the color normalized image.

2.4.4 Discussion of CHOG

Figure 2.9 shows example output of the feature extraction method for both HOG and CHOG. We show the conventional HOG descriptor using max-RGB gradient (see Section 2.3), and the CHOG descriptor using the gradient computed on the soft segmentation including color normalization (Section 2.4.3).

Stronger Gradient Cues

Note how in the top-left inset CHOG gives a strong response to the boundary of the coat, and suppresses responses to shading and background clutter. In the top-right inset CHOG substantially attenuates the gradients introduced by background clutter and emphasizes the pedestrian's edge.

In the bottom-left inset CHOG features are not confused by the white bag the pedestrian is carrying but give a clean outline of the pedestrian's shape. This is in contrast to HOG features which result in strong gradient responses around the bag and weaker gradient responses for the pedestrian outline.

For the patch region shown in the top-right inset CHOG is able to substantially reduce the amount of background clutter and emphasize the edge belonging to the pedestrian *c.f.* HOG which gives strong gradient responses in the background and results in 'noisy' feature representations.

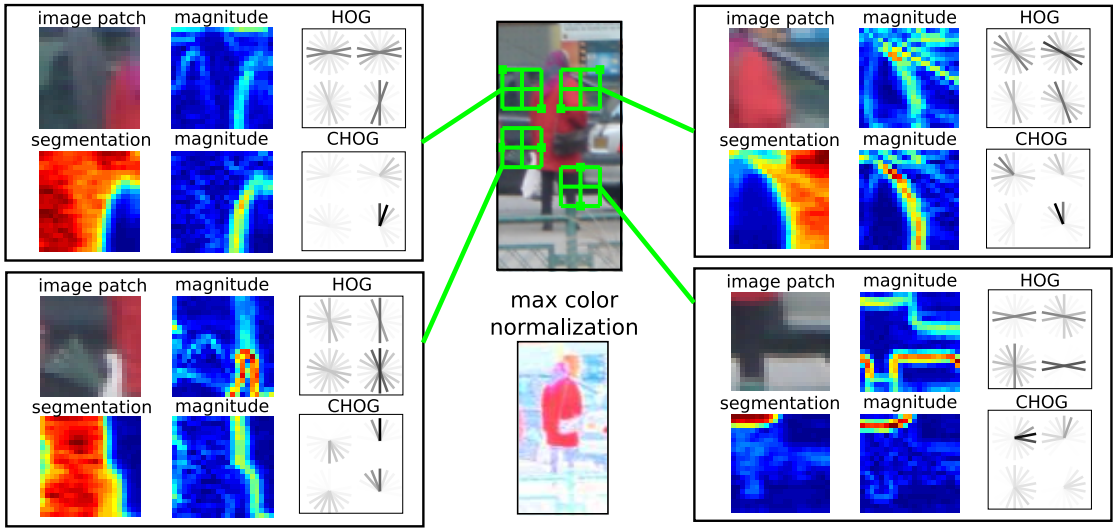
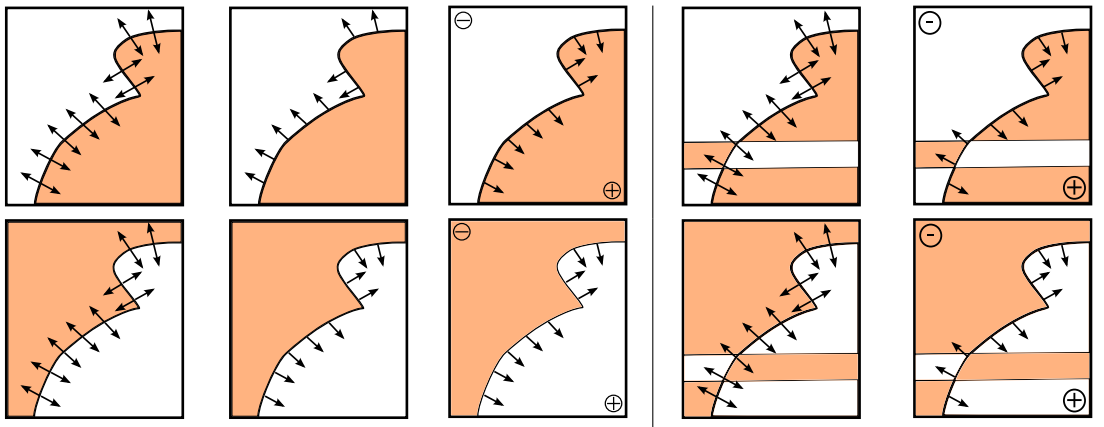


Figure 2.9: Comparison of feature descriptors. For each block of the window (center top) a HOG descriptor and CHOG descriptor is computed (insets, right column), each consisting of a histogram over orientation. CHOG descriptors are computed on a soft segmentation of pixels in the block into foreground and background (insets, bottom left). For each block, hypothesized foreground and background samples are taken at reference points relative to the block (center top). Color normalization (center bottom) is used to enhance color edges in the CHOG descriptor complementary to the intensity edges in the HOG descriptor. Note that the CHOG features emphasize the edges of the pedestrian while attenuating shading on the clothing and edges due to clutter in the background.



(a) HOG unsigned (b) HOG signed (c) CHOG signed (d) HOG unsigned (e) CHOG signed

Figure 2.10: Additional modeling capacity of CHOG features. (a) HOG features usually use an unsigned gradient and consequently are not able to distinguish between dark object / light background (top row) and dark background / light object (bottom row) since the gradient sign is discarded; (b) If a signed gradient is used, HOG features distinguish between both cases but provided with the different brightness relationships between object and background for both cases, the final (linear) classifier must assign equal weight to both gradient signs; (c) CHOG features use a signed gradient and consistently capture background-to-foreground transitions, allowing coherence of gradients to be learned by weights for a single gradient sign; (d) HOG features are not able to identify inconsistent color transitions along an edge while (e) CHOG features identify the color switch along the edge.

Additional Modeling Capacity

Figure 2.10 shows the modeling capacities of HOG and CHOG descriptors for the common case where the foreground object may be brighter or darker than the background. By using consistent projection of the image gradients in a foreground-to-background direction CHOG features combine the invariance of HOG to the bright/dark relationship between object and background with the ability to capture coherence of gradient sign given locally coherent background intensity. This is important as it gives gradients of similar orientation for the two cases of dark/light and light/dark foreground and background (Figure 2.10 (c)) *c.f.* HOG with unsigned gradient, which does not have this capability (Figure 2.10 (a) & (b)).

Coherence in Edges

By use of consistently signed gradients CHOG is also able to distinguish between noisy edges in background regions and coherent gradients corresponding to true ob-

ject/background edges. We are therefore able to model color consistency along an edge: if the color configuration along an edge switches, CHOG features will respond in a 'flip' of the gradient direction (Figure 2.10 (e)) while HOG features are not able to identify the color inconsistency ((Figure 2.10 (d)). This is important because HOG features tend to hallucinate the presence of an object in highly textured regions, *i.e*. regions with strong gradient responses.

2.5 Classification

We create a composite descriptor by concatenating HOG and CHOG blocks (see Figure 2.9). Combining the two is effective since the gradient information from the original and intensity-invariant soft segmentation images is complementary as shown in Figure 2.8.

2.5.1 Support Vector Machines

Dalal & Triggs [23] paired the HOG descriptor with a Support Vector Machine (SVM) [133, 10] and we follow this approach.

To classify an individual feature vector $\boldsymbol{x}$ the SVM uses a linear function:

$$h\left(\boldsymbol{x}\right) = \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{2.10}$$

where $\boldsymbol{w}$ is a vector modeling the appearance of an object class and b is a bias term. Throughout the thesis we refer to function $h\left(\cdot\right)$ as a *detector*.

The training algorithm of an SVM searches for the maximum separating hyperplane $\boldsymbol{w}$ of positive and negative training data, which is represented by the training dataset

$$\mathcal{T} = \left\{\left(\boldsymbol{x}^1, y^1\right), \left(\boldsymbol{x}^2, y^2\right), ..., \left(\boldsymbol{x}^n, y^n\right)\right\} \tag{2.11}$$

where $\boldsymbol{x}^k$ is the feature vector of the k'th training example. $y^k \in \{+1, -1\}$ is the class of the window (either object or non-object) and n is the number of training examples.

SVM training minimizes a loss function to obtain the vector $\boldsymbol{w}$ which separates the training data. $\boldsymbol{w}$ is usually regularized to avoid over-fitting of the detector on the training data. We express the training of SVM as an energy minimization problem with the

following energy function:

$$E\left(\boldsymbol{w}, b\right) = \frac{\lambda}{2}\left\|\boldsymbol{w}\right\|^2 + \sum_{k=1}^{n} L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) \tag{2.12}$$

where $\left\|\cdot\right\|^2$ is the ℓ_2-norm and $L\left(\cdot\right)$ is a loss function. SVMs use the hinge loss

$$L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) = \max\left\{0, 1 - y^k h\left(\boldsymbol{x}^k\right)\right\} \tag{2.13}$$

The parameter λ in Equation 2.12 sets the relative importance of the regularization term w.r.t. the loss term. It effectively defines a tradeoff between generalization (low norm of $\boldsymbol{w}$) and the ability to correctly predict the training data (loss term). The hinge loss satisfies a max-margin definition: only training examples that score inside the margin (*i.e*. the hinge loss is not zero) contribute to the overall energy.

Learning a detector is equivalent to minimizing the energy function of Equation 2.12 w.r.t. the appearance vector $\boldsymbol{w}$ and the bias term b:

$$(w^*, b^*) = \arg\min_{\boldsymbol{w}, b} E\left(\boldsymbol{w}, b\right) \tag{2.14}$$

This optimization problem can for example be solved by deriving subgradients [59] of the energy function and 'plugging' these subgradients into a steepest descent optimization scheme, *e.g*. gradient descent [12] or L-BFGS [16]. The subgradients, which are necessary because the hinge loss is not differentiable at every position, are given by

$$\frac{\partial E\left(\boldsymbol{w}, b\right)}{\partial w_t} = \lambda w_t - \sum_{y^k h\left(\boldsymbol{x}^k\right) < 1} y^k x_t^k \tag{2.15}$$

$$\frac{\partial E\left(\boldsymbol{w}, b\right)}{\partial b} = - \sum_{y^k h\left(\boldsymbol{x}^k\right) < 1} y^k \tag{2.16}$$

i.e. only elements which lie inside the margin of the SVM contribute to the subgradients. In Chapters 3 and 4 we use this optimization procedure for an extended SVM formulation. In this chapter, however, we use the SVMLight [64] package, which optimizes a

constrained dual formulation [133] of the energy. We choose SVMLight over other methods in this chapter because it is significantly faster for SVMs with a kernel, which will be introduced in the next section.

2.5.2 Kernel Support Vector Machines

SVMs can be learned in feature spaces with higher dimensionality than the original feature space by the use of a kernel function. When training, such kernel SVM search for a linearly separating hyperplane in the higher dimensional feature space, effectively defining a non-linear separation measure in the original feature space.

A kernel SVM maps a feature vector $\boldsymbol{x}$ to a (usually) higher dimensional space $\mathcal{H}$ using a 'lifting' function $\phi(\cdot)$ and hence a detector is redefined as

$$h(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x}) \tag{2.17}$$

This lifting function is usually not explicitly computed but instead implicitly defined by representing the dot products of any two feature vectors $\boldsymbol{x}^1, \boldsymbol{x}^2$ in the higher dimensional space with the help of a kernel function [116]:

$$K\left(\boldsymbol{x}^1, \boldsymbol{x}^2\right) = \phi\left(\boldsymbol{x}^1\right) \cdot \phi\left(\boldsymbol{x}^2\right) \tag{2.18}$$

The kernel trick [10, 116] allows us to compute a detector response to a feature vector $\boldsymbol{x}$ as a weighted sum over the dot products of $\boldsymbol{x}$ and the training examples in the higher dimensional feature space. To do so the weight vector $\boldsymbol{w}$ is expressed as a weighted sum over all training examples in $\mathcal{T}$:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \tag{2.19}$$

The response of a detector to a feature vector $\boldsymbol{x}$ can now be expressed as

$$\begin{aligned} h(\boldsymbol{x}) &= \boldsymbol{w} \cdot \phi(\boldsymbol{x}) + b \\ &= \left(\sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \right) \cdot \phi(\boldsymbol{x}) + b \\ &= \sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \cdot \phi(\boldsymbol{x}) + b \\ &= \sum_{i=1}^{n} \alpha_i K\left(\boldsymbol{x}^i, \boldsymbol{x}\right) + b \end{aligned} \tag{2.20}$$

essentially allowing the calculation of a detector response to a feature vector $\boldsymbol{x}$ without explicitly computing $\phi(\boldsymbol{x})$. A kernel function of the form $K(\boldsymbol{x}^1, \boldsymbol{x}^2) = \boldsymbol{x}^1 \cdot \boldsymbol{x}^2$ is often called a linear kernel and if it is used the original linear SVM formulation (Equation 2.10) can be recovered:

$$h(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K\left(\boldsymbol{x}^i, \boldsymbol{x}\right) + b = \left(\sum_{i=1}^{n} \alpha_i \boldsymbol{x}^i \right) \cdot \boldsymbol{x} + b = \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{2.21}$$

To learn a kernel SVM we minimize the previously defined energy in Equation 2.12 w.r.t. $\boldsymbol{\alpha}$ and b. The regularization term is appropriately redefined to represent the ℓ_2-norm of the weight vector $\boldsymbol{w}$ in the higher dimensional feature space $\mathcal{H}$:

$$\begin{aligned} \|\boldsymbol{w}\|_{\mathcal{H}}^2 &= \left\| \left(\sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \right) \right\|^2 \\ &= \left(\sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \right) \left(\sum_{i=1}^{n} \alpha_i \phi\left(\boldsymbol{x}^i\right) \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \phi\left(\boldsymbol{x}^i\right) \phi\left(\boldsymbol{x}^j\right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right) \\ &= \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{K} \boldsymbol{\alpha} \end{aligned} \tag{2.22}$$

The matrix $\boldsymbol{K}$ with elements $K_{ij} = K(\boldsymbol{x}^i, \boldsymbol{x}^j)$ is called the Gram matrix. It should be noted that kernel SVMs are much slower to evaluate compared to linear SVMs due to the

computation of the dot-product between a feature vector and all training examples with non-zero α-value.

2.5.3 Inhomogeneous Quadratic Kernel

Dalal & Triggs [23] investigated the use of linear and radial basis function (RBF) kernel functions [116, 10] in combination with HOG features. They only found a small improvement over a much faster linear kernel when using a RBF kernel. However, as reported in Sections 2.6 and 2.7, we found that results of both HOG and our proposed method CHOG could be improved by use of an inhomogeneous quadratic kernel:

$$K(\boldsymbol{x}^1, \boldsymbol{x}^2) = (\boldsymbol{x}^1 \cdot \boldsymbol{x}^2 + 1)^2 \tag{2.23}$$

The quadratic kernel is a natural choice in our framework since it can represent multiplicative dependencies between pairs of features [116]. For example a feature vector with only two components $\boldsymbol{x} = \langle x_1, x_2 \rangle$ one (of many) possible lifting functions for the quadratic kernel in Equation 2.23 would be:

$$\phi(\boldsymbol{x}) = \left\langle (x_1)^2, (x_2)^2, x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1 \right\rangle \tag{2.24}$$

The third element of the new feature vector models the dependency between features x_1 and x_2. This is particularly relevant for the CHOG descriptor where gradient sign is meaningful (see Section 2.4.4) and by modeling these dependencies we can model color consistency along object edges in a better manner *c.f.* a linear classifier which can only assign individual weights to elements x_1 and x_2 but not to their multiplicative combination. Hussain & Triggs [62] have also used a quadratic kernel classifier in combination with a dimensionality-reduced feature vector [143] but, in opposition to our findings, report only modest quantitative improvements.

2.5.4 Bootstrapping

Common datasets, such as the PASCAL VOC datasets [32], contain thousands of training images and consequently millions of possible windows. It is not possible to store the features of all of these windows in the training dataset.

For this reason we follow Dalal & Triggs [23] and perform bootstrapping [95] to select

a subset of negative windows salient for the classifier. Bootstrapping applies a learned detector to the images in the training dataset and scans for high ranking false positives. These false positives are then added to the training dataset and the detector is retrained.

The goal of bootstrapping is to collect all negative training windows from the training dataset which lie inside the margin of a hypothetical SVM trained on all windows. Training on all windows and only on the training examples inside the margin will yield the same classifier because of the max-margin definition of the hinge loss (Equation 2.13).

2.6 Empirical Results on INRIAPerson Dataset

In this section we will first discuss implementation details and then present quantitative and qualitative experimental results on the INRIAPerson dataset [23].

2.6.1 Implementation Details

Sliding window detectors have a large number of parameters. This section outlines our choice of parameters and provides further implementation details for the task of pedestrian detection on the INRIAPerson dataset.

Window Descriptor

For HOG cells we follow Dalal & Triggs [23] and use 9 unsigned orientation bins. For CHOG cells, 18 signed orientation bins were used – as noted in Section 2.4.4 the gradient sign carries 'inside/outside' information for CHOG. For both descriptors we use a cell size of 8×8 pixels and a block size of 2×2 cells, with blocks overlapping by one cell (8 pixels) in horizontal and vertical directions, as suggested by Dalal & Triggs. To compute image gradients we follow Dalal & Triggs and use filters $\langle -1, 0, 1 \rangle$ and $\langle -1, 0, 1 \rangle^{\mathsf{T}}$ in x and y direction.

For CHOG the neighborhood around each reference point (Figure 2.7) is a 3×3 square and the means (Equation 2.7) are efficiently computed by convolving the image with a 3×3 box filter prior to computation of the CHOG features.

We note that CHOG descriptors are very high-dimensional feature descriptors: each CHOG block (Figure 2.7) carries 22 real numbers when using the dimensionality reduction technique presented in Section 2.3.2. For all four combinations of reference points (Figure 2.7), HOG+CHOG features therefore consist of $4 \times 22 + 13 = 101$ components

per block. A window of size 128×64 pixels, consisting of 15×7 blocks, yields a feature vector with $15 \times 7 \times 101 = 10,605$ components. HOG+CHOG features are therefore significantly higher dimensional than their HOG counterparts and this should be kept in mind when considering training/test time and the physical requirements of RAM.

Training Protocol

We adopt the protocol used by Dalal & Triggs [23]. We initialize the negative training examples of the dataset with randomly sampled windows of the negative training images, *i.e*. images not containing pedestrian instances. The sliding window scheme uses a scale factor of 1.2 between scale pyramid levels and a stride of 8 pixels between consecutive windows in x and y direction. Windows are of size 128×64 pixels. This set of parameters replicates the scheme proposed by Dalal & Triggs [23].

Throughout this chapter the SVMLight package [64] is used for SVM training. We set the regularization parameter of SVMLight to the default setting.

Bootstrapping Implementation

We perform 3 rounds of bootstrapping on all negative training images. For each scale the highest ranked false positive window is extracted. Windows are considered false positives if their score exceeds -1.

Testing Protocol

Results on per-window classification accuracy are reported using the Detection Error Tradeoff (DET) curve. These curves plot False Positives Per Window (FPPW) against miss rate and have been explained in Section 1.2.2. We follow the test protocol of Dalal & Triggs [23] exactly. The sliding window scheme is identical to the one used at training time (scale factor of 1.2, window stride of 8 pixels).

In addition to the evaluation using DET curves we also present results using precision/recall curves following the PASCAL VOC evaluation methodology [32]. This evaluation methodology counts the highest ranked detection that overlaps at least 50% with a ground-truth bounding box as a true positive and all additional detections as false positives. Further explanation is provided in Section 1.2.1.

Non-maximum Suppression

Since additional detections to the highest ranked one which also overlap by at least 50% count as false positive (see Section 1.2.1), we require a non-maximum suppression (NMS) scheme which eliminates redundant detections. For experiments on the INRIAPerson dataset which yield a precision/recall curve, we adopt the approach of Dalal [22] by treating detections as points in scale-space and then using a mean-shift technique [20], which uses an iterative procedure to converge on modes of the distribution of the detections.

2.6.2 Experimental Evaluation

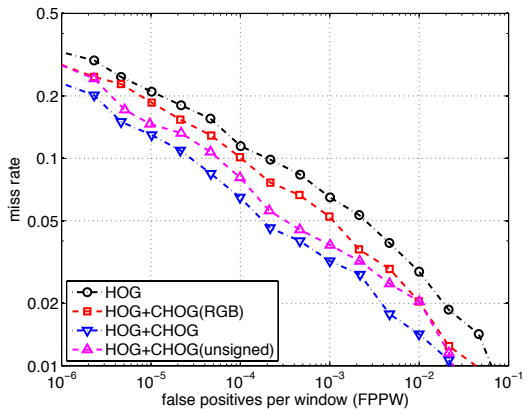
In this section we report results by comparing HOG, CHOG, and published results of other methods on the INRIAPerson dataset. Results are compared using conventional Detection-Error-Tradeoff (DET) curves – see Section 1.2.2 for further details. In addition we also provide results in the form of a visualization of positive SVM weights of the HOG+CHOG detector.

Comparison of Descriptors

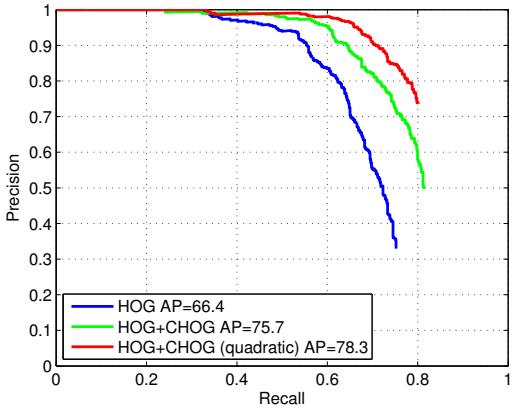
Figure 2.11 (a) shows a comparison of our proposed method (HOG+CHOG) to the original HOG method (HOG) on the INRIAPerson dataset. It is conventional to report results at a false positive per window (FPPW) rate of 10^{-4} [23]. At this FPPW rate our method decreases the miss rate from 0.115 to 0.065, a relative improvement of $\sim 44\%$.

Figure 2.11 (b) shows the corresponding comparison of the two methods using the PASCAL VOC methodology [32]. In this case our method increases the average precision (AP) from 66.4% to 75.7%, a relative improvement of $\sim 14\%$ (HOG+CHOG compared to HOG). The proposed method HOG+CHOG increases both recall and precision, with no loss of precision at lower recall. Note that DET and precision/recall curves are not directly correlated since the latter requires suppression of multiple detections which are counted as false positives [32]. Note that the recall in Figure 2.11 (b) is truncated but dropping the threshold of the detector further does not improve it significantly.

Figure 2.11 (a) also shows the effect of using subsets of components in the proposed method. HOG+CHOG(RGB) shows the combination of the original HOG descriptor with CHOG descriptors using the un-normalized RGB image, giving an improvement of 12% at 10^{-4} FPPW. This shows the effectiveness of the proposed color normalization (Section 2.4.3). As can be seen, the use of the normalized color gradients (HOG+CHOG)



(a) Color Normalization and Gradients



(b) VOC evaluation methodology

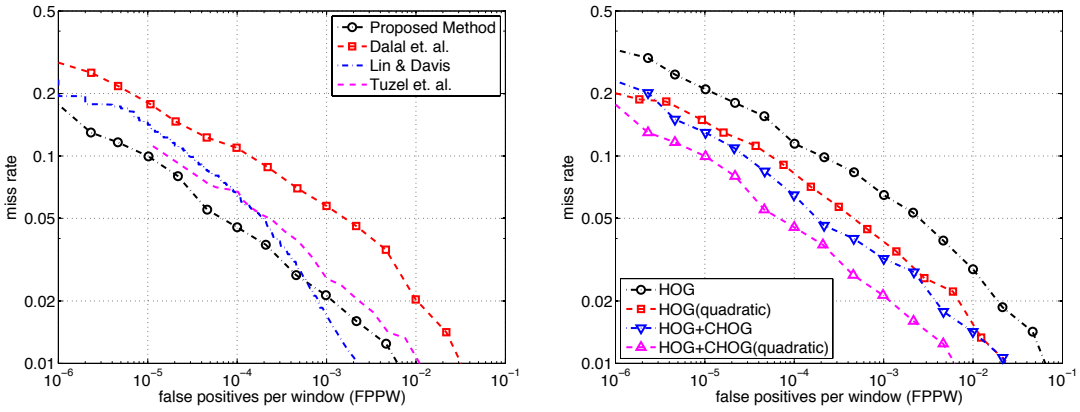
Figure 2.11: (a) Comparison of descriptors. DET curves are shown for HOG, HOG+CHOG without color normalization (RGB), HOG+CHOG with unsigned gradients, and the proposed method HOG+CHOG. Curves nearer to the southwest corner are better. Color normalization and signed gradients significantly improve detection performance. (b) Comparison of HOG and the proposed HOG+CHOG method using the PASCAL VOC [32] methodology. Precision/recall curves are shown for the two methods. Curves nearer the northeast corner are better. CHOG features significantly improve Average Precision (AP) over HOG features alone. SVM classifiers with quadratic kernel further increase AP (HOG+CHOG (quadratic)).

gives a substantial improvement compared to the standard RGB image.

HOG+CHOG(unsigned) shows results of the proposed method using *unsigned* gradients for the CHOG descriptor – this demonstrates that the CHOG descriptor can exploit 'inside/outside' relations by use of signed gradients. As expected, performance is worse than the signed gradients (HOG+CHOG), in contrast to the original HOG descriptor, where comparable results for signed and unsigned gradients have been reported [23]. The use of signed gradients accounts for $\sim 30\%$ of the overall reduction in miss rate at 10^{-4} FPPW (HOG+CHOG compared to HOG).

Comparison of Classifiers

We also compared the performance of the proposed descriptor using both, a linear SVM classifier and an SVM classifier with a kernel. Figure 2.12 (b) shows results for HOG and HOG+CHOG with linear and quadratic kernel. Use of the quadratic kernel improves results for both descriptors – at 10^{-4} FPPW the miss rate decreased from 0.115 to 0.08 for HOG (relative improvement $\sim 31\%$) and from 0.065 to 0.045 for HOG+CHOG – a relative improvement of $\sim 31\%$). On the precision/recall curve using the PASCAL VOC



(a) Comparison to other methods (b) Effect of Kernel

Figure 2.12: (a) Comparison of the proposed detector to other state-of-the-art methods. DET curves are shown for our proposed method, HOG [23], the methods of Lin *et al*. [82] and Tuzel *et al*. [131]. Curves nearer to the southwest corner are better. Our proposed method gives better miss rate for FPPW rates smaller than 10^{-3} compared to all other methods. (b) DET curves are shown for HOG and combined HOG+CHOG descriptors with linear and quadratic kernels ('quadratic'). The use of a quadratic kernel significantly improves performance for both feature descriptors.

methodology [32] (Figure 2.11(b)) using the kernel gives an additional 3.5% improvement in AP. These are substantial improvements, suggesting it may be fruitful to re-consider using kernels with these descriptors if computational expense of the kernel evaluation can be reduced.

Comparison to other Methods

The combination of HOG and CHOG descriptors with a quadratic kernel forms our final method. Figure 2.12 (a) compares the results obtained with this method to the original HOG method and recent work by Lin *et al*. [82] and Tuzel *et al*. [131]. Comparing at the standard FPPW rate of 10^{-4} our method reduces the miss rate by $\sim 30\%$ compared to the next best method [131]. For FPPW rates less than around 10^{-3} our method gives consistently lower miss rate than all other results reported.

Learned Features

Figure 2.13 shows a visualization of the positive SVM weights for a complete set of HOG and CHOG features. For each cell the orientation with maximum weight is shown. Because blocks overlap in our feature descriptor we accumulate the positive weights in

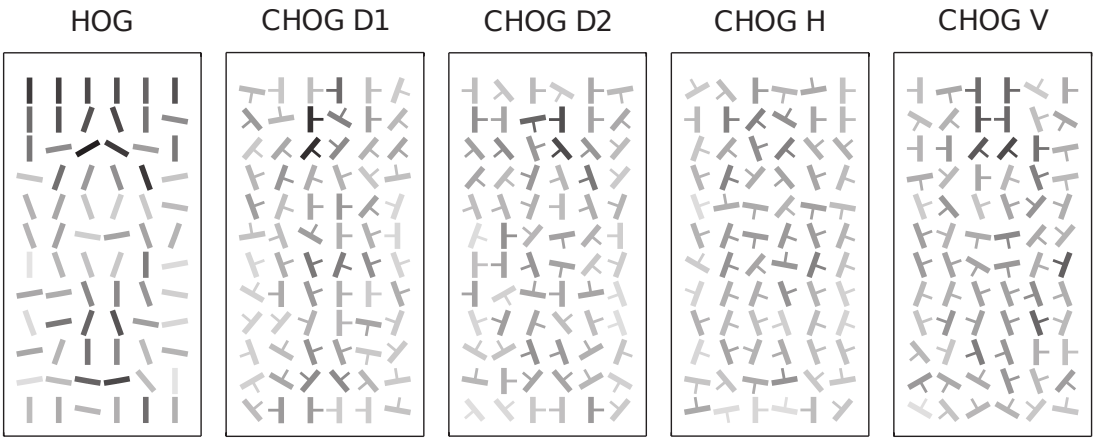


Figure 2.13: Visualization of the positive weights of the final SVM. For each cell the orientation of the bar shown denotes the orientation bin with maximum weight. For CHOG cells, which use signed gradient, orthogonal lines indicate the sign (foreground/background *vs.* background/foreground) of the orientation bin. The labels correspond to the choice of projection reference points (see Figure 2.7). CHOG channels assign high weights to pedestrian parts that emphasize the corresponding block-specific projections, *e.g*. the shoulder regions of the pedestrian for 'CHOG D1' and 'CHOG D2'.

each cell.

It can be seen that the HOG channel models the complete outline of a pedestrian while the individual CHOG channels assign high weight to the parts that are emphasized by the corresponding block-specific projection. For 'CHOG D1' and 'D2' (diagonal reference points – see Figure 2.7) these are the shoulder regions. 'CHOG H' (horizontal reference points) assigns high weights to the side regions of the pedestrian.

Note that the maximally-weighted orientations for the horizontal projections have the same sign – since the reference points for a block are fixed this indicates learning of consistent inside-to-outside gradient. This shows that, in contrast to HOG features, CHOG features are capable of learning coherence in color information between foreground (pedestrian) and background – for further discussion see Figure 2.10.

The SVM also assigns relatively strong weights to vertical edges in the top of the window (Figure 2.13, 'HOG'). Some previous work [109, 85] has proved erroneous in learning artifacts of the INRIAPerson training data due to cropping of positive images and introduction of boundary effects. In addition to strictly following the prescribed test procedure we have validated these results by excluding the topmost blocks. Examination of the positive training data also reveals that these edges are a property of the image context of the pedestrians *e.g*. arising from surrounding pedestrians and buildings rather

than artifacts of the collection procedure used for the INRIAPerson dataset.

2.7 Empirical Results on PASCAL VOC 2010

In this section we present quantitative and qualitative experimental results on the PASCAL VOC 2010 dataset. As before we will discuss details of our implementation first and then mention quantitative and qualitative results.

For training we use the PASCAL VOC 2010 `train` subset while cross-validation on the `val` subset is used to set parameters of the detector. This approach also allows us to report precision/recall curves as well as qualitative results, such as example detections, on the `val` subset. For quantitative evaluation on `test` we retrain the detectors on `trainval`. We use this training/testing protocol for all models trained in this thesis with the exception of the detectors using a kernel in this chapter, which we do not retrain on `trainval`. Retraining and evaluating detectors using an SVM classifier with kernel is very time-consuming: most object classes require several days for training and testing.

2.7.1 Implementation Details

Window Descriptor

The window descriptor is almost identical to the one used for experiments on the INRIAPerson dataset *i.e*. 9 unsigned orientation bins for HOG cells and 18 signed orientations bins for CHOG cells. In addition to using HOG features with 9 unsigned orientation bins as a baseline, we also adopt the approach of Felzenszwalb *et al*. [38] and combine HOG features using 9 unsigned orientation bins with HOG features using 18 signed orientation bins. In combination with the dimensionality reduction technique presented in Section 2.3.2 this yields a 31-dimensional feature vector per block.

Different to previous experiments, a cell size of 6×6 pixels is used. Blocks consist of 2×2 cells and overlap by one cell in horizontal and vertical direction. We chose the slightly smaller cell size over the 8×8 variant of Dalal & Triggs [23] because we have found that it yields slightly better results.

Experiments with a polynomial kernel SVM only use CHOG features D1 and D2 (see Figure 2.7) for efficiency reasons – executing the kernel on a very high-dimensional feature vector is very time consuming and we have found that only using D1 and D2 features does not significantly reduce detection accuracy on PASCAL VOC 2010 when

object classes	height
aeroplane, bird, cat, sofa, train	48 pixels
bike, boat, bus, car, cow, table, dog, horse, mbike, sheep, tv	54 pixels
bottle, chair, person, plant	96 pixels

Table 2.1: Height of the bounding box used for feature extraction for different object classes. Because some object classes are naturally taller than others (*e.g.* person compared to car), they require a taller window.

used in combination with a polynomial kernel.

Window Size

We obtain initial positive training examples by extracting them from training images containing the object class of interest. We only consider positive training examples which are *not* flagged difficult, occluded or truncated by the PASCAL VOC 2010 annotation.

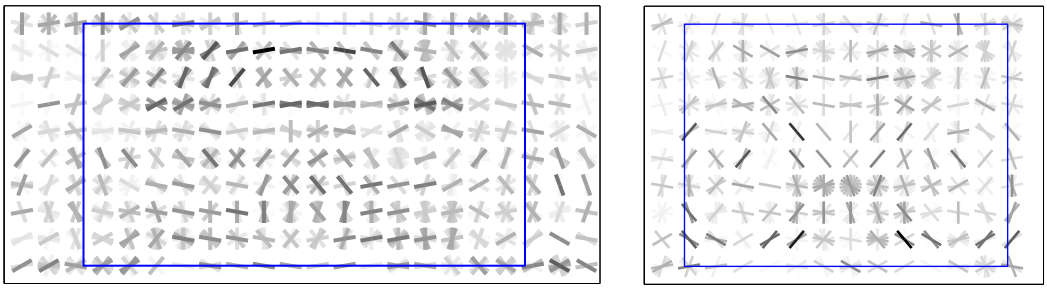
Each training example is normalized to have a fixed height. Because some object classes are naturally taller than others (*e.g.* 'person' compared to 'car') we initialize the height of each object class independently. To do so we arrange the object classes intuitively into three groups and assign heights to these groups as shown in Table 2.1. The first group consists of relatively long objects. The third group consists of objects that are usually tall (*e.g.* a person standing) while the second group consists of all other objects. The width of a window is set equal to the 75th percentile of the width of all bounding boxes. We add a 6 pixel border around each window to capture possible cues in the background.

Predicted Bounding Box

We predict an *'average' bounding box* per detection. This average bounding box is chosen in such a way that it maximizes overlap with all bounding boxes of the positive training instances. Our detector effectively gives the center point of a detection and we predict the average bounding box around this point. The average bounding box for a 'car' and 'bicycle' detector (HOG features only) is shown in Figure 2.14.

Initial Feature Extraction

To extract HOG/CHOG features of the positive training examples we center the window on these examples according to their ground-truth bounding box and extract the feature vector. Every positive training example is left/right flipped to create mirrored 'virtual'



(a) car detector (b) bicycle detector

Figure 2.14: This figure shows car and bicycle detectors by plotting the positive SVM weights for each orientation bin in a HOG block. The blue bounding box represents the average bounding box.

training examples, essentially doubling the amount of available positive training data. Initial negative examples are obtained by extracting 200 windows from the center of training images which do not contain the object class of interest.

Non-maximum Suppression

To create a final set of windows for a test image we adopt the greedy non-maximum-suppression (NMS) scheme of Felzenszwalb *et al*. [38]. This scheme iteratively selects the highest scoring window and removes all windows overlapping by a certain degree. The scheme stops if all windows have either been selected or removed.

Cross-validation on the PASCAL VOC validation dataset (`val`) is used to set the overlap threshold. We consider possible values in the range of 0.2 to 1.0 and select the one with the highest AP. We chose this NMS scheme over the mean-shift-based technique of Dalal [22] because it performs slightly better and is also used by the method of Felzenszwalb *et al*. [38].

It is important to note that the overlap measure of Felzenszwalb *et al*. is not the one used by the PASCAL VOC challenges [32] (see Equation 1.1) but the division of the intersection area of two bounding boxes by the area of the lower ranked bounding box. That is, given two detections with bounding boxes B_l, B_k where the detection of B_l is the selected higher ranked detection, the overlap measure is

$$ov\left(B_l, B_k\right) = \frac{area\left(B_l \cap B_k\right)}{area\left(B_k\right)} \tag{2.25}$$

This overlap measure has the main effect that lower ranked detections which are fully contained within the higher ranked detection are removed as they always overlap 100% given the above measure of overlap.

We also apply non-maximum suppression for each scale separately when bootstrapping, using an overlap parameter of 70% [38]. This has the main effect that not all neighboring false positives are added to the training dataset but only the highest ranked one.

Training Protocol

We use a maximum of 10 rounds of bootstrapping and stop early if fewer than 200 false positives are extracted. Every round we scan 200 randomly selected training images, extracting a maximum of 3 false positives per scale.

Different to the experiments on the INRIAPerson dataset a stride of 6 pixels is used for the sliding window detector in x and y direction, and the resizing factor between scale-levels is 1.2. We chose a smaller stride between consecutive windows because it yields better performing detectors.

Bootstrapping Implementation

During bootstrapping we scan training images for high ranking false positives. We consider windows as false positives if they have a score above a threshold of -1 (the SVM margin) *and* the predicted *average* bounding box does not overlap at all with any ground-truth bounding box of the object of interest.

Testing Protocol

The test protocol matches the training protocol. The stride of the sliding window detector is set to 6 pixels and the resizing factor between scale-levels is 1.2.

Truncated Objects

To compute HOG and CHOG features, we first compute HOG/CHOG cells at every pixel in the image. For HOG features this can be efficiently realized by the use of integral images [148, 138]. Look-up-tables are a fast way to compute gradient orientation and magnitude per pixel. Block normalization is then a simple matter of picking the cells in a pre-defined stride, which in our implementation corresponds to the size of a cell, concatenating them and applying the ℓ_2 normalization to achieve contrast invariance.

To detect objects which are partially visible due to truncation by the image boundary, we pad the HOG/CHOG cell representation prior to block normalization with additional cells at each side of the image by 50% of the size of the window. These additional cells have all bin responses set to 0. This padding is applied to *all* training and test images at each scale separately. During block normalization the entire feature vector of a block is set to 0 if it contains as much as a single cell which is considered not visible. We have found experimentally that padding the image with more than 50% of the window size does not improve detection accuracy further.

We recall that the sliding window detector densely scans an image and once the scan is completed the image is rescaled by a fixed factor. This scheme usually exits once no window fits into the resized image anymore. In our implementation images are padded with 50% of the window size at each scale, causing the sliding window detector not to exit anymore because the actual image size will always be larger than the window size due to the additional padded areas. We therefore stop creating additional scale levels for the sliding window detector once either the width or the height of the original unpadded image is smaller than a third of the width or height of the window.

Image padding is only performed for the PASCAL VOC and not for the INRIAPerson dataset because Dalal & Triggs [23] dictate a strict evaluation protocol which does not allow for padding of the image boundary.

2.7.2 Quantitative Results for PASCAL VOC 2010

Table 2.2 shows Average Precision (AP) for all 20 VOC classes on the `comp3` challenge `test`-set. We compare HOG features and HOG+CHOG features utilizing an SVM with linear and quadratic kernel.

Comparison of Descriptors

Similar to the results on the INRIAPerson dataset, we observe a substantial performance improvement when combining HOG and CHOG features compared to using HOG features alone. For a linear SVM classifier HOG+CHOG gives best AP for *all classes*. Substantial improvements in AP can be observed for various classes, including rigid object classes such as ‘aeroplane’, ‘motorbike’ and ‘train’ as well as more articulated object classes such as ‘cow’, ‘horse’ and ‘sheep’. For example, AP for the ‘motorbike’ class improves from 10.8% to 20.2% – a relative improvement of 87.0%. For the ‘train’ class AP improves from 2.8% when using HOG features to 11.9% when using

AP (on `test`)	aero	bicycle	bird	boat	bottle	bus	car
HOG	16.5	24.2	0.0	0.1	0.1	22.7	14.0
HOG-31	22.2	**25.3**	0.0	0.8	5.0	25.7	17.5
HOG+CHOG	21.9	25.0	0.4	2.2	6.1	24.7	16.2
HOG (quad)	**27.1**	24.5	**2.2**	**4.6**	**9.1**	27.3	18.0
H+C (quad)	24.4	24.3	1.7	4.4	**9.1**	**27.5**	**18.3**

AP (on `test`)	cat	chair	cow	table	dog	horse	person
HOG	0.1	2.0	0.4	1.9	0.1	0.6	6.9
HOG-31	0.5	3.3	2.2	2.7	0.5	8.3	9.7
HOG+CHOG	2.3	4.4	3.4	2.1	2.8	8.8	**15.6**
HOG (quad)	4.1	5.1	4.6	**2.5**	**7.1**	12.8	11.8
H+C (quad)	**4.9**	**5.4**	**5.4**	1.9	7.0	**13.1**	14.7

AP (on `test`)	mbike	plant	sheep	sofa	train	tv	mean
HOG	10.8	2.4	0.3	1.4	2.8	21.3	6.4
HOG-31	15.6	3.4	10.0	2.5	11.1	25.1	9.6
HOG+CHOG	20.2	**4.9**	10.8	2.2	11.9	26.3	10.6
HOG (quad)	20.7	3.3	7.6	2.5	9.6	25.3	11.5
H+C (quad)	**22.8**	4.6	**12.4**	**2.6**	**12.3**	**26.4**	**12.2**

Table 2.2: PASCAL VOC 2010 results (`comp3`, `test`-set). We show Average Precision (AP) for detectors using HOG features (with unsigned gradient), HOG features with signed and unsigned gradient (HOG-31) and a combined feature vector of HOG and CHOG features. In addition, HOG and HOG+CHOG feature descriptors are either combined with a linear SVM or an SVM with inhomogeneous quadratic kernel ('quad'). The best AP per object class is shown in bold font. AP improves for all object class when comparing for HOG+CHOG to HOG.

HOG+CHOG feature descriptors. Overall mean AP increases from 6.4% for HOG to 10.6% for HOG+CHOG – a substantial relative improvement of of 65.6%.

We also compare our results to the previously discussed 31-dimensional HOG feature vector introduced by Felzenszwalb *et al*. [38]. This feature extraction scheme, which we name HOG-31, combines HOG features with unsigned and signed gradients in one feature descriptor. In our experiments we have found that it performs better than HOG features using 9 unsigned gradients (HOG in Table 2.2) with some improvements being substantial *e.g*. the 'motorbike' object class which improves from 10.8% for HOG to 15.6% for HOG-31. Mean AP increases from 6.4% to $9.6\$$ – a relative improvement of 50%.

When compared to HOG+CHOG, HOG-31 tends to perform worse with the exception of the 'aeroplane', 'bicycle', 'bus', 'car', 'diningtable' and 'sofa' object classes. However, for these object classes AP results are usually quite close for the two methods. Mean AP improves from 9.6% for HOG-31 to 10.6% for HOG+CHOG – a relative improvement of 10.4%.

Figure 2.15 shows precision/recall-curves for six selected object classes using the `val`-subset of the PASCAL VOC 2010 for evaluation. We observe significant AP improvements for HOG+CHOG (megenta curves) in comparison to the usage of HOG features alone (HOG, red curves). Precision generally increases for all recall values for the presented classes. For the 'person' class (e) AP improves from 6.3% to 14.9% – a relative improvement of $\sim 137\%$. AP improvements are even stronger for the 'sheep' class (f): 1.2% for HOG vs. 10.3% for HOG+CHOG. The 'car' class (b) also shows a significant relative performance improvement of $\sim 41.7\%$.

In addition we compare HOG+CHOG with the 31-dimensional HOG feature vector which combines signed and unsigned gradients into one descriptor [38] – green curves in Figure 2.15. This feature descriptor can provide a significant boost in AP *e.g*. AP for the 'person' object class increases from 6.3% for HOG to 9.2% for HOG-31 – a relative improvement of 46%. However, as mentioned earlier HOG+CHOG generally performs better than HOG-31. Precision/recall curves for the remaining object classes can be found in Appendix A.1.

These results suggest that detection accuracy (measured by AP) can always be improved by the use of CHOG features in combination with regular HOG features independent of object class.

Comparison of Classifiers

For either feature descriptor, performance can usually be increased further through the use of a quadratic kernel with the SVM classifier. For HOG features alone the nonlinear SVM improves AP for all classes and mean AP improves from 6.4% to 11.5%. When the combined feature vector of HOG and CHOG features is used in combination with the nonlinear SVM, AP improves for 16 of 20 classes and mean AP increases from 10.6% to 12.2%. Note that the detectors using kernel SVMs are not retrained on `trainval`: we expect slightly improved AP values ($\sim +1\%$) in case detectors would be retrained.

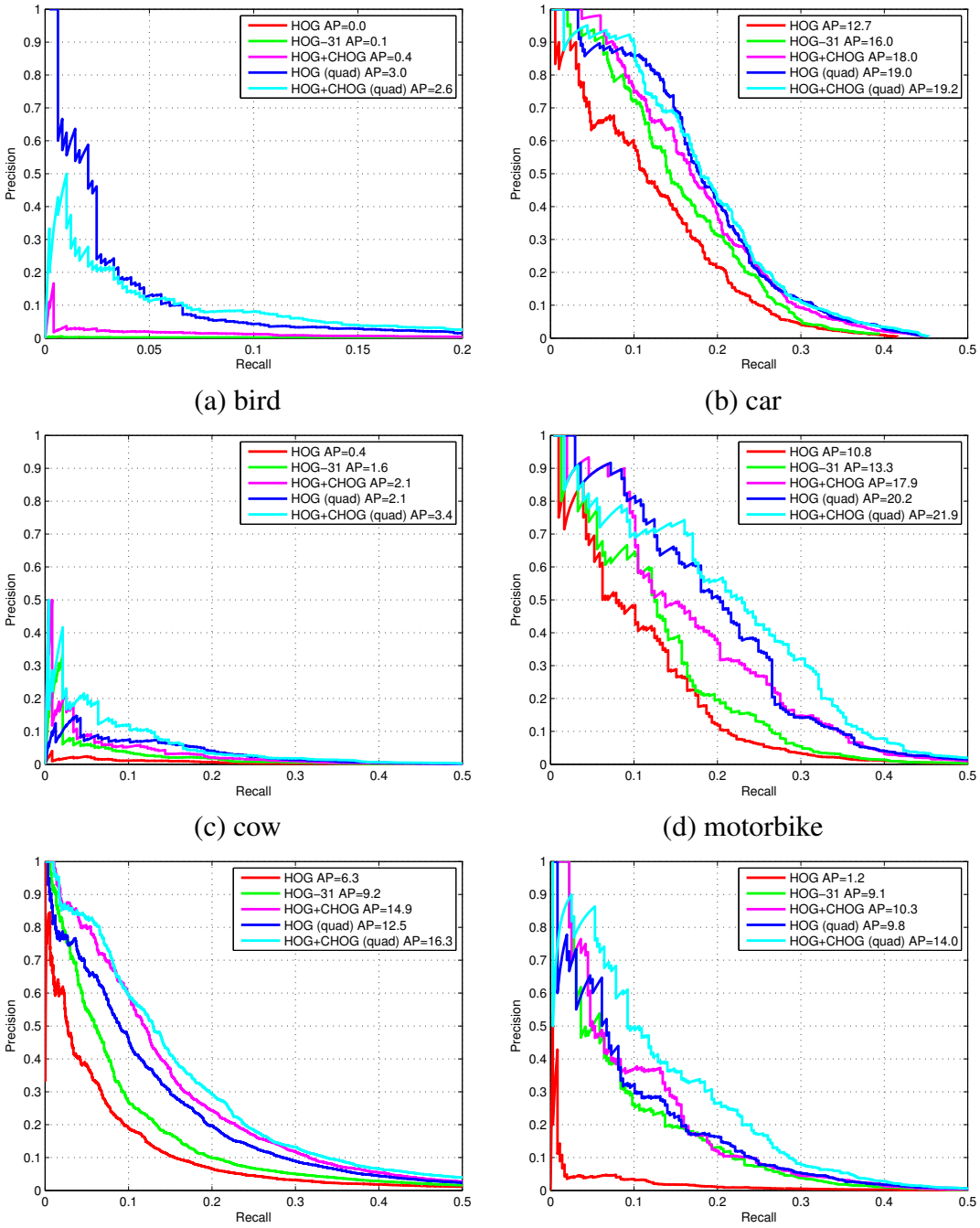
In comparison to all other listed methods HOG and CHOG features in combination with a quadratic kernel (H+C (quad)) gives the best AP for 12 of 20 classes as shown in Table 2.2. Overall the mean AP improves from 6.4% for the Dalal & Triggs [23] baseline (HOG) to 12.2% for our proposed method – a relative improvement of 90.6%. We did not evaluate the HOG-31 feature descriptor with a kernel SVM for efficiency reasons.

In Figure 2.15 it can be seen that a quadratic kernel usually improves performance further for both HOG features (magenta curve) and HOG and CHOG features combined (blue curve). For the 'bird' class (a) the AP improvements are significant when a kernel SVM is used (HOG quad) – 3.2% vs. 0.1% and 3.6% vs. 0.5% for HOG and CHOG features (HOG+CHOG (quad)). A quadratic kernel also gives a strong improvement in AP for the 'person' class (e) – 11.3% vs. 6.2% for HOG features and 14.6% vs. 12.8% for HOG and CHOG features. In general we see strong performance improvements for the quadratic kernel when it is used to model appearance of a highly articulated class, such as 'person', 'bird' or 'dog'.

2.7.3 Qualitative Results for PASCAL VOC 2010

Learned Features

Figures 2.16, 2.17 and 2.18 show positive detector weights for all orientation bins of CHOG and HOG blocks for 'car', 'bicycle' and 'bus' detectors. For the CHOG representation the maximum positive weight per cell is shown, by plotting a bar which denotes the orientation of the bin with the maximum weight. For HOG features we show all positive bin responses. We additionally show the accumulated positive weights of all channels in the bottom right subplot. These plots are created by projecting the (dimensionality-reduced) features of each block to its top-left cell, essentially removing the overlap between blocks.



(a) bird (b) car

(c) cow (d) motorbike

(e) person (f) sheep

Figure 2.15: Precision/recall curves for selected PASCAL VOC 2010 classes on `val`, comparing HOG to HOG+CHOG (linear kernel) and HOG+CHOG using a quadratic kernel. Combining HOG and CHOG features generally improves performance measured in Average Precision (AP), for most classes substantially. Similarly, the combination of an SVM with a quadratic kernel improves AP *c.f.* a linear classifier for HOG and HOG+CHOG features over a linear SVM. Note that we plot a constrained recall range.

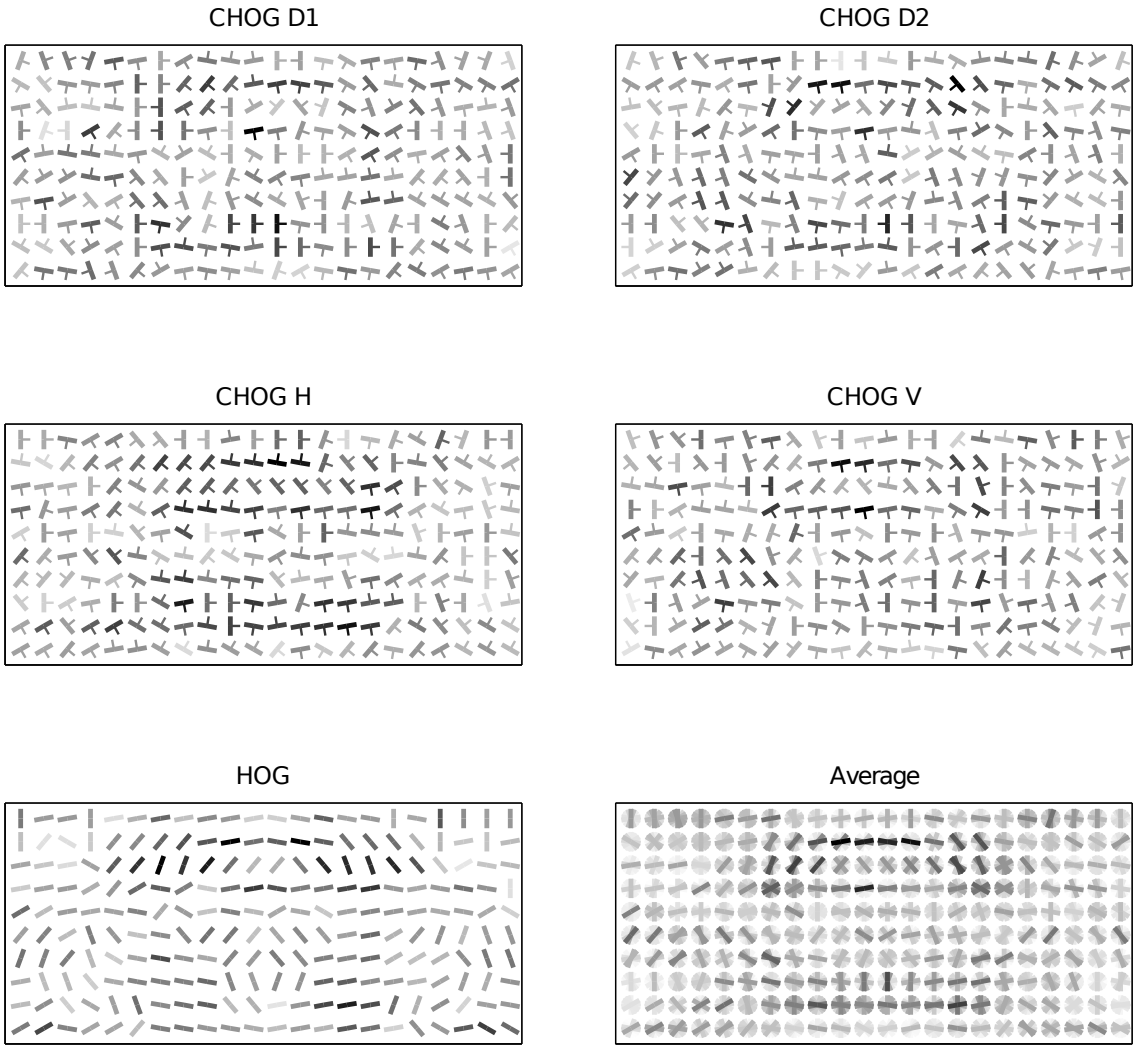


Figure 2.16: Visualization of the positive weights of a car detector. As in previous visualizations (Figure 2.13) the orthogonal lines in the CHOG visualization indicate the sign of the orientation bin. The labels correspond to the choice of projection reference points, see Figure 2.7. We show the positive weights of the HOG descriptors, which use an unsigned gradient in the bottom left subplot, and the positive weights of all channels averaged in the bottom right subplot. Note the consistent gradient sign for CHOG D1, D2, and V in the roof area of the car. This indicates that the detector has learned consistent foreground/background relationships.

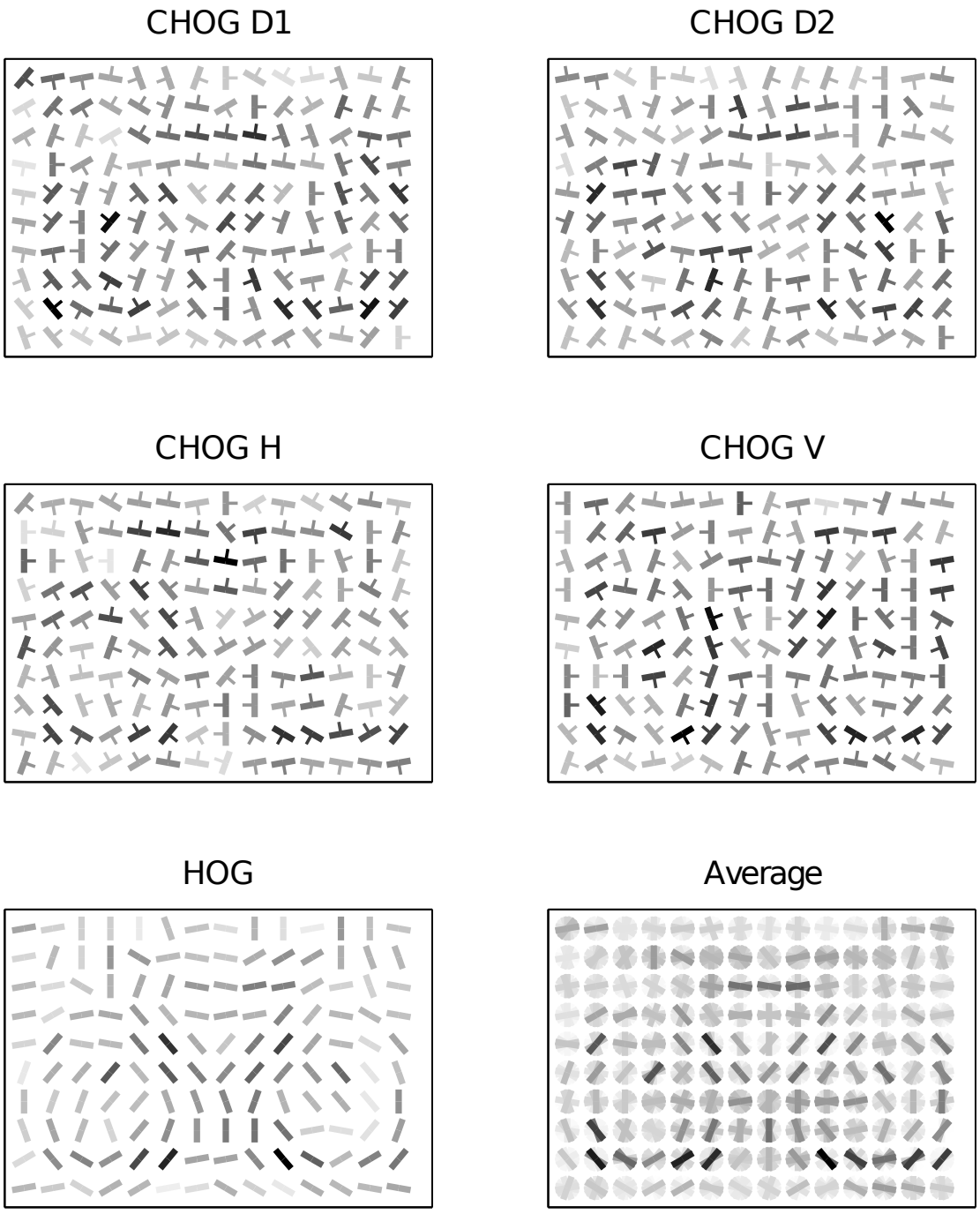


Figure 2.17: Visualization of the positive weights of a bicycle detector in each feature channel. Bicycles usually consists of thin and small parts and hence the local segmentation model will fail. However, due to the color normalization presented in Section 2.4.3 the CHOG channels are still able to represent complementary characteristics of the object *c.f*. the HOG channel.

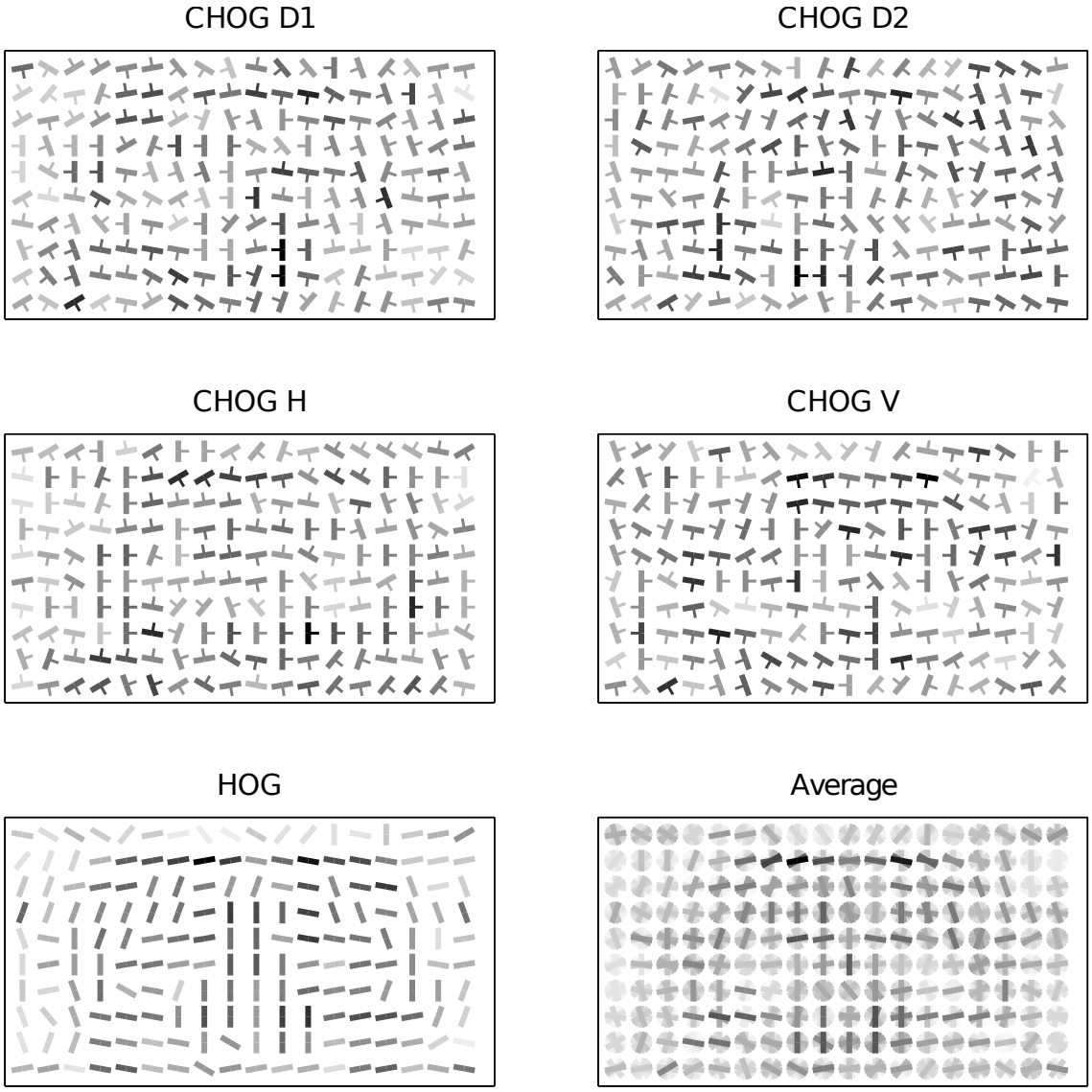


Figure 2.18: Visualization of the positive weights of a bus detector in each feature channel. Similar to the car detector (Figure 2.16) the CHOG channels model consistent foreground/background relationships. Notice for example the representation of the roof of the bus in CHOG V – most gradient directions are consistently pointing towards the second reference point.

Similar to the positive pedestrian detector weights in Figure 2.13, the HOG channel models the complete outline of the object classes while individual CHOG channels concentrate on block-specific projections. Note how for example the gradient orientation of CHOG D1, D2 and V is similar in the roof region of the 'car' and 'bus' detectors. This indicates that most maximally-weighted orientations learn consistent foreground/background relations along the edge of the roof.

Bicycles usually consist of many smaller and thinner parts and consequently our proposed segmentation scheme will not be very successful at locally segmenting a bicycle. In such a case the HOG descriptors usually dominate the final classifier while the CHOG descriptors still contribute due to the proposed color normalization which extracts complementary edges to the HOG feature channel.

Example Detections

Figure 2.19 shows example detections for HOG and HOG+CHOG when used in combination with a linear SVM. The detector utilizing HOG+CHOG feature descriptors yields additional detections (shown as blue bounding boxes) for uniformly colored objects, *e.g.* (c) or (d), or objects which exhibit poor lighting conditions – see *e.g.* (a). In both of these cases segmentation features will be helpful in yielding stronger gradient responses than HOG features alone. HOG+CHOG also provides additional detections for 'visual outliers' *i.e.* object instances which appear very differently to the regular appearance of that object class – see *e.g.* Figure 2.19 (i,h). We hypothesize that the additional modeling capacity of HOG features helps in efficiently representing some outliers.

Figure 2.20 compares detections of detectors using HOG+CHOG features with linear or nonlinear SVM respectively. The nonlinear SVM helps in detecting (strongly) occluded object instances – see *e.g.* Figure 2.20 (c,i,k). Compared to the linear SVM, the nonlinear SVM also provides tighter bounding box predictions (g,k).

2.8 Conclusions

This chapter proposed a novel feature extraction scheme scheme which incorporates 'soft' segmentation cues directly into sliding-window object detection. Because the proposed method computes segmentations in an implicit manner, it does not require a specific classifier architecture and can therefore be combined with different architectures [133, 38, 134]. The computation of the proposed CHOG feature descriptors is



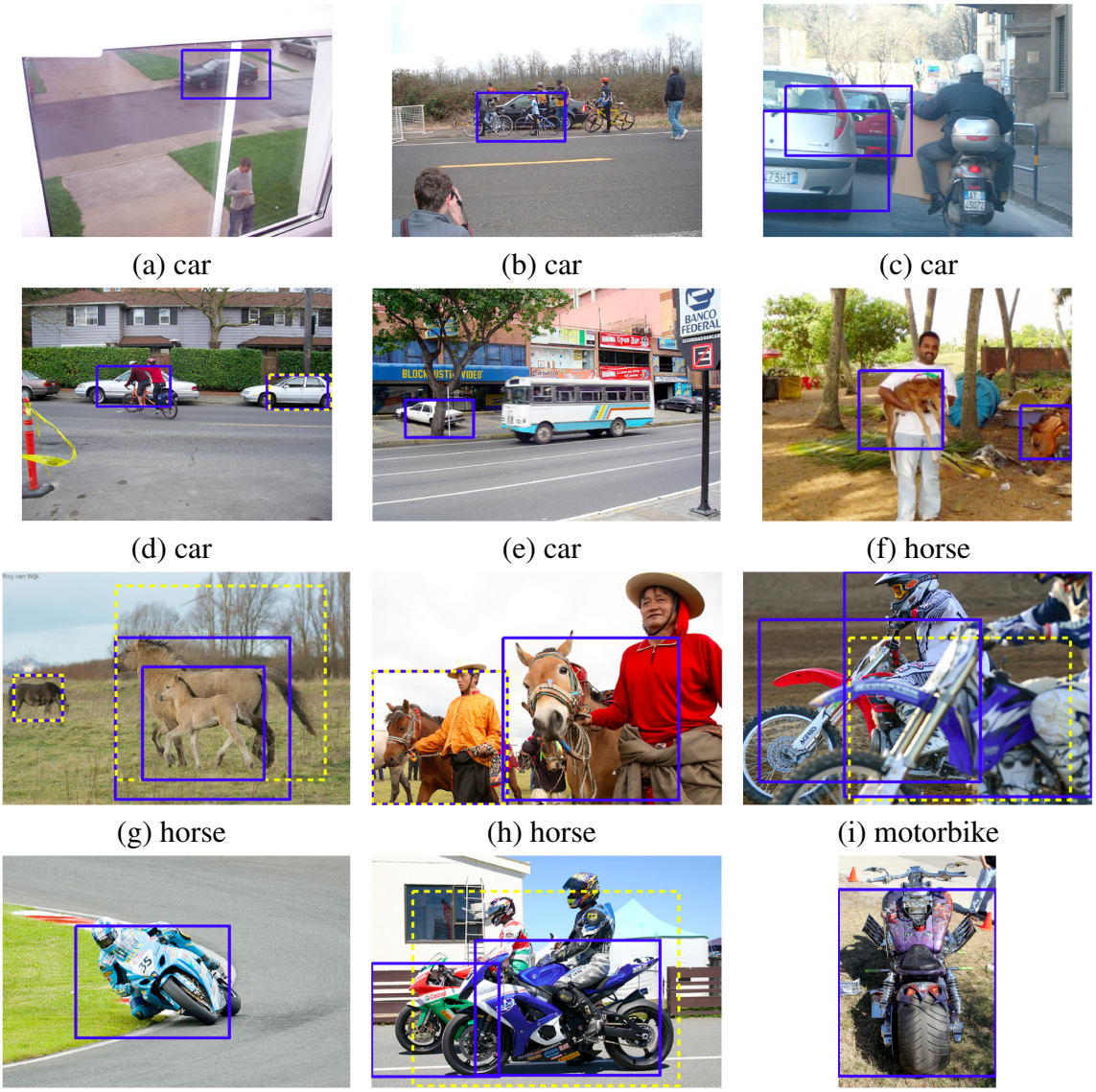
(a) car (b) car (c) car

(d) car (e) car (f) horse

(g) horse (h) horse (i) horse

(j) horse (k) motorbike (l) motorbike

Figure 2.19: Example detections for HOG and HOG+CHOG when used in combination with a linear SVM. Blue bounding boxes indicate detections of the HOG+CHOG classifier while dotted yellow bounding boxes indicate detections of the baseline (HOG). HOG+CHOG gives additional detections when objects exhibit bad lightning conditions (a) or are mainly of uniform color (e, k). In these cases the local segmentation will yield more reliable gradient responses than HOG features alone. HOG+CHOG features are also able to represent object instances in somewhat uncommon configurations, see *e.g.* (i, l) and are more reliable in detecting occluded and truncated object instances – see *e.g.* (b, c, f, h).



(a) car (b) car (c) car

(d) car (e) car (f) horse

(g) horse (h) horse (i) motorbike

(j) motorbike (k) motorbike (l) motorbike

Figure 2.20: Example detections for HOG+CHOG when used in conjunction with a linear SVM (yellow dotted bounding boxes) or SVM with quadratic kernel (blue bounding boxes). The kernel helps to detect 'visual outliers' (f, l) as well as (strongly) occluded objects – see *e.g.* (a, c, k). The detector using the kernel SVM also provides more accurate bounding boxes (g, k).

efficient due to the ability to express the gradients of the soft segmentation as a linear combination of original RGB image gradients (Equation 2.8).

Compared to HOG features CHOG features share the same invariant properties but possess additional modeling capability due to the implicit soft segmentation cues (Section 2.4.4). Our proposed feature vector has the ability to model consistent foreground/background relationships and yields descriptors which have meaningful gradient directions (Figure 2.10). This has the effect that CHOG descriptors are able to represent the existence of consistent colors along both sides of an edge.

In Section 2.5.3 we additionally proposed to combine HOG and HOG+CHOG features with an SVM using an inhomogeneous quadratic kernel. This specific kernel allows for the learning of dependencies between pairs of features and hence enables the learning of relationships between features along edges (Equation 2.23). This property is specifically useful in combination with CHOG features where gradient sign is meaningful and color consistency along an edge can be a useful indicator regarding the presence of an object instance.

The quantitative results show substantial performance improvements for the tasks of pedestrian detection (Section 2.6.2) and object detection (Section 2.7.2) alike for our proposed feature descriptor HOG+CHOG compared to the original HOG features of Dalal & Triggs [23]. For experiments on the PASCAL VOC 2010 dataset [30], mean AP improves from 6.2% for HOG to 10.1% for HOG+CHOG. The use of an SVM classifier with inhomogeneous quadratic kernel further improves detection accuracy: mean AP for HOG+CHOG improves from 10.1% (linear SVM) to 12.2% for the kernel SVM.

Qualitative results (Section 2.7.3) show that detectors using HOG+CHOG descriptors are able to detect further objects. Due to the implicit color segmentation scheme, CHOG features are particularly useful for uniformly colored objects and objects exhibiting poor lighting conditions. We are also able to detect additional occluded object instances.

In summary HOG+CHOG features offer superior modeling capability and detection accuracy compared to regular HOG features while maintaining the relatively low computational complexity of HOG features.

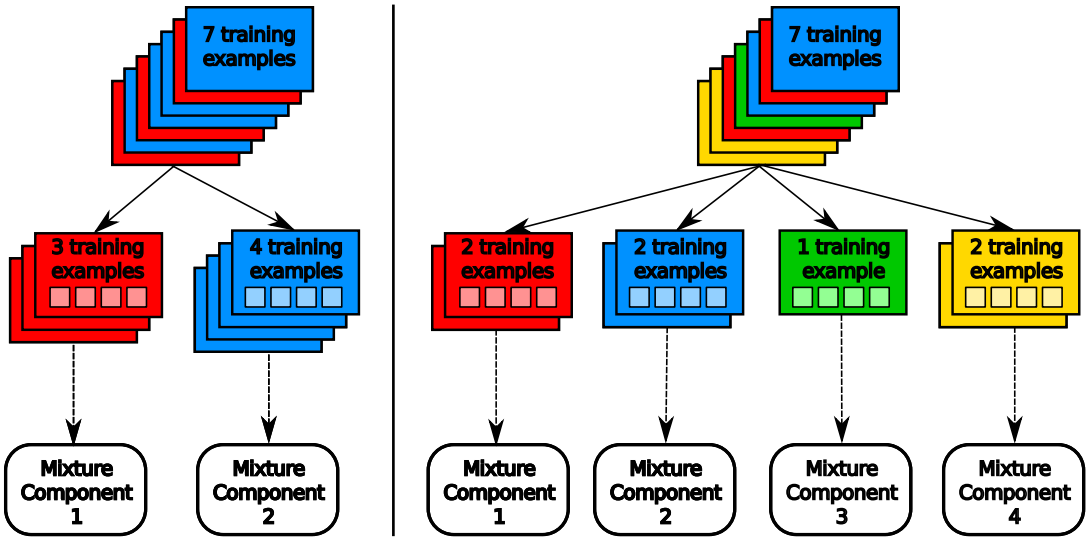
CHAPTER 3

SHARED PARTS FOR DEFORMABLE PART-BASED MODELS

3.1 Introduction

Most state-of-the-art approaches to object detection are based on a sliding-window framework. We have described sliding window object detection in Section 1.3 of this thesis and the work presented in Chapter 2 is an example of a sliding-window detector.

A recent method using this approach, which stands out for its success on the PASCAL VOC challenges [32] is the Deformable Part-based Model (DPM) of Felzenszwalb *et al*. [38]. Most previous sliding-window methods have used a fixed global window descriptor, *e.g*. the Histogram of Oriented Gradients (HOG) descriptor of Dalal & Triggs [23] (see Chapter 2). The DPM method incorporates a higher degree of *learned* invariance by partitioning the object model into a set of local parts which are allowed to move around subject to soft spatial constraints. In addition the DPM also uses a global "root part" which covers the entirety of the object. The root part usually models the coarse appearance of an object class while local parts focus on more detailed characteristics of object appearance.



(a) 2 mixture components (b) 4 mixture components

Figure 3.1: The DPM training "dilemma". We visualize the separation of training data for DPMs. Each mixture component has 4 parts visualized by smaller and brighter rectangles. (a) Training data is divided between 2 mixture components. The entire DPM has 8 parts. (b) Training data is divided between 4 mixture components. Each mixture component has less training data available compared to (a). The entire DPM has 16 parts, twice the number of the DPM in example (a). In practice, the linear growth in number of parts in combination with the linear decrease in amount of available training data per mixture component results in poor generalization.

The use of parts allows the image descriptor to adapt to the underlying image structure, and potentially gives improved generalization by the independence assumptions in part appearance made by the model.

While the notion of parts has a long history in computer vision [45], the use of *discriminative* part learning in the DPM method, in contrast to the predominance of generative approaches in other part-based models [89, 146, 44] has proven particularly effective.

The DPM method uses the idea of a mixture model [63, 10] to capture the large variation in appearance that an object category may exhibit. Different detectors are learned for different 'aspects' or modes of appearance of an object, *e.g.* frontal/lateral viewpoint. In the original DPM formulation [38] each of these 'mixture components' is effectively a separate detector, consisting of a linear classifier applied to a global window descriptor and a set of parts specific to that detector. At training time, each mixture component is essentially trained independently, from a subset of training examples assigned to that component.

While the mixture model approach used by the DPM has been shown to be effective

for a small number of mixture components *e.g*. representing front/side views of an object, attempting to improve accuracy by increasing the number of mixture components is not straightforward for two specific reasons:

1. In a DPM each part has a fixed number of parameters that need to be learned at training time. Since mixture components have separate parts, the number of parameters increases linearly with the number of mixture components. We visualize this behavior in Figure 3.1 by plotting each part as a small, brighter colored rectangle in each mixture component.

2. Training examples are explicitly assigned to a single mixture component only. A linearly increasing number of mixture components therefore results in a linearly *decreasing* amount of available training data for each mixture component. This effect can also be observed in Figure 3.1.

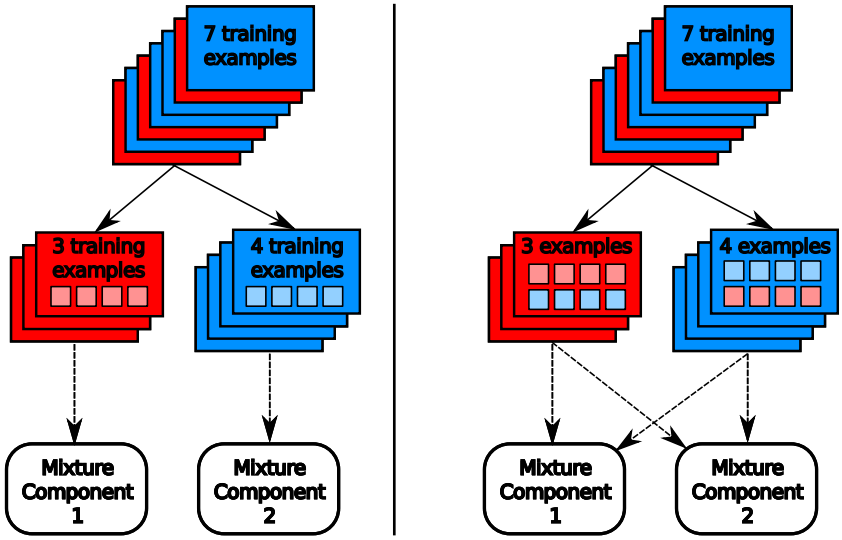
In practice, the increase in number of parameters and coinciding decrease in training data results in poor generalization. In addition, the computational resources needed both at training and test time increase linearly with the number of parts and mixture components.

3.1.1 Chapter Contributions

In this chapter we propose extensions to the DPM which allow more powerful detectors to be built by making efficient use of the available training data. The key idea is to *share* parts between detectors so that (i) the overall number of parameters is reduced, encouraging generalization from finite training data; (ii) training examples are shared across all relevant parameters, such that the paucity of available training data has less negative impact; (iii) computational expense at training and test time is reduced, giving potential for scaling to a large number of classes and mixture components. The first two points are illustrated in Figure 3.2 which compares the original DPM [38] to a DPM with part sharing.

We propose to share parts both (i) *within* object categories, *e.g*. a 'wheel' part may be shared across mixture components representing different viewpoints of a car; (ii) *across* object categories, *e.g*. the same wheel part may be shared by detectors for both cars and motorbikes.

We pose learning of the extended DPM with shared parts as minimization of a novel energy function allowing simultaneous learning of parts for multiple object classes and



(a) Original DPM [38] (b) Proposed DPM with shared parts

Figure 3.2: We compare the effect of part sharing for a DPM with 2 mixture components. (a) The original DPM of Felzenszwalb *et al*. [38] divides the training data between mixture components and learns a separate classifier for each mixture component. In the example each mixture component uses 4 parts. (b) The proposed scheme makes more efficient use of available training data by sharing parts across mixture components. This is visualized by plotting each part (smaller rectangles) in each mixture component. Due to part sharing, training data is implicitly shared across mixture components, which are learned in combination.

mixture components. This additionally results in a 'modular' reformulation of the original DPM [38] which allows for a more compact and intuitive explanation of the original DPM and the proposed model.

Chapter Outline

Section 3.2 reviews previous work on part sharing and DPMs. In Section 3.3 we fully review and reformulate the DPM method of Felzenszwalb *et al*. [38]. Section 3.4 describes our proposed model for sharing parts and Section 3.5 introduces a learning scheme that allows for coupled learning of detectors for multiple object classes. Section 3.6 presents qualitative and quantitative results on the PASCAL VOC 2010 dataset [30]. The chapter closes by presenting conclusions in Section 3.7.

3.2 Related Work

This section presents related work in the area of part-based modeling and part sharing. We will first introduce related work on deformable part-based models and then discuss related work in other areas. Sections 3.2.3, 3.2.4 and 3.2.5 specifically discuss approaches that do not fall into the general sliding window detection scheme which was introduced in Section 1.3.

3.2.1 Deformable Part-based Models

Part-based models have received a great deal of attention in the literature. The DPM method of Felzenszwalb *et al*. [38] simultaneously learns the object detector (a binary SVM classifier) and parts without part-level training annotation, effectively casting the problem of part 'discovery' as a multiple instance SVM learning problem [3]. We review this method in Section 3.3.

Zhu *et al*. [147] reformulate the DPM as a structural SVM learning problem and adopt the Concave-Convex Programming (CCCP) procedure [145] to allow for efficient learning of part appearance and part positions, which (similar to the DPM) are treated as latent variables. CCCP solves the optimization problem that arises in structural SVMs. The main difference to the original DPM [38] is that Zhu *et al*. use parts at multiple levels of feature granularity and organize them in a hierarchical tree-structured model, which defines the dependencies of parts across different granularities. In contrast, the model of

Felzenszwalb *et al*. [38] places all object parts at the same feature granularity while also using a root part, which models the entire object at a coarser feature level.

Vedaldi & Zisserman [135] also propose a structured output model for object detection which implicitly models parts by accounting for alignment of the features representing an object class. Their model also accounts for multiple aspects of an object class as well as truncation by the image boundary. The key differences to the model of Felzenszwalb *et al*. are the use of a structural SVM learner in conjunction with a loss function that takes the overlap of the predicted and ground-truth bounding box into account as well as the ability of explicitly modeling truncation by the image boundary.

All of these methods divide the space of object appearance into a separate mixture of components and define a separate set of object parts for each component. Each mixture component is learned separately, meaning that training data is split between components, as visualized in Figure 3.1. This can lead to an insufficient amount of available training data for each mixture component. In contrast, the method proposed in this chapter suggests sharing parts across mixture components, making more efficient use of available training data.

Very recently, Girshick *et al*. [54] proposed grammar models for object detection, which essentially adopt previous approaches [38, 147] by using DPMs with a tree-hierarchy of parts over multiple feature granularities as well as a structural SVM formalism for learning. In contrast to other presented DPM methods in this section, the method of Girshick *et al*. explicitly shares parts across different mixture components, which correspond to different degrees of occlusion. However, they only perform experiments on the 'person' class of PASCAL VOC 2010 [30] with a hand-crafted model, *i.e*. the way in which parts are shared amongst mixture components is pre-defined by a set of hard-coded rules. In opposition to that, the methods presented in this chapter *learn* how to combine different parts in each mixture component and are not constrained to only working with a single object class.

3.2.2 Sharing Parts in Boosted Classifiers

Previous work on sharing parts across object models has focused on boosted classifiers [111], which compute the confidence of a window with features $\boldsymbol{x}$ by accumulating responses to a set of weak classifiers, *i.e*.

$$h\left(\boldsymbol{x}\right) = \sum_{i} \alpha_i h_i\left(\boldsymbol{x}\right) \tag{3.1}$$

where $h_i(\cdot)$ are the weak classifiers and α_i are classifier-specific weights. In the context of part-based models, the weak classifiers correspond to object parts and can be shared across different object models.

Torralba *et al*. [127] propose a method for selecting weak classifiers simultaneously for a set of object class detectors. The weak classifiers are essentially template matchers using a dictionary of image 'fragments' inspired by Vidal & Ullman [137]. Torralba *et al*. observe that these features, when learned jointly across object classes, correspond to generic edge-like features, whereas training each object class individually gives more object-specific features. The run-time cost of their classifiers scales approximately logarithmically with the number of object classes, and hence sharing of weak classifiers allows for a more efficient framework to be built *c.f*. not sharing them.

Opelt *et al*. [94] use a similar approach to learn a 'visual alphabet' of shape and appearance, in which each entry is a weak classifier that can vote for the center of a possible object instance by using the Generalized Hough Transform [6]. In their framework, shape is represented by small boundary-fragments [93] (a piece of linked edges), which can be learned across multiple object classes. Opelt *et al*. report a sub-linear growth of the visual alphabet when shape features are learned in conjunction across object categories.

Both methods, Torralba *et al*. [127] and Opelt *et al*. [94], provide a significant improvement in the efficiency of the detectors as parts can be reused across object classes. A notable improvement made by the DPM over these methods [127, 94] is the ability to explicitly model different 'aspects' of an object, *e.g*. viewpoints which may not be annotated in the training images.

3.2.3 Conditional Random Field Approaches

In addition to methods adopting a 'conventional' classifier-based approach to object detection, Conditional Random Fields (CRF) [73] have been used to represent object models with a dense arrangement of parts. A CRF is a discriminative undirected graphical model [10], which is globally conditioned on an observation sequence *i.e*. the image. For the task of object detection, nodes in the graphical model often correspond to the part-based evaluation of small image patches and the edges model the dependencies between these patches.

A particularly elegant piece of work in this area is that of Winn *et al*. [142] who propose the 'layout consistent random field' (LayoutCRF). To model the appearance of an object class the LayoutCRF combines local part detectors with a CRF which places intra-

and inter-class constraints on neighboring parts. Intra-class constraints ensure the proper layout of an object instance *e.g*. that the middle of a car should be next to the front, while inter-class constraints acknowledge different possibilities of object-to-object and object-to-background transitions. Inference in such a model is expensive and hence Winn *et al*. [142] only applied the LayoutCRF at a single scale for experimental evaluation.

The 3DLayoutCRF [61] additionally provides a 3D object representation to guide the placement of parts, essentially allowing for multi-view object localization. Inference in this model is still (very) expensive ("1-10 minutes per image" [61]) and requires the limitation to a range of scales as well as the downsampling of the image to a lower resolution. Compared to the original LayoutCRF, the 3DLayoutCRF provides slightly improved results on the UIUC car dataset [2, 1]. The authors also report interesting qualitative results in that their method is able to accurately segment object instances as well as estimate their orientation.

Schnitzspan *et al*. [114] follow a similar approach and model part appearance and location as nodes in a latent CRF, using an EM algorithm to simultaneously discover parts and learn models of their appearance. The model allows for the learning of the long-range dependencies of parts rather than just considering neighbor dependencies (*c.f*. the LayoutCRF [142]) by iteratively removing and adding possible 'candidate' edges to the graph using structure learning [112]. Schnitzspan *et al*. have evaluated their method on the PASCAL VOC 2007 dataset [29] and present competitive results compared to DPMs [38]. However, similar to the LayoutCRFs [142, 61] and different to DPMs, inference in this model is very expensive ("450 sec per image" [114]) and usually requires each window to be filtered by the prior execution of a Dalal & Triggs [23] detector to speed up the detection process.

3.2.4 Implicit Shape Models

Leibe *et al*. [79] propose Implicit Shape Models (ISMs). These models use the Generalized Hough Transform [6] to cast votes for possible object centers in scale-space according to a learned codebook of appearances of an object class. Each codebook entry can be thought of as a part in a part-based model. At test time, each 'activated' part casts multiple votes for the centers of hypothesized object instances according to the learned spatial distribution of the part. A mean shift procedure is then used to derive the final object locations. The ISM uses a segmentation-based hypothesis verification in a second step to improve detection accuracy. This verification procedure iterates on the originally predicted object centers and only uses codebook responses linked to these hypothesized

object instances, potentially discarding misleading evidence from the background

Many variations of the ISM are available. Gall *et al*. [48] replace the codebook representation of Leibe *et al*. with a set of class-specific decision trees (a forest) to allow for discriminative and joint learning of part detectors which are well-localized. Maji & Malik also propose a discriminative extension to the ISM by defining max-margin Hough Transforms [86]. Their framework treats the responses to the codebook as well as the spatial distribution of each entry as a black box, which allows them to learn discriminative weights for the codebook responses. Barinova *et al*. [7] make the point that the mean-shift algorithm used by Leibe *et al*. to derive the final object locations can be potentially harmful [20] to detection accuracy as it requires a significant amount of parameter tuning and often fails if objects are closely located. In contrast, their method reasons about which codebook response 'belongs' to which object instance in the scene. Barinova *et al*. [7] report a significant increase in detection accuracy for their method as well as the ability to detect closely located object instances. The original ISM of Leibe *et al*. [79] uses a robust additive combination of codebook responses to vote for object locations. Lehmann *et al*. [78] note that this additive combination is not justified probabilistically. They overcome this issue by reasoning about the ISM in a sliding window manner and explain the summation of codebook responses as a consequence of their imposed linear model.

Almost all ISM approaches [79, 86, 7] can potentially be easily extended to share parts by learning a joint codebook of the appearance of multiple object classes. Two approaches, which explicitly consider the sharing of parts, are those of Mikolajczyk *et al*. [89] and Thomas *et al*. [124]. Mikolajczyk *et al*. [89] propose a scale and rotation invariant model based on a hierarchical codebook representation of object parts in which appearance clusters are shared among different object classes. According to the authors, the hierarchical layout of the codebook allows for fast matching at test time. Thomas *et al*. [124] learn individual ISMs for multiple viewpoints which lie on a circle around the object. In this framework codebook entries from one viewpoint can be transferred to neighboring viewpoints.

3.2.5 Generative Approaches

Earlier methods have incorporated the idea of part-based models in a generative fashion, by modeling the probability distribution over object appearance using a factorizable composition of object parts, which can also allow for the possibility of sharing parts.

One of the first pieces of work on the generative modeling of object appearance, which

has also motivated other work such as the DPM [36], is the Pictorial Structure Model (PSM) [45, 36]. The key idea in these models is to represent an object as a collection of connected parts. The connections impose dependencies such as relative location and allow the overall model to deform. Inference can be made efficient by using tree-structured models and simple distributions to describe edge dependencies [39].

Fergus *et al*. [41] propose a generative model for object detection based on a constellation of parts. The authors define a joint probability density over shape, appearance, scale and part-level occlusion. At test time the model assigns part labels to affine regions [67] of the image. The constellation of parts is very complex to evaluate and learn because inference requires marginalization over all possible assignments of affine image regions to model parts. Motivated by the PSM [45, 36], Fergus *et al*. [42] propose a solution to this and other shortcomings of the original model by using star models [45] to significantly reduce complexity while training and during detection. Such models organize parts into a star structure, enabling efficient modeling of dependencies between parts, such as relative location.

The method of Fidler & Leonardis [44] also adopts a hierarchical representation of an object class. In this work "parts are composed of parts" [44] *i.e*. iteratively built from smaller subparts defined in the hierarchical representation. At the lowest level edgelet-type features represent the smallest building blocks. The hierarchical representation allows for sharing of parts across object classes and while the lower layers (fine grained descriptors) are highly dense and hence shareable, higher layers are sparser and specific to individual categories. The method has only been evaluated on the relatively simple UIUC image database for car detection [2, 1] for which it provides competitive results.

Sun *et al*. [123] propose a generative model based on a Dirichlet Process Gaussian mixture [11] and define a joint probability density over viewpoint, part appearance (using a codebook representation), part position and part assignments to affine image regions. The model is particular interesting because it allows the share parts across different viewpoints of an object class. Given a viewpoint, an affine transformation is applied to each part to fit with the geometric configuration of a particular object instance *e.g*. wheels of a three-quarter view car usually appear shorter than those of a side-view car. Similar to an ISM [79], the model casts votes for the center of an object instance into scale space for each detected part at test time. Sun *et al*. have evaluated their method on the PASCAL VOC 2006 dataset [33] and report Average Precision values which are half as good as the current state-of-the-art on this dataset [53].

Zhu *et al*. consider part sharing for multi-view and multi-object detection by propos-

ing Recursive Compositional Models [146] based on shape appearance, which allow for hierarchical modeling of parts. However, the models rely on a complex inference scheme and have not been evaluated on the most challenging datasets [32]. As noted by the authors [146], their framework might require discriminative learning to "produce a system with high performance" [146]. This is usually the case for generative models, which have been outperformed by relatively simpler architectures trained using state-of-the-art discriminative methods [32]. We take this into account and build our proposed method on the discriminative DPM which has proven performance, extending this approach by incorporating new mechanisms for part sharing among different viewpoints and object classes.

3.3 Deformable Part-based Model

In this section we review the DPM [38] and reformulate it in order to incorporate the notion of shared parts. The proposed reformulation is 'modular' in the sense that we define the classifier in terms of individual part responses rather than as operating on a feature vector formed by concatenating a subset of features determined by the part positions, *c.f*. Felzenszwalb *et al*. [38]. While we propose a renewed formulation of the DPM, we still use the standard efficient methods for optimization such as the generalized distance transform [39, 38].

Figure 3.3 (top) provides an overview of the detection process in the original DPM.

3.3.1 Feature Descriptor

The DPM essentially adopts a standard sliding window detector scheme, extracting features $\boldsymbol{x}$ for each window of the image. As in the previous chapter and previous work on DPM [38] we use features based on the HOG descriptor [23], in which local histograms of gradient orientation are computed in a set of square 'cells' laid out on a regular 2D grid. Adjacent cells are aggregated and normalized to give 'blocks' with greater invariance to local lighting and spatial deformation. We describe HOG features in greater detail in Chapter 2.

In a DPM two layers of HOG features with different granularity are generally used as shown in Figure 3.3 (a). The first layer models the coarse appearance of the object class and the second layer models the fine appearance. This specific setup allows for a more detailed modeling of object appearance as coarse resolution edges such as boundaries

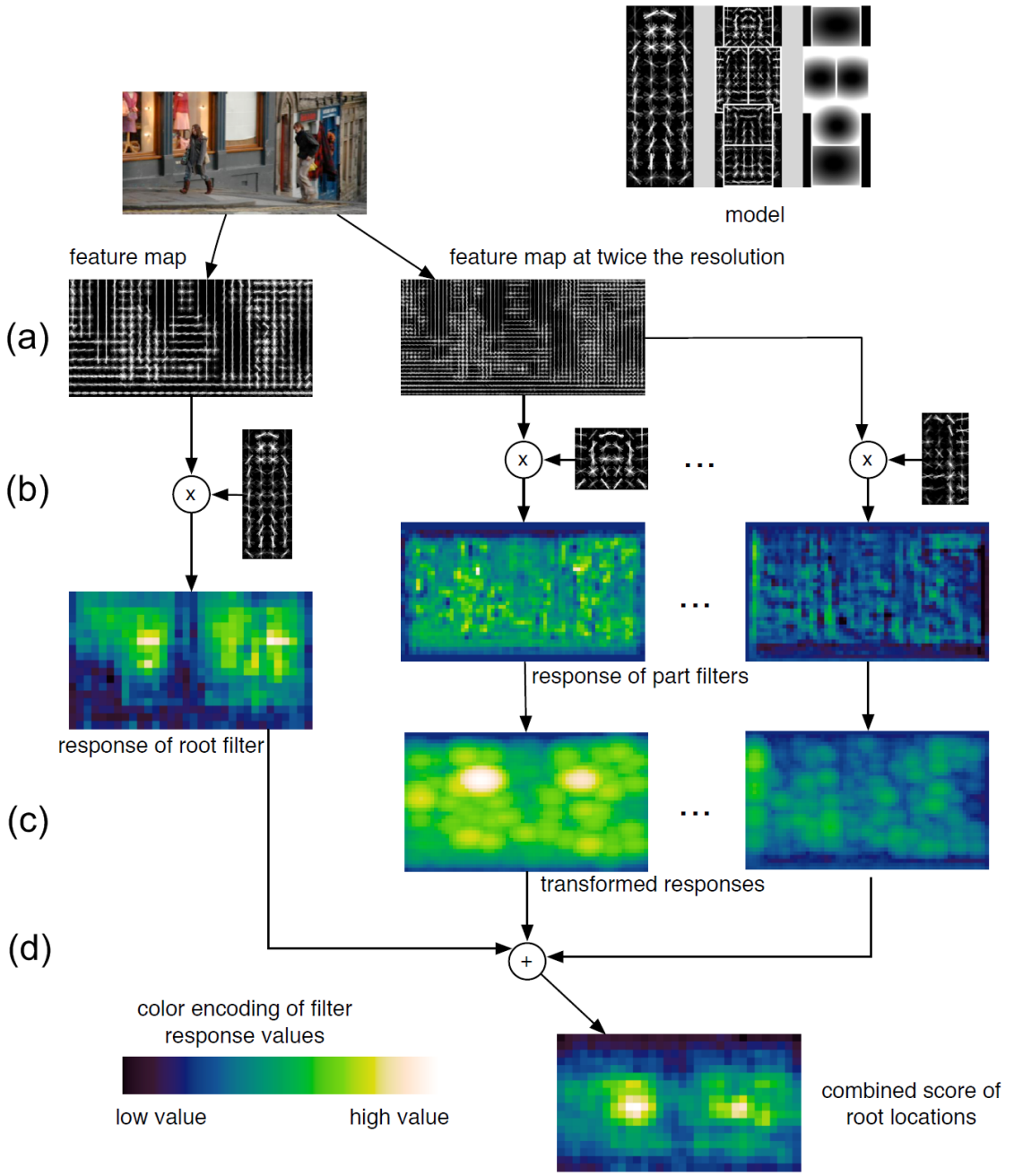


Figure 3.3: Computing detection scores in the Deformable Part-based Model (DPM) of Felzenszwalb *et al*. [38]. The DPM executes separate sliding window detectors for a set of object parts: (b, left) one root part models the rough outline of the object while (b, right) part filters model the detailed appearance of object parts. Root parts generally operate at half the resolution of part filters (a). Each part is allowed to move around subject to some spatial constraints. Given these constrains, a spatial transformation [39] is applied to each part response (c). The transformed part responses are then accumulated and added to the response of the root part (d). This gives the final score of a window. Figure reproduced with permission from Felzenszwalb *et al*. [38].

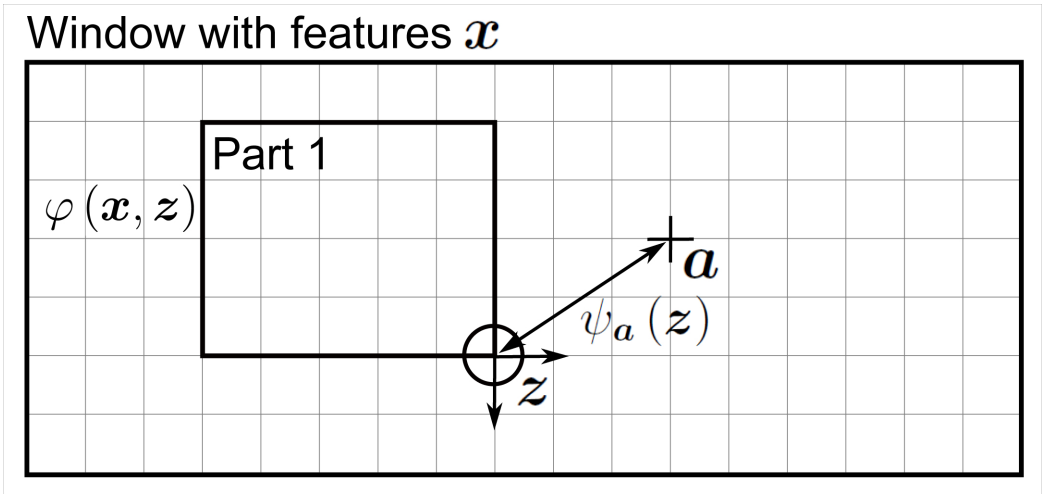


Figure 3.4: Part extraction and spatial configuration. The window is represented by features $\boldsymbol{x}$. $\varphi(\cdot)$ extracts a rectangular patch of features from the $2D$ structure of $\boldsymbol{x}$. The function $\psi_{\boldsymbol{a}}(\boldsymbol{z})$ in combination with the spatial prior (see text) penalizes the position of a part $\boldsymbol{z}$ w.r.t. an anchor position $\boldsymbol{a}$. For visualization purposes we plot the part position $\boldsymbol{z}$ in the lower right corner of the part.

of the object can be captured by the first layer while the second layer can capture more detailed properties of the object class.

3.3.2 Parts

Due to the 2D grid structure of the HOG descriptor, object parts can naturally be defined by extracting a subset of contiguous features from $\boldsymbol{x}$ as shown in Figure 3.4. Let $\boldsymbol{z} = \langle z_x, z_y \rangle^{\mathsf{T}}$ represent the position of a part within a window, and let us further assume parts are rectangular and of fixed size. The features describing the image area the part covers are represented by $\varphi(\boldsymbol{x}, \boldsymbol{z})$, *i.e.* the operator $\varphi(\cdot)$ extracts a rectangular patch of features from the $2D$ structure of $\boldsymbol{x}$ – see Figure 3.4. Note that throughout this section we assume that part positions $\boldsymbol{z}$ are constrained to the extent of the window as this allows for an intuitive explanation of the DPM as well as our extension to the DPM. However, in general and in our implementation of the DPM parts may move to any position in the image.

The position $\boldsymbol{z}$ of a part is not static but inferred at test time, with soft constraints on its position learned at training time. By allowing parts to move, the classifier is essentially re-aligned to the underlying image structure, increasing invariance and improving generalization.

3.3.3 Inference of Part Positions

Each part is linked to an anchor position and incurs a penalty for moving too far away from its anchor. The displacement of a part with respect to an anchor position $\boldsymbol{a} = \langle a_x, a_y \rangle^{\mathsf{T}}$ is represented by

$$\psi_{\boldsymbol{a}}(\boldsymbol{z}) = \left\langle (a_x - z_x)^2, (a_x - z_x), (a_y - z_y)^2, (a_y - z_y) \right\rangle \tag{3.2}$$

The DPM defines a linear cost in terms of this displacement $\boldsymbol{s} \cdot \psi_a(\boldsymbol{z})$, which is a quadratic function of the part's offset from its anchor position. At test time, given features $\boldsymbol{x}$ the best position $\boldsymbol{z}^*$ of a part is chosen to maximize the following objective function:

$$\vartheta(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{a}) = \max_{\boldsymbol{z} \in \mathcal{Q}} \{\boldsymbol{w} \cdot \varphi(\boldsymbol{x}, \boldsymbol{z}) - \boldsymbol{s} \cdot \psi_{\boldsymbol{a}}(\boldsymbol{z})\} \tag{3.3}$$

where $\mathcal{Q}$ is the space of all possible part positions, $\boldsymbol{w}$ is a vector of weights representing the part filters and $\boldsymbol{a}$ is an anchor position. Part filters can be thought of as a linear classifier capturing part appearance *i.e.* a sliding window detector. We call the vector $\boldsymbol{s}$ the *spatial prior* of the part since $-\boldsymbol{s} \cdot \psi_{\boldsymbol{a}}(\boldsymbol{z})$ defines a (un-normalized) Gaussian log likelihood over the part position. This objective moves parts into positions where the part filters score highest, while the displacement cost soft-bounds the score by penalizing part displacement w.r.t. the anchor position $\boldsymbol{a}$. Inference of the part positions is made efficient using the generalized distance transform [39, 38], which is visualized in Figure 3.3 (c). In the remainder we refer to $\vartheta(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}, \boldsymbol{a})$ as a *part response*.

3.3.4 Window Responses

We now briefly outline how part responses defined by Equation 3.3 are combined into window responses which can be ranked and subjected to non-maximum suppression to provide detections. This step is illustrated in Figure 3.3 (d).

Let us assume that there is a set of p parts with part filters $\mathcal{W} = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^p\}$, spatial priors $\mathcal{S} = \{\boldsymbol{s}^1, ..., \boldsymbol{s}^p\}$ and anchors $\mathcal{A} = \{\boldsymbol{a}^1, ..., \boldsymbol{a}^p\}$ as well as a bias term b. To obtain a window response corresponding to confidence that the window contains the object class of interest, the individual part responses are accumulated by summing and adding the bias

term:

$$g\left(\boldsymbol{x}\right) = b + \sum_{i=1}^{p} \vartheta\left(\boldsymbol{x}, \boldsymbol{w}^i, \boldsymbol{s}^i, \boldsymbol{a}^i\right) \tag{3.4}$$

When computing the confidence $g\left(\boldsymbol{x}\right)$, part positions $\mathcal{P} = \{\boldsymbol{z}^1, ..., \boldsymbol{z}^p\}$ are inferred by Equation 3.3. We recall that for the purpose of the explanation of the DPM we have assumed that part positions are constrained to the extent of the window. However, in the DPM parts can move to any position in the image and are merely constrained by their spatial priors. As mentioned earlier, DPMs use a root part which models the entirety of an object. This root part can be interpreted as an individual window in a sliding window detector and offers a reference frame to define anchor positions.

Note that because the part responses are accumulated by summing, the position of each part can be inferred independently [38]. This also leads to a 'modular' formulation of the framework – the final classification confidence is defined as a simple sum of individual part responses. We return later to the question of whether such a simple accumulation scheme is the best choice.

3.3.5 Mixture Model

Recent datasets for object localization, such as the PASCAL VOC datasets [32], include objects with substantial variation in intra-class appearance due to both different types of objects, *e.g.* models of car, and large variations in viewpoint. To cope with this variability we model a class as a mixture model consisting of a set of linear classifiers (mixture components), where each component may be specialized to a particular sub-class or viewpoint, learned at training time. We follow the approach of Felzenszwalb *et al.* [38] and compute the final classification confidence $h(\boldsymbol{x})$ for a window as the maximum over all the component classifiers $g^i(\boldsymbol{x})$:

$$h\left(\boldsymbol{x}\right) = \max_{i=1...d} g^i\left(\boldsymbol{x}\right) \tag{3.5}$$

where d is the number of mixture components. We refer to function $h\left(\cdot\right)$ as the *detector* and to the functions $g^i(\cdot)$ as the *mixture components*. The maximally scoring mixture component $m^* = \arg\max_m \left\{g^m\left(\boldsymbol{x}\right)\right\}$ is called the *mixture assignment* of the example with features $\boldsymbol{x}$.

3.4 Sharing Parts

In our extension to the DPM we propose to share parts among mixture components and also across object classes allowing for parts and training data to be used more effectively. Figure 3.5 (bottom) provides an overview of the proposed framework using part sharing and illustrates the differences compared to the standard DPM implementation, *c.f*. Figure 3.5 (top). In both frameworks part responses are derived from summing responses to linear part filters (solid lines in Figure 3.5) and spatial priors (dotted lines). In our proposed method, all part responses are linearly combined in each mixture component while in the standard DPM parts are strictly assigned to one mixture component.

3.4.1 Motivation

Since notionally, and empirically, the mixture components correspond to different viewpoints of an object, it is natural to have parts which may appear in multiple mixture components, *e.g*. a part which is visible from a range of views. Representing such a shared part explicitly, rather than requiring an identical part be learned in a set of mixture components, enables the learning of stronger models given a finite training set.

Sharing parts also allows us to learn stronger models in the sense that efficient modeling of 'intermediate' visual modes, such as three-quarter views of an object, becomes possible. Consider for example a car detector with frontal and lateral mixture components. Individually both mixture components will respond rather weakly when presented with a car from a three-quarter viewpoint. However, by 'blending' their part-responses we can increase the response to such an example.

3.4.2 Shared Parts

While by definition we assume that the appearance of a part is independent of mixture component *e.g*. viewpoint, it is reasonable to assume that the spatial configuration of parts should vary. Consider *e.g*. a set of mixture components representing a car seen from viewpoints differing in azimuth from 3/4 rear to 3/4 frontal views – as the viewpoint changes the appearance of a wheel changes modestly, but the relative position of the front and rear wheels changes grossly over the range of viewpoints. We therefore propose a model of part sharing in which appearance is shared across mixture components, but anchor positions and spatial priors are unique to each component.

In addition we learn 'combination weights' β for each part in each mixture component

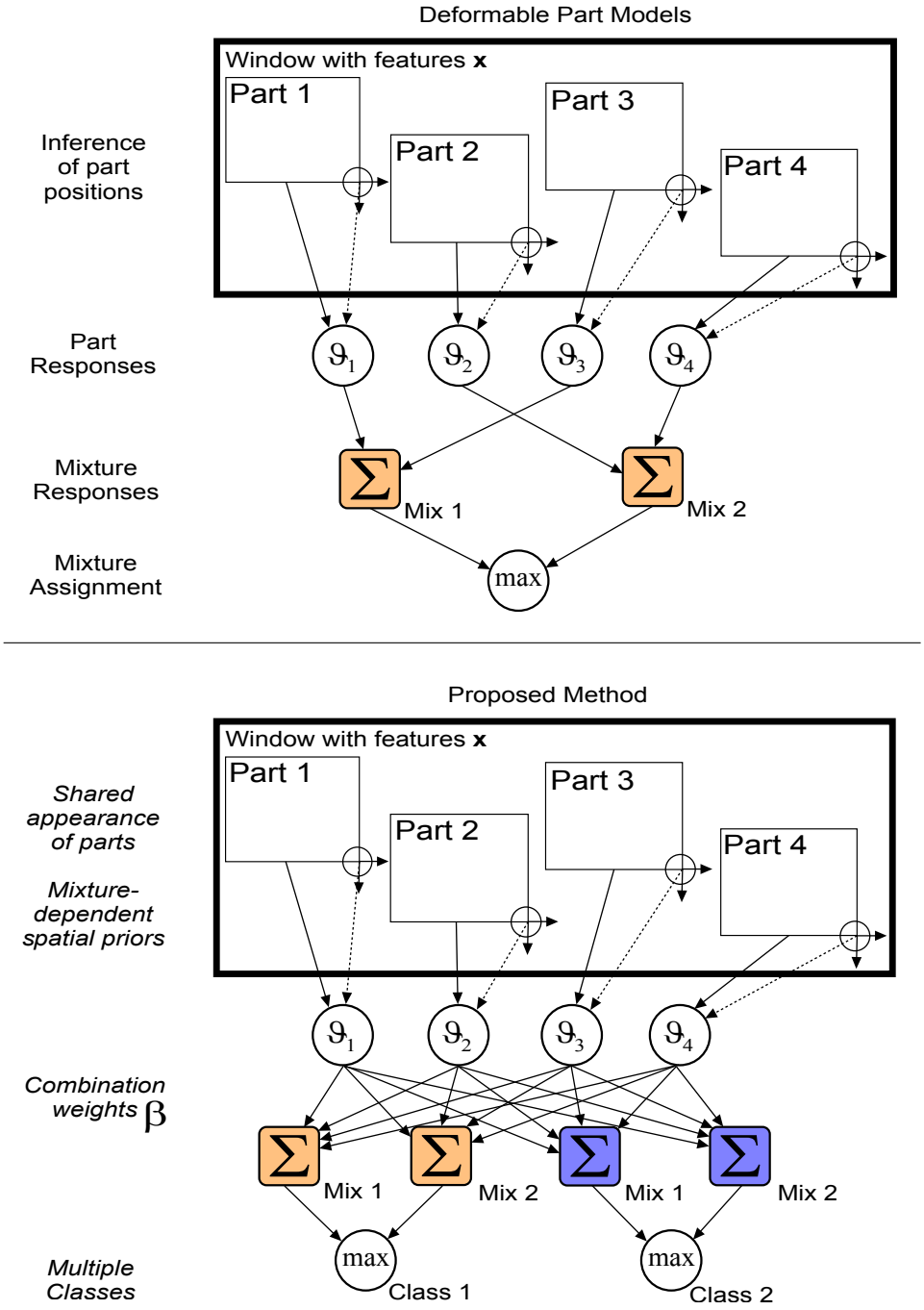


Figure 3.5: Comparison of the detection framework of Felzenszwalb *et al.* [38] and the framework proposed in this chapter. In both frameworks part responses are derived by summing responses to linear part filters (solid lines) and spatial priors (dotted lines). Felzenszwalb *et al.* framework (top): parts are strictly assigned to only one mixture component and object class models are trained independently. Proposed framework (bottom): part responses are linearly combined in each mixture component. Multiple object detectors can be trained simultaneously while sharing part responses among them.

which represent (i) whether we expect to observe the part in a particular mixture component – this is important since some parts might only be visible in particular components *e.g.* in particular viewpoints; (ii) the discriminative ability of a part relative to others in the same component – this is important in 'calibrating' the reliability of different parts.

The response of the lth mixture component for a window with features $\boldsymbol{x}$ is accordingly defined as

$$g^l(\boldsymbol{x}) = b_l + \sum_{i=1}^{p} \beta_i^l \vartheta\left(\boldsymbol{x}, \boldsymbol{w}^i, \boldsymbol{s}^{l,i}, \boldsymbol{a}^{l,i}\right) \tag{3.6}$$

b_l is the bias term assigned to the l'th mixture component. We recall that $\vartheta\left(\boldsymbol{x}, \boldsymbol{w}^i, \boldsymbol{s}^{l,i}, \boldsymbol{a}^{l,i}\right)$ is the part response of the ith part given a part filter $\boldsymbol{w}^i$, spatial prior $\boldsymbol{s}^{l,i}$ and anchor position $\boldsymbol{a}^{l,i}$. Note that the part filters are shared (*i.e.* a single superscript i for all parts) while spatial priors and anchors are not shared (two-component superscript l, i). As a result, spatial priors and anchors are each linked to a specific mixture component.

$\boldsymbol{\beta}$ can be interpreted as a matrix in which the lth row $\boldsymbol{\beta}^l$ represents the combination weights used in the lth mixture component. This is also shown in Figure 3.5 (bottom) in which all parts are connected to all mixture components, individually weighted by $\boldsymbol{\beta}$. In our scheme values of $\boldsymbol{\beta}$ are constrained to be positive, so parts can not explicitly be used to represent 'negative' evidence against the presence of an object class. An individual value of $\boldsymbol{\beta}$ is zero if a part is not used by specific mixture component.

3.5 Learning

In this section we describe how we learn a detector $h(\cdot)$ given the training data provided in the PASCAL VOC 2010 dataset [30]. We cast detector learning in terms of an energy minimization and explain an optimization scheme in Section 3.5.2. The energy function in combination with the optimization procedure ensures guarantees of convergence, which fully motivate the given approach, and are explained in Section 3.5.3.

Notation

Similar to the SVM classifier in Chapter 2 we use a training dataset which consists of feature vectors of image windows: $\mathcal{T} = \{(\boldsymbol{x}^1, y^1), ..., (\boldsymbol{x}^n, y^n)\}$ where $y^k \in \{+1, -1\}$ is the class of the window (object/non-object) and $\boldsymbol{x}^k$ is the feature vector of the kth

training example. Each training example is assigned *latent variables* $\mathcal{L} = \{\mathcal{M}, \mathcal{Z}\}$, where $\mathcal{M} = \{m_1, ..., m_n\}$ represent the latent mixture assignments and $\mathcal{Z} = \{\mathcal{P}^1, ..., \mathcal{P}^n\}$ the part positions of each example in $\mathcal{T}$.

Recall that $\{\mathcal{W}, \mathcal{S}\}$ define the appearance and spatial configuration of the parts while $\boldsymbol{\beta}$ defines how those parts are linearly combined in the mixture components. We call these variables the *model variables* $\mathcal{O} = \{\mathcal{W}, \mathcal{S}, \boldsymbol{\beta}\}$.

In the following sections we will first explain how to learn these variable for a single object class, and then provide an extension to model multiple object classes simultaneously while sharing parts among them.

3.5.1 Energy Function

The energy function is dependent on the model variables $\mathcal{O} = \{\mathcal{W}, \mathcal{S}, \boldsymbol{\beta}\}$ and assumes a training dataset $\mathcal{T}$ as given. To represent this we write the energy function as $E(\mathcal{O}|\,\mathcal{T})$. Note that going forward we discard the bias term from the presentation of the optimization scheme for the sake of notational clarity.

The energy function consists of two parts: (i) a regularization term, ensuring good generalization on unseen data and (ii) a loss term, determining how well the detector predicts the training data:

$$E(\mathcal{O}|\,\mathcal{T}) = \frac{\lambda}{2} R(\mathcal{W}, \boldsymbol{\beta}) + \sum_{k=1}^{n} L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) \tag{3.7}$$

where λ sets the relative importance of the regularization term compared to the loss term. This loss term accumulates prediction errors in form of a loss function $L(\cdot)$ which determines how deviations of a prediction $h\left(\boldsymbol{x}^k\right)$ from the target values y^k should be penalized. As in the previous chapter, the hinge loss is used:

$$L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) = \max\left\{0, 1 - y^k h\left(\boldsymbol{x}^k\right)\right\} \tag{3.8}$$

The hinge loss is considered a robust loss function due to its max-margin definition: only training examples that lie inside the margin (*i.e.* the hinge loss is not zero) contribute to the overall energy.

Note that the proposed energy function allows for simultaneous learning of multiple mixtures as $h(\cdot)$ always picks the mixture component that satisfies the inference scheme

of Equation 3.5. Furthermore the evaluation of $h(\cdot)$ implicitly performs inference on mixture assignment and part positions for each training example in $\mathcal{T}$.

Multi-class Extension

At this point the energy $E(\cdot)$ is constrained to a single object class. We can extend our formulation to multi-class learning by reformulating the loss term so that it accumulates the losses of multiple detectors:

$$E(\mathcal{O}|\mathcal{T}) = \frac{\lambda}{2} R(\mathcal{W}, \boldsymbol{\beta}) + \sum_{s=1}^{r} \sum_{k=1}^{n_s} L\left(y^{s,k}, h^s\left(\boldsymbol{x}^{s,k}\right)\right) \tag{3.9}$$

where r is the number of object classes we are considering for training. Each object class has its own training dataset $\mathcal{T}^s = \{(\boldsymbol{x}^{s,1}, y^{s,1}), ..., (\boldsymbol{x}^{s,n_s}, y^{s,n_s})\}$ and detector $h^s(\cdot)$, but note that these detectors may share part responses according to the learned combination weights $\boldsymbol{\beta}$.

Regularization

The regularization term $R(\cdot)$ encourages good generalization by controlling over-fitting of the learned detector on the training data. Previous approaches such as Felzenszwalb *et al.* [38], which do not share parts, implement regularization by penalizing the squared ℓ_2-norm of the individual part filters as well as spatial priors for each mixture component:

$$R(\mathcal{W}) = \sum_{l=1}^{d} \sum_{i=1}^{p} \left\|\boldsymbol{w}^{l,i}\right\|^2 + \left\|\boldsymbol{s}^{l,i}\right\|^2 \tag{3.10}$$

In the case of part sharing penalizing the ℓ_2-norm of the part filters and spatial priors alone is not sufficient since $\boldsymbol{\beta}$ can still grow towards unreasonably high values and consequently the window responses (Equation 3.6) can be large, leading to detectors which over-fit on the training data $\mathcal{T}$.

As mentioned earlier, we constrain all values of $\boldsymbol{\beta}$ to be positive. This allows us to

rewrite an individual part response (including the value of $\boldsymbol{\beta}$ for that part) as

$$\begin{aligned}
&\beta\vartheta\left(\boldsymbol{x},\boldsymbol{w},\boldsymbol{s},\boldsymbol{a}\right) \\
&\quad= \beta\max_{\boldsymbol{z}\in\mathcal{Q}}\left\{\boldsymbol{w}\cdot\varphi\left(\boldsymbol{x},\boldsymbol{z}\right)-\boldsymbol{s}\cdot\psi_{\boldsymbol{a}}\left(\boldsymbol{z}\right)\right\} \\
&\quad= \max_{\boldsymbol{z}\in\mathcal{Q}}\left\{\left(\beta\boldsymbol{w}\right)\cdot\varphi\left(\boldsymbol{x},\boldsymbol{z}\right)-\left(\beta\boldsymbol{s}\right)\cdot\psi_{\boldsymbol{a}}\left(\boldsymbol{z}\right)\right\}
\end{aligned} \tag{3.11}$$

We recognize the distributivity of the individual combination weight in the above equation. Regularization hence requires the penalization of the combined entity $\beta\boldsymbol{w}$ and therefore leads to the following regularization term for models with part sharing:

$$R\left(\mathcal{W},\boldsymbol{\beta}\right)=\sum_{l=1}^{d}\sum_{i=1}^{p}\left\|\beta_i^l\boldsymbol{w}^i\right\|^2+\left\|\beta_i^l\boldsymbol{s}^{l,i}\right\|^2 \tag{3.12}$$

This regularization term effectively encourages small weights in the detector which is defined by the weighted combination of parts (according to combination weights $\boldsymbol{\beta}$), such that the overall detector response satisfies the desire to have a smooth response in feature space. For the case of multi-class learning the regularization term is simply extended by summation over all object detectors. Note that in the case that parts are not shared, *i.e*. each row of $\boldsymbol{\beta}$ has only one non-zero value, the regularization term (Equation 3.12) simplifies to the standard regularization term (Equation 3.10) of a DPM [38].

3.5.2 Optimization

We would like to minimize the energy function (Equation 3.7), *i.e*. reducing the accumulated losses of the second term while also ensuring a low regularization term:

$$\min_{\mathcal{O}}E\left(\mathcal{O}|\,\mathcal{T}\right)=\min_{\mathcal{W},\mathcal{S},\boldsymbol{\beta}}E\left(\mathcal{W},\mathcal{S},\boldsymbol{\beta}|\,\mathcal{T}\right) \tag{3.13}$$

This optimization problem consists of determining the model variables $\mathcal{O}=\{\mathcal{W},\mathcal{S},\beta\}$ while simultaneously inferring latent variables $\mathcal{L}=\{\mathcal{M},\mathcal{Z}\}$ for all training examples in $\mathcal{T}$. We provide a visual outline for the optimization of $E\left(\cdot\right)$ in Figure 3.6. The following sections explain the optimization procedure in detail.

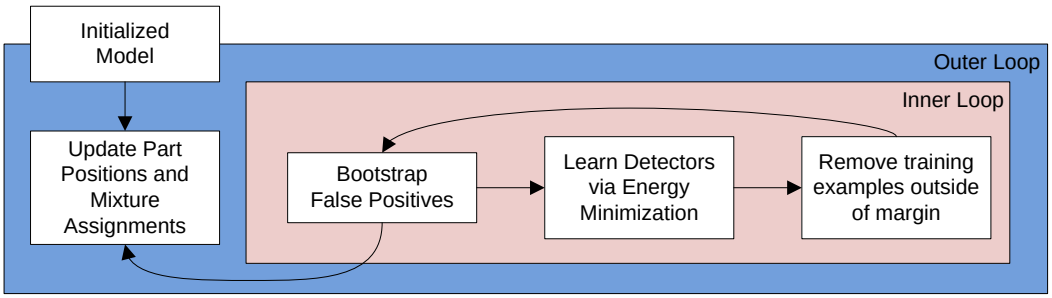


Figure 3.6: Optimization scheme for deformable part-based models. Optimization alternates between executing an "inner loop" and an "outer loop". The outer loop updates latent variables on the positive training examples. The inner loop performs three steps: (i) bootstrapping to iteratively 'mine' all negative training windows that lie inside the margin; (ii) the optimization of the energy $E(\cdot)$ w.r.t. to model variables defined by the SVM classifier and (iii) the removal of training examples that lie outside the margin after optimization of $E(\cdot)$.

Motivation

The motivation for the proposed optimization scheme considers only the case when combination weights $\boldsymbol{\beta}$ are fixed, *e.g.* when parts are not shared. We observe that for fixed combination weights $\boldsymbol{\beta}$, the optimization problem takes a continuous and convex form when provided with mixture assignments $\mathcal{M}$ as well as part positions $\mathcal{Z}$ for each training example in $\mathcal{T}$, because the hinge loss is continuous and convex. In fact, to arrive at a convex function it is only required that the latent variables of the *positive* training examples are fixed. This has also been noted by Felzenszwalb *et al.* [38], who provide a similar motivation for their optimization scheme.

To represent that the latent variables $\mathcal{L} = \{\mathcal{M}, \mathcal{Z}\}$ are fixed we extend the notation of the energy function to $E(\mathcal{O}|\mathcal{T}, \mathcal{L})$. A fixed set of latent variables $\mathcal{L}$ provides an upper bound on the energy function $E(\mathcal{O}|\mathcal{T})$ w.r.t. model variables $\mathcal{O}$:

$$E(\mathcal{O}|\mathcal{T}, \mathcal{L}) \geq E(\mathcal{O}|\mathcal{T}) \tag{3.14}$$

This upper bound essentially means that if the latent variables $\mathcal{L}$ are fixed w.r.t. to an initial set of model variables $\mathcal{O}^{init}$ then any step away from these parameters can only increase the energy or leave it unchanged, compared to making the step and re-inferring $\mathcal{L}$. Given that observation, we propose an optimization scheme which alternates between optimizing $E(\mathcal{O}|\mathcal{T}, \mathcal{L})$ w.r.t. model variables $\mathcal{O}$ in an 'inner loop' and updating the latent

variables $\mathcal{L}$ in an 'outer loop'.

Inner Loop Optimization

Once a fixed set of latent variables $\mathcal{L}$ has been provided, the energy minimization in the inner loop can be written as:

$$\min_{\mathcal{O}} E\left(\mathcal{O}|\,\mathcal{T},\mathcal{L}\right) = \min_{\mathcal{W},\mathcal{S},\boldsymbol{\beta}} E\left(\mathcal{W},\mathcal{S},\boldsymbol{\beta}|\,\mathcal{T},\mathcal{L}\right) \tag{3.15}$$

This optimization problem is still non-convex if the combination weights $\boldsymbol{\beta}$ are not fixed. To resolve this problem we propose to alternate between minimizing $E\left(\cdot\right)$ while holding $\boldsymbol{\beta}$ fixed, *i.e.*

$$(\mathcal{W}^*,\mathcal{S}^*) = \min_{\mathcal{W},\mathcal{S}} E\left(\mathcal{W},\mathcal{S},\boldsymbol{\beta}^*|\,\mathcal{T},\mathcal{L}\right) \tag{3.16}$$

and minimizing w.r.t. $\boldsymbol{\beta}$ while holding $\{\mathcal{W},\mathcal{S}\}$ fixed, *i.e.*

$$\boldsymbol{\beta}^* = \min_{\boldsymbol{\beta}} E\left(\mathcal{W}^*,\mathcal{S}^*,\boldsymbol{\beta}|\,\mathcal{T},\mathcal{L}\right) \tag{3.17}$$

To optimize each of these subproblems we use L-BFGS (Byrd *et al*. [16]) for a fixed number of iterations – details are given in Section 3.6.1. Splitting the optimization problem (Equation 3.15) in such a way also allows us to efficiently cache part responses once $\{\mathcal{W},\mathcal{S}\}$ are fixed and consequently compute Equation 3.17 very quickly as it corresponds to a SVM with only a few variables (part responses and bias terms). To optimize each of these problems, subgradients of the energy (Equation 3.7) are computed and passed to the L-BFGS optimizer – see Section 2.5.1 for further details.

Bootstrapping

It is infeasible to train detectors on the entire training dataset provided by the PASCAL VOC 2010 challenge because the number of negative training examples is very large ($> 10,000$ per image). Instead we iteratively build the negative training dataset given a fixed set of latent variables. This is motivated by the fact that it is only required to train detectors on all training examples which lie inside the margin defined by the hinge loss (Equation 3.8). Such examples have a non-zero hinge loss and hence contribute to the

loss term of the energy function (Equation 3.7). All other training examples have a loss of 0 and do not contribute to the energy. It is therefore not required to store them in the training dataset $\mathcal{T}$ because the optimization of the energy function will arrive at the same solution with or without these training examples.

We perform bootstrapping [95, 23] by extracting false positives with a non-zero hinge loss ($h\left(\boldsymbol{x}^k\right) > -1$) from windows not containing the target object class. We add those false positives to the original training dataset $\mathcal{T}$ and re-optimize $E\left(\mathcal{O}|\,\mathcal{T}, \mathcal{L}\right)$ w.r.t. $\mathcal{O}$. Bootstrapping leads to more robust detectors and we repeat it for a fixed number of iterations or until no more than a fixed number of false positives has been extracted.

Caching Scheme

In addition to bootstrapping we employ a caching scheme following the approach of Felzenszwalb *et al*. [38] in order to remove 'easy' negative training examples from the training dataset. After adding high-ranking false positives to the training dataset and re-optimizing the energy function during bootstrapping, we remove all negative training examples which then lie outside the margin defined by the hinge loss (Equation 3.8). This caching scheme allows us to maintain a relatively small training dataset, which ensures fast optimization speed. Especially in early rounds of training, many potential negative training examples are added to the training dataset which can be removed later because they do not lie inside the margin anymore.

We implement a more conservative form of the above described caching scheme, which only removes negative training examples that persistently lie outside the margin [38]. An example is removed from the training dataset if it lies outside of the margin for at least four inner loop iterations. This is helpful for training examples close to the margin, which may lie inside of the margin after one round of optimization of the energy and outside the margin after another round of optimization. In addition we limit the number of negative training examples to $25,000$ to ensure that the training dataset fits into RAM. If after caching a larger number of negative training examples remain, we retain the $25,000$ highest scoring examples in each mixture component. Caching is only applied to negative training examples but not to positive ones.

As mentioned earlier this bootstrapping and caching scheme serves the purpose of 'mining' a training dataset of examples which lie inside the margin defined by the hinge loss. It can be shown that the proposed scheme converges towards this final dataset. The proof is provided by Felzenszwalb *et al*. [38] and is based on the observation that there is only a finite number of subsets of the final training dataset.

Latent Updates

Once model variables $\mathcal{O}$ have been obtained by execution of the inner loop, we update the latent variables $\mathcal{L}$ of the *positive* training examples in the outer loop. To perform updates on the mixture assignments $\mathcal{M}$ and part positions $\mathcal{Z}$ the learned detector is evaluated on the training images containing the object class of interest. The part positions and mixture assignments which give the greatest score *and* which overlap with the ground-truth bounding box by at least 70% are selected [53]. New values for the mixture assignment m and part positions $\mathcal{P}$ replace the old values in $\mathcal{M}$ and $\mathcal{Z}$. This latent update scheme is similar to the one presented by Felzenszwalb *et al*. [38].

Relation to other Methods

Our method is closely related to the optimization framework of the Deformable Part-based Models (DPM) by Felzenszwalb *et al*. [38]. We observe the same notion of semi-convexity and define a similar upper bound (Equation 3.14) to motivate the optimization scheme. In the next section we will furthermore present and prove convergence guarantees of the optimization scheme. Felzenszwalb *et al*. also state these guarantees but do not prove them. Note that our method performs 'batch' learning by computing the exact subgradients of the energy function (Equation 3.7) while the method of Felzenszwalb *et al*. [38] performs stochastic learning and only approximates the subgradients of the energy function in each step of the learning scheme. Our method can therefore be considered more precise than the optimization scheme of Felzenszwalb *et al*.

The proposed optimization scheme can also be interpreted as an extended Multiple Instance (MI) SVM [3] scheme by treating all combinations of mixture assignments and part positions as an (exponentially large) 'bag' of instances for each positive example. In our approach negative training examples can be considered as bags with a cardinality of one.

The Concave-Convex Programming (CCCP) procedure by Yu *et al*. [145] is a method to solve the optimization problem that arises in structural SVMs. The procedure has the ability to consider latent variables for each training example and proposes an alternation scheme comparable to the one presented in this chapter. Zhu *et al*. [147] use a modification of the CCCP algorithm to drive learning of a DPM. In essence their particular implementation of the CCCP procedure is not significantly different from the method proposed in this chapter. However, it should be noted that the structural SVM formulation allows for more sophisticated (*e.g*. non-binary) loss functions to be used, which could

potentially lead to more powerful learning frameworks.

3.5.3 Convergence Guarantees

In this section we present convergence guarantees for the optimization scheme, which ensure a decrease of the energy during learning. Let us assume that function $z(\cdot)$ maps a set of model variables $\mathcal{O}$ to a set of latent variables $\mathcal{L}$, *i.e.* it performs inference on the positive training examples. Let us further assume that we do not perform any bootstrapping but only alternate between updating latent variables $\mathcal{L}$ and model variables $\mathcal{O}$ with a fixed training dataset $\mathcal{T}$. We now show that the proposed energy function in combination with the optimization scheme always decreases or maintains the energy by showing the following inequalities:

$$E\left(\mathcal{O}^{t+1}\middle|\,\mathcal{T}, z\left(\mathcal{O}^{t+1}\right)\right) \leq E\left(\mathcal{O}^{t+1}\middle|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right) \leq E\left(\mathcal{O}^{t}\middle|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right) \tag{3.18}$$

where $\mathcal{O}^t$ is a set of model variables at step t and $\mathcal{O}^{t+1}$ at step $t+1$ after optimization w.r.t. to $\mathcal{O}$, *i.e.* $\mathcal{O}^{t+1} = \arg\max_{\mathcal{O}} E\left(\mathcal{O}|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right)$. The above inequation holds because:

- $E\left(\mathcal{O}^{t+1}|\,\mathcal{T}, z\left(\mathcal{O}^{t+1}\right)\right) \leq E\left(\mathcal{O}^{t+1}|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right)$ is true because $z(\cdot)$ will only update the latent variables of the positive training examples, *i.e.* always improves or maintains their classification score. Consequently it will decrease or maintain the cost produced by the hinge loss and as a result decrease or maintain the energy.
- $E\left(\mathcal{O}^{t+1}|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right) \leq E\left(\mathcal{O}^{t}|\,\mathcal{T}, z\left(\mathcal{O}^{t}\right)\right)$ holds because the set of latent variables is fixed to $z\left(\mathcal{O}^{t}\right)$ and the L-BFGS optimizer on the model variables $\mathcal{O}$ is guaranteed to lower or maintain the energy.

The presentation of these convergence guarantees completely motivates the proposed optimization scheme.

Applicability of the Convergence Guarantees

Note that the convergence guarantees solely ensure a decrease of the energy for a *fixed training dataset*. It does not provide any indication for the generalization of the proposed optimization method, which commonly arrives in a local minimum of the energy function. The convergence guarantees also only hold when no bootstrapping is used. However, because the purpose of the bootstrapping process is to mine a final dataset, we interpret the convergence guarantees as operating on this final dataset.

3.6 Empirical Results

In this section we report experimental results, and demonstrate the positive effects of sharing parts in a DPM as well as sharing parts over multiple object classes. All experiments in this section will be carried out on the PASCAL VOC 2010 dataset [30].

3.6.1 Implementation Details

Before reporting qualitative and quantitative results in the next section, we first discuss the details of our implementation. DPMs have a large number of parameters to set prior to training and this section explains our choice of those parameters.

Feature Descriptors

As mentioned earlier we follow Felzenszwalb *et al*. [38] and use two layers of HOG features [23]. The first layer consists of cells of size 6×6 pixels and models fine appearance of an object class while the second layer (12×12 pixels) models coarse appearance of an object class. HOG features are computed as described in Section 2.3. On the first layer, HOG blocks overlap by one cell (6 pixels) while on the second layer they overlap by half a cell (also 6 pixels). Blocks consist of 2×2 cells and we apply the dimensionality reduction technique of Felzenszwalb *et al*. [38] to arrive at a 13-dimensional feature vector per block (see Section 2.3).

Part filters are of size 6×6 HOG blocks and use the first (fine) level of HOG features. As mentioned previously a root filter is used in addition to the p movable parts. The root filter acts on the second (coarse) level of HOG features – see also Figure 3.3.

Truncated Objects

We adopt our approach of Chapter 2 and pad the feature representation of an image by 50% of the window size on each side to detect truncated objects. HOG features in these areas are all set to 0. Details are given in Section 2.7.1.

In addition we follow Girshick *et al*. [53] and append visibility flags to the feature vector of each HOG block. These flags indicate whether a block is inside of the image (set to 0) or outside of the image *i.e*. padded (set to 1). A separate weight is learned for the visibility flag of each block in the learning scheme described in Section 3.5.

Initialization of Parts and Mixtures

To compute the initial mixture assignment of each positive training example k-means clustering with d cluster centers is applied to the aspect ratio (width/height) of the bounding boxes, where d is the number of mixture components [38].

To initialize parts we adopt the approach of Girshick *et al*. [53] by trying to arrange parts in such a way that they cover areas of high positive weights in each mixture component. An equal number of parts is initially assigned to each mixture component, *i.e*. using p parts overall results in $p/(r \times d)$ parts per mixture component, where r is the number of object classes. We adopt the approach of Felzenszwalb *et al*. [38] by training individual sliding window detectors, which provide an initial distribution of positive detector weights in each mixture component. An initial 'motorbike' detector with 3 mixture components is visualized in Figure 3.7. We now define a greedy initialization strategy for the anchor positions $\mathcal{A}$, consisting of two steps which are repeated until all parts are placed.

Step 1: to initialize a part's anchor position we select one mixture component and search for the position which maximizes the sum of all positive detector weights covered by that part. If a mixture component already has $p/(r \times d)$ initialized parts assigned to it, we do not consider it for the further placement of parts.

Step 2: to initialize anchors $\mathcal{A}$ in the other mixture components (for part sharing) we extract the positive weights from the area the part covers in the mixture component to which it has been initialized to. We then convolve this set of positive part weights with the positive detector weights of the other mixture components in a sliding window manner. The position with the highest 'score' is used as the anchor position for the part in the respective mixture component.

Once initialized the positive detector weights covered by the part in each mixture component are multiplied by a factor of 0.5 when parts are shared. In case the part is not shared the positive detector weights of the mixture component the part belongs to are multiplied by a factor of 0.01. This ensures that the next part will not be initialized to the same position. We use different factors for models with and without part sharing because for DPMs without part sharing each part is fully 'focused' on the mixture component to which it is assigned. For models with part sharing however, a part is used in all mixture components according to the combination weights $\boldsymbol{\beta}$.

We repeat this greedy selection scheme several times by randomly selecting mixture components in step 1 as well as randomly selecting one of the the 5 highest ranked part positions for each mixture component in steps 1 and 2. We finally chose the part covering

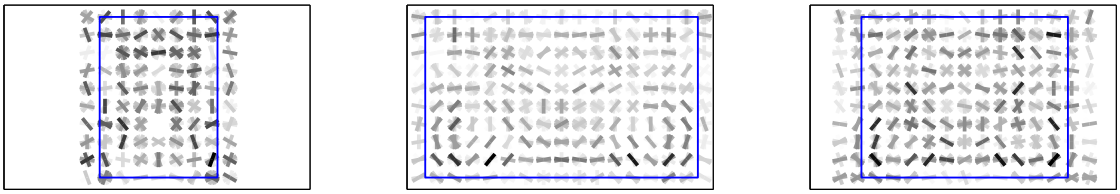


Figure 3.7: Initial sliding window 'motorbike' detector with average bounding box. We show the initial sliding window detector of the 'motorbike' class used for part initialization. The "average" (best fitting) bounding box per mixture component is shown in blue. The window frame is drawn in black.

which provides the largest amount of accumulated positive detector weights [53].

Window Size

Similar to our implementation in Chapter 2 each training example (in each mixture component) is normalized to have a fixed height according to Table 2.1. We compute the 75th percentile of all height-normalized bounding box widths in each mixture component and set the width of a window to the largest of these 75th percentiles. We add a 6 pixel border around the windows to capture possible recognition cues in the background in close vicinity to the object.

For root parts and the sliding window detectors used for part initialization we ignore features that lie outside the 75th percentile of all bounding box widths (+ the 6 pixel border) in each mixture component – see Figure 3.7. We found that these areas did not hold any valuable recognition cues.

Following our approach in Chapter 2 we predict an "average" bounding box per mixture component, which is chosen in such a way that it maximizes overlap with all ground-truth bounding boxes assigned to a mixture component. This average bounding box is shown in blue color in Figure 3.7.

Initial Training Examples

We extract initial positive training examples by centering the window on the ground-truth bounding box of each object instance in the `train` subset of the PASCAL VOC 2010 dataset. For the initial feature extraction we only consider examples which are not flagged 'occluded', 'truncated' or 'difficult' in the provided annotation. We left/right-flip every positive training example to create additional mirrored 'virtual' training examples.

Initial negative training examples are accumulated by extracting features from the center window of the first 200 training images not containing the object class(es) of interest.

Training Protocol

The training protocol consists of a set of steps which gradually allows the learning scheme to optimize the energy w.r.t. a larger subset of model variables in $\mathcal{O}$. We have found that gradually learning model variables $\mathcal{O}$ increases the chances of arriving in a well generalizing local (or even global) minimum of the energy function. Optimizing the energy function w.r.t. to all model variables $\mathcal{O}$ from the start very often causes optimization to 'get lost' in a region of the energy surface with badly generalizing local minima.

In the *first step* of training we only learn initial part appearance $\mathcal{W}$ by fixing parts to their respective anchor positions *i.e*. they are not allowed to move during inference. Values of $\boldsymbol{\beta}$ are fixed to 0.5 in the case of part sharing. In case where parts are not shared, values of $\boldsymbol{\beta}$ are initialized to 1 for the mixture component to which each part belongs. For root parts values of $\boldsymbol{\beta}$ set to 1 for their respective mixture component independent of whether parts are shared or not. We use 2 rounds of latent variable updates (outer loop) and 20 rounds of bootstrapping (inner loop).

In the *second step* parts are allowed to move, *i.e*. they are not fixed to their respective anchor positions. The quadratic terms of the spatial priors $\mathcal{S}$ are initialized to 0.1 [38] (see Equation 3.2) and values of $\boldsymbol{\beta}$ remain fixed. This step allows us to obtain a good initialization of spatial priors $\mathcal{S}$. Again 2 rounds of latent variable updates (outer loop) and 20 rounds of bootstrapping (inner loop) are used.

In the *main step* we 'activate' part sharing by allowing for optimization w.r.t. $\boldsymbol{\beta}$, performing 6 rounds of latent updates on $\{\mathcal{Z}, \mathcal{M}\}$ and 20 rounds of bootstrapping. Values of $\boldsymbol{\beta}$ of the root parts are not changeable and fixed to 1 *i.e*. root parts are not shared in our framework.

As a *final step* we fix $\boldsymbol{\beta}$ again and perform 20 final rounds of bootstrapping on 800 randomly picked training images in every round. We have found empirically that this final step creates better performing detectors *c.f*. using the detectors from the main step.

Bootstrapping

During bootstrapping we extract 3 false positives per scale per mixture component. Once completely scanned, the image is resized by a factor of 1.2. We exit the inner loop early if the number of extracted false positives drops below 200 for each mixture component.

Regularization & Constraints

Cross-validation on the `val`-subset is used to set the parameter λ. We consider suitable values in the range of 10^{-1} to 10^{2}, which have been determined in initial experiments. Cross-validation also serves the purpose of reducing the risk of arriving at a poor local minimum of the energy function: repeatedly relearning the entire detector at different values of λ improves the chance to arrive at a well generalizing solution.

Similar to Felzenszwalb *et al*. [38] a lower bound of 0.01 is enforced on the quadratic terms of the spatial priors in $\mathcal{S}$ (Equation 3.2) to ensure convexity and a 'not-too-flat' surface of the deformation cost. In addition we fix the 'absolute' components of the spatial priors $\mathcal{S}$ to 0 (Equation 3.2), causing them to be centered on their anchor position. We have found that this constraint does not harm performance and improves the rate of convergence of the learning scheme.

L-BFGS Implementation

To optimize model variables $\mathcal{O}$ in the inner loop, we use the L-BFGS optimizer of Byrd *et al*. [16]. If the values of $\boldsymbol{\beta}$ are fixed (*i.e*. parts are not shared or the training protocol is in the first, second of final step) we perform 250 L-BFGS iterations on $\{\mathcal{W}, \mathcal{S}\}$. In the case that values of $\boldsymbol{\beta}$ are not fixed (*i.e*. the main step is executed *and* parts are shared) we alternate between optimizing the energy w.r.t. $\{\mathcal{W}, \mathcal{S}\}$ and w.r.t. to $\boldsymbol{\beta}$ as previously discussed. For each subproblem 50 L-BFGS iterations are used and overall we perform 5 updates on $\{\mathcal{W}, \mathcal{S}\}$ and $\boldsymbol{\beta}$.

When optimizing $E(\cdot)$ we always re-initialize $\mathcal{O} = \{\mathcal{W}, \mathcal{S}, \boldsymbol{\beta}\}$ to the solution of the previous run, hence establishing a "gradient descent"-like learning scheme.

Early Stopping

Throughout each step in the training protocol we exit the outer loop early if the energy converged. We consider the energy as converged if

$$\frac{E^{+}\left(\mathcal{O}|\,\mathcal{T}, \mathcal{L}^{k+1}\right)}{E^{+}\left(\mathcal{O}|\,\mathcal{T}, \mathcal{L}^{k}\right)} > 0.999 \tag{3.19}$$

where $E^{+}\left(\mathcal{O}|\,\mathcal{T}, \mathcal{L}^{k}\right)$ is the accumulated loss of all positive training examples *before* the update of the latent variables and $E^{+}\left(\mathcal{O}|\,\mathcal{T}, \mathcal{L}^{k+1}\right)$ is the accumulated positive loss *after* the update of the latent variables. The above equation will be true if both energies are

almost similar, which indicates that the recent update of latent variables only provides a very small decrease of the energy.

Notes on the Final Solution

The number of inner/outer loop runs, the number L-BFGS iterations as well as the bootstrapping setup were picked to ensure good convergence of the latent values $\mathcal{L}$ as well as the model variables $\mathcal{O}$ *in combination* with a reasonable training time.

Especially for low values of λ the found solution might not always precisely correspond to the true local minimum of the energy function. Reaching this minimum requires significantly more L-BFGS iterations as well as a more conservative early-stopping for bootstrapping, which in return would significantly prolong training time. In our implementation we therefore choose a reasonable number of iterations to yield a solution of good precision.

Testing Protocol

The test protocol is identical to the training protocol *i.e*. a scale-factor of 1.2 and a window stride of 6 pixels is used. To create a final set of windows for a test image we adopt the greedy non-maximum-suppression scheme of Felzenszwalb *et al*. [38], which has already been explained in Section 2.7.1. We use cross-validation on the `val`-subset of PASCAL VOC 2010 [30] to find the best overlap value in the range of 0.2 to 1.0.

3.6.2 Quantitative Results

For quantitative performance evaluation we use the PASCAL VOC 2010 [30] dataset and methodology – precision/recall curve with bounding box overlap of 50% (see Section 1.2.1).

Terminology

Throughout this section we compare DPMs using part sharing to DPMs not sharing parts. We generally state that a DPM uses p parts and each mixture component additionally uses one root part. In case of part sharing there is no strict assignment between mixture components and parts. However, in the case that we do not share parts each DPM essentially represents a model of Felzenszwalb *et al*. [38] and we distribute the parts equally over the mixture components. For example, using 12 parts and 3 mixture components

AP (on `test`)	aero	bicycle	bird	boat	bottle	bus	car
not shared	34.4	33.5	0.2	4.6	1.4	34.8	32.3
shared	**36.8**	**33.9**	**0.4**	**4.9**	**4.5**	**43.3**	**35.3**
shared (H+C)	–	36.0	–	–	–	41.0	36.1
`voc-R4` [53]	52.4	54.3	13.0	15.6	35.1	54.2	49.1
AP (on `test`)	cat	chair	cow	table	dog	horse	person
not shared	5.6	**7.7**	**6.6**	**9.9**	5.8	20.6	28.1
shared	**8.4**	7.4	6.0	9.8	**7.8**	**22.9**	**30.7**
shared (H+C)	–	–	8.4	–	–	–	–
`voc-R4` [53]	31.8	15.5	26.2	13.5	21.5	45.4	47.5
AP (on `test`)	mbike	plant	sheep	sofa	train	tv	*mean*
not shared	28.6	5.1	14.8	**10.9**	17.7	25.4	*16.1*
shared	**31.6**	**5.6**	**17.1**	10.3	**22.5**	**25.8**	***18.2***
shared (H+C)	35.5	–	24.7	–	–	–	–
`voc-R4` [53]	51.6	9.1	37.8	33.0	41.5	38.0	*33.8*

Table 3.1: PASCAL VOC 2010 results (`comp3`, `test`-set). We show Average Precision (AP) for models with 3 mixtures and 12 parts (*not shared* and *shared*). AP improves for 16 of 20 object classes when parts are shared *c.f.* models without part sharing. Mean AP improves from 16.1% for models without part sharing to 18.2% for models using part sharing. In addition we present AP results for six object classes for a DPM with shared parts and HOG+CHOG features (*shared H+C*). For this detector AP improves for 5 of 6 object class compared to only using HOG features (*shared*). We also list the results of Girshick *et al.* (`voc-release4` [53]) for the sake of completeness. This model is significantly more complex than ours – see text for details.

means 4 parts per component. Similarly using 8 parts in 2 detectors, each using 2 mixture components, means 2 parts per component per detector.

Evaluation of Part Sharing within Classes

Table 3.1 provides a full set of results for the PASCAL VOC 2010 challenge (`comp3`) for models with 3 mixture components and 12 parts (first and second row of Table 3.1). AP improves for 16 of 20 classes while mean AP improves from 16.1% to 18.2% – a relative improvement of 13.0%.

Improvements include the 'rigid' object classes such as 'bicycle', 'bus', 'car', 'motorbike' or 'diningtable'. In these object classes part sharing is natural as parts are well defined (*e.g.* wheel of car) and are present across multiple viewpoints. For the 'bus' object class, AP improves from 34.8% to 43.3%, yielding a relative improvement of 24.4%. Other improvements can also be observed for the 'aeroplane' class for which AP increases

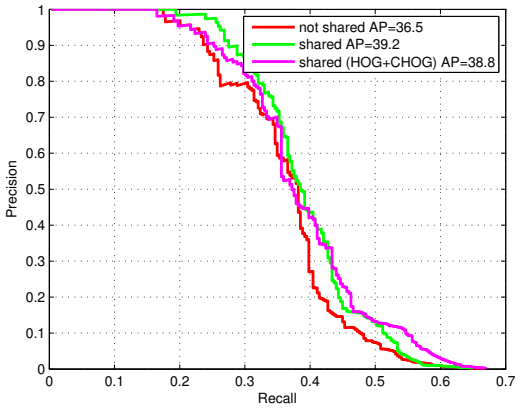
from 34.4% to 36.8% or for 'motorbike' (28.6% to 31.6%). Furthermore AP improves from 17.7% to 22.5% for the 'train' object class – a relative improvement of 27.1%. AP improvements are however not limited to these rigid object classes as can for example be observed for the 'horse' class which improves from 20.6% to 22.9% or the 'person' class (28.1% to 30.7%).

In addition to comparing DPMs with and without shared parts using HOG features we also compare to a DPM with shared parts and HOG+CHOG features in Table 3.1 . The enriched feature vector generally results in increased AP – a notable improvement is the 'sheep' object class which improves from 17.1% to 24.7% when comparing HOG to HOG+CHOG. Given the observed improvements in AP it would appear worthwhile to further research the combination of a DPM with different feature descriptors rather than using pure HOG.

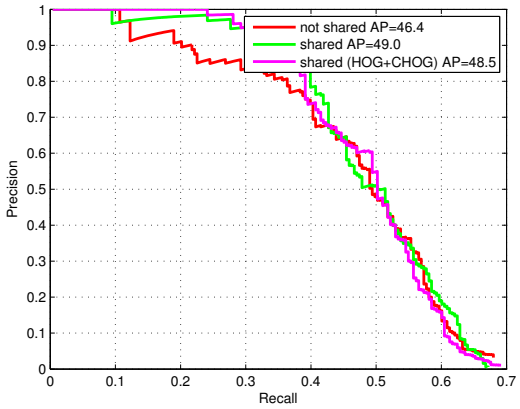
Table 3.1 also states APs for the DPM of Girshick *et al*. (`voc-release4` [53]). These models yield very good results and are significantly more complex than the models presented in this chapter. Girshick *et al*. use 6 mixture components, which correspond to the left/right-reflections of three different viewpoints. In addition, their method uses a more advanced feature descriptor by combining HOG features with signed and unsigned gradients. The method of Girshick *et al*. [53] also starts detection at a scale of 0.5 (twice the image resolution) and is therefore able to detect very small objects. Context-based post processing of the detection results is used to further improve detection accuracy.

Figure 3.8 presents precision/recall curves for selected object classes. A model with 3 mixture components and 12 parts was evaluated on the `val` subset to create these curves. The results show that AP improves for all presented object classes and it can furthermore be seen that precision increases for most recall levels when parts are shared. For some classes the difference in precision at specific recall points can be significant – see *e.g*. the 'sheep' class at a recall level of 20%. Precision/recall curves for the remaining object classes can be found in appendix A.2.

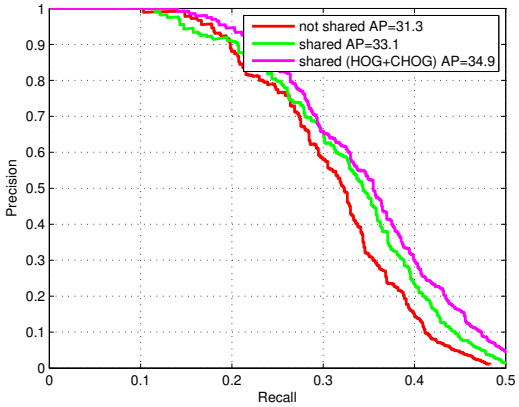
In addition, we also present precision/recall curves for DPMs using part sharing and HOG+CHOG features (see Chapter 2). The curves establish that the usage of HOG+CHOG features in comparison to pure HOG features can further increase precision and similar recall levels – see *e.g*. Figure 3.8 (c) in which precision increases at all levels of recall levels for the 'car' class.



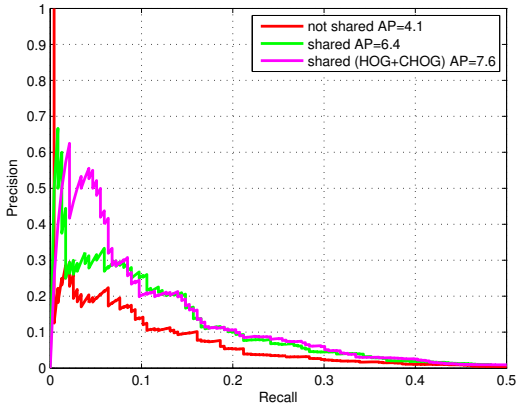
(a) bicycle



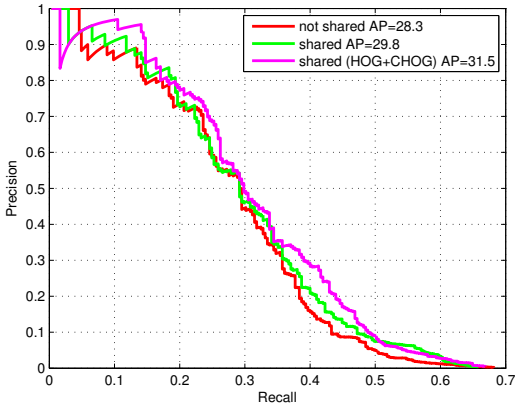
(b) bus



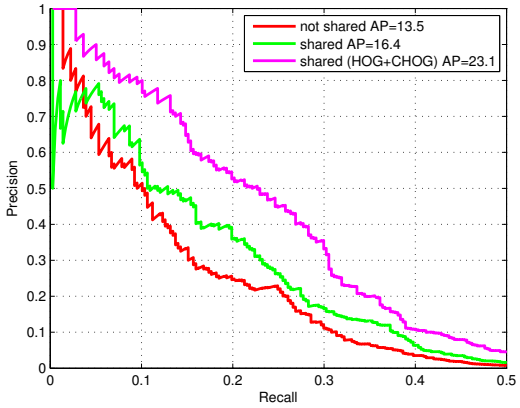
(c) car



(d) cow



(e) motorbike



(f) sheep

Figure 3.8: Precision/recall curves for selected PASCAL VOC 2010 classes. We compare models with 3 mixture components and 12 parts not using shared parts (red curves) and using shared parts (green). AP improves for all shown classes and precision improves for most recall levels for models with part sharing compared to their counterparts without part sharing. HOG+CHOG features can usually further improve AP.

Varying Number of Parts

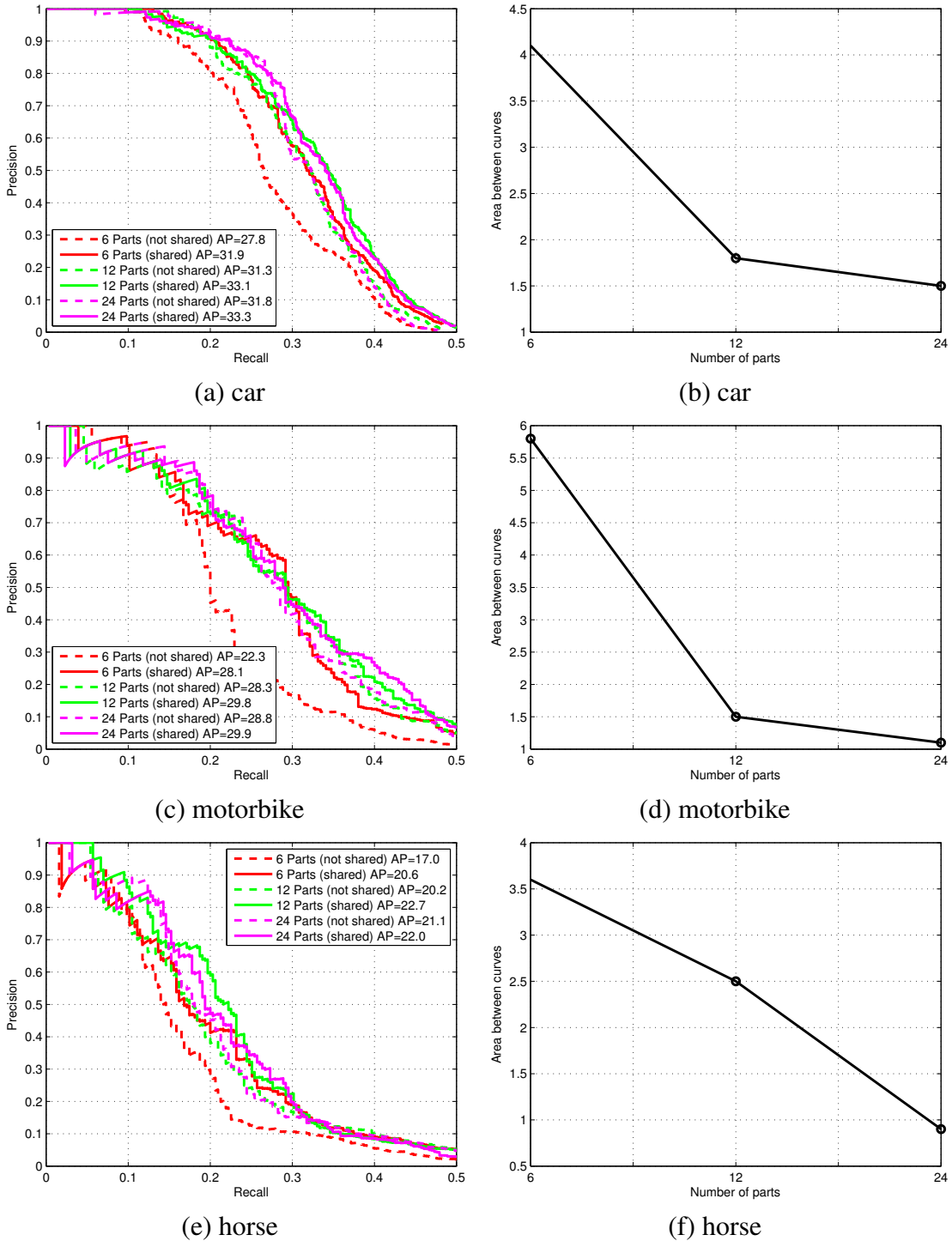
To evaluate single class performance for a varying number of parts we learn individual detectors for the 'car', 'motorbike' and 'horse' classes of the PASCAL VOC 2010 dataset using 3 mixture components. We establish that part sharing can provide better accuracy when using models with fewer parameters by comparing precision/recall curves for models having 6, 12 and 24 parts with and without part sharing.

Figure 3.9 (a) shows the precision/recall curves for a 'car' detector with varying number of parts. We observe that part sharing always improves performance. Furthermore the detectors using 6 and 12 shared parts (green solid line) give AP values of 31.9% and 33.1%, outperforming the detector using 24 non-shared parts (AP 31.8%, magenta dotted line). This establishes our idea of performing 'more with less' given that the detector using 24 non-shared parts utilizes significantly more parameters than the detector with 12 shared parts – $18,915$ vs. $12,951$ in our implementation. We hypothesize that models with fewer parameters are easier to learn and their optimization is less prone to arriving at badly generalizing solutions.

Figure 3.9 (b) also shows that sharing parts is beneficial by plotting the area between precision/recall curves. We compute the area between precision/recall curves by subtracting the AP of the models with non-shared parts from their counterparts with shared parts. A positive area between the curves indicates an improvement in performance for models with shared parts. As expected, the area between the curves reduces with an increasing number of parts because each mixture component can make use of more parts for models without part sharing. However, there is still a substantial improvement for detectors using 24 parts, indicating the effectiveness of part sharing in DPMs.

In Figure 3.9 (c) we plot precision/recall curves for a 'motorbike' detector using 6, 12 and 24 parts with and without part sharing. AP always improves when parts are shared – this is also indicated by a constantly positive area between the curves in Figure 3.9 (d). The detector with 24 shared parts gives the best AP (29.9%). We were not able to explore configurations with more than 24 parts due to the large memory requirements of such models. The model with 6 shared parts performs significantly better than the model with 6 non-shared parts – 28.1% vs. 22.3%, a relative improvement of 26%.

We also perform a similar comparison for the 'horse' class (Figures 3.9 (e, f)). As before, detectors with part sharing outperform their counterparts without part sharing. AP peaks for models with 12 shared parts. Increasing the number of parts to 24 shared parts does not provide a further increase in AP.



(a) car (b) car

(c) motorbike (d) motorbike

(e) horse (f) horse

Figure 3.9: Results of part-sharing for PASCAL VOC 2010. We plot precision/recall curves for models with 6, 12, and 24 parts with and without part sharing for the (a) 'car', (c) 'motorbike' and (e) 'horse' class. In addition, the area between curves for models with and without part sharing is shown in (b, d, f). Part sharing generally improves performance. While the area between curves decreases with an increasing number of parts, the performance improvement is still substantial even for the DPMs with 24 parts.

Varying Number of Mixture Components

Here we compare models with part sharing to models without part sharing while holding the number of parts fixed and varying the number of mixture components. This experiment shows how detection accuracy is influenced under the impression of an increasing number of mixture components given a fixed number of parts.

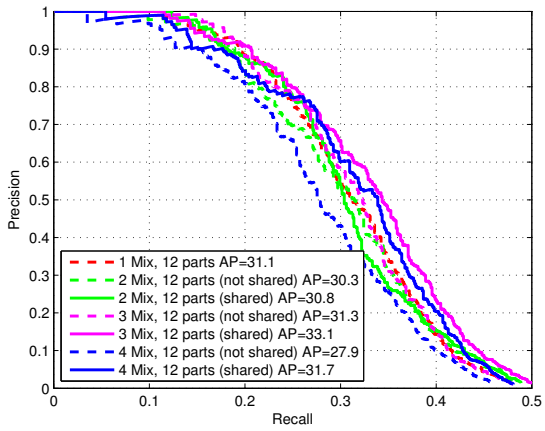
Figure 3.10 presents results for detectors with 12 parts and 1 to 4 mixture components. We disable part sharing for models with 1 mixture component because all parts are assigned to a single mixture component and optimization w.r.t. to combination weights $\boldsymbol{\beta}$ would be useless.

For the 'car' class (Figure 3.10 (a, b)) part sharing is beneficial for all configurations and the area between the curves increases with the number of mixture components. This is to be expected as the detector with 4 mixture components can only use 3 parts per mixture component if they are not shared. Such a low number of parts per mixture component is often not sufficient to efficiently model the appearance and spatial configuration of an object class.

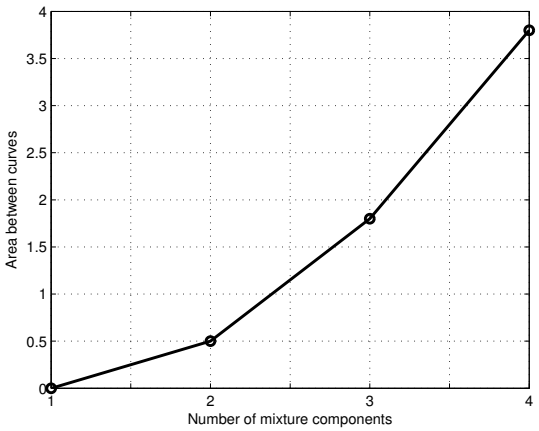
In Figure 3.10 (c, d) and (e, f) similar experiments are performed for the 'motorbike' and 'horse' class. In both cases detectors using 4 mixture components perform weakly. There seems to be no benefit from increasing the number of mixture components beyond 3. We have found in our experiments that the additional mixture component does not represent a significant part of the positive training data. For the 'motorbike' class the detector using 1 mixture component performs very well but is outperformed by 3 mixture components with shared parts.

The 'horse' class benefits from additional mixture components even when parts are not shared. AP improves from 17.1% for 1 mixture component to 20.2% for 2 mixture components without shared parts. The best result is again given by the model with 3 mixture components and shared parts (AP 22.7%).

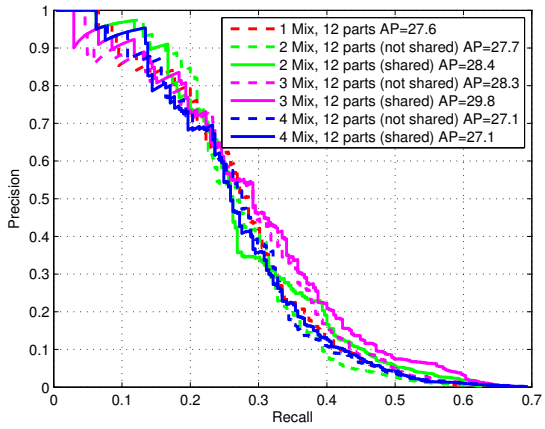
Overall we conclude that detectors with a single mixture component perform surprisingly well when parts are not shared, sometimes even outperforming models with multiple mixture components. Models with part sharing on the other hand can make more efficient use of all parts and therefore offer a substantially increased AP for models with multiple mixture components *c.f.* models with a single mixture component. For all presented object classes the best AP is achieved for models with 3 mixture components.



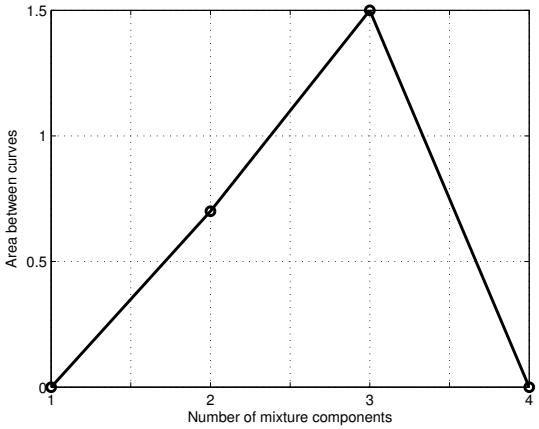
(a) car



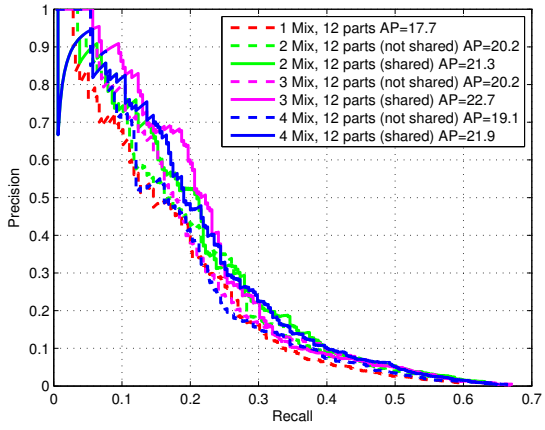
(b) car



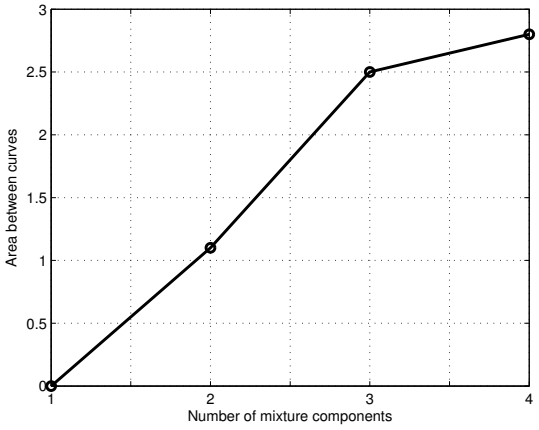
(c) motorbike



(d) motorbike



(e) horse



(f) horse

Figure 3.10: Results of part-sharing for PASCAL VOC 2010. We plot precision/recall curves for models with 1, 2, 3 and 4 mixture components. All models use 12 parts. Best performance is achieved for models with 3 mixture components and shared parts. The area between the curves generally increases with the number of mixture components as fewer parts are available to each mixture component if parts are not shared.

Varying Number of Parts and Mixture Components

In Figure 3.11 we evaluate models with a varying number of parts *and* mixture components to show how detection accuracy is influenced under the impression of an increasing number of mixture components given a fixed number of parts *per mixture component*. Each detector uses 4 parts per mixture component and we compare models with 1 (4 parts), 2 (8 parts), 3 (12 parts) or 4 (16 parts) mixture components.

Part sharing always improves AP with the exception of the 'motorbike' detector using 2 mixture components. Additional mixture components usually improve performance but 4 mixture components do not provide a further gain in AP. Three mixture components with 12 shared parts can be considered the best configuration for all object classes in Figure 3.11.

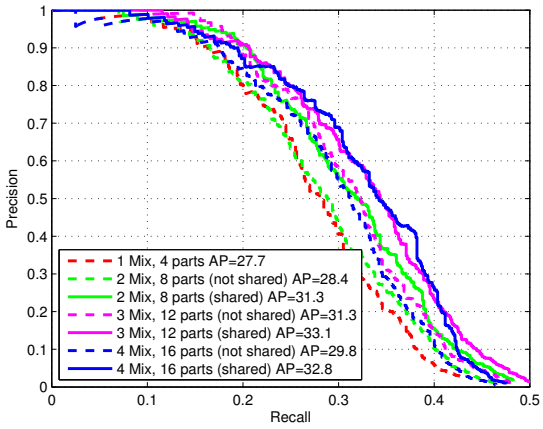
Area between the curves is stable for the 'car' detector and increases for the 'motorbike' detector (with the exception of 2 mixture components) as well as the 'horse' detector. This is expected because part sharing is still beneficial even if the number of parts increases with the number of mixture components.

We conclude from this experiment that part sharing is useful even if the number of parts linearly increases with the number of mixture components. As in the previous experiment we find the models with 4 mixture components do not provide a further gain in detection accuracy compared to models with 3 mixture components.

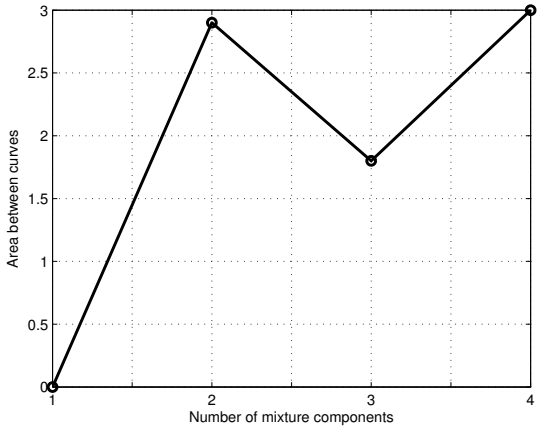
Evaluation of Part Sharing across Classes

Section 3.5 introduced the possibility of jointly learning object detectors for multiple classes. We have evaluated multi-class performance by jointly training 'motorbike' and 'car' detectors as well as 'horse' and 'cow' detectors. We choose these pairs of object classes as they can be considered 'related' in the sense that they share object parts such as wheels or legs.

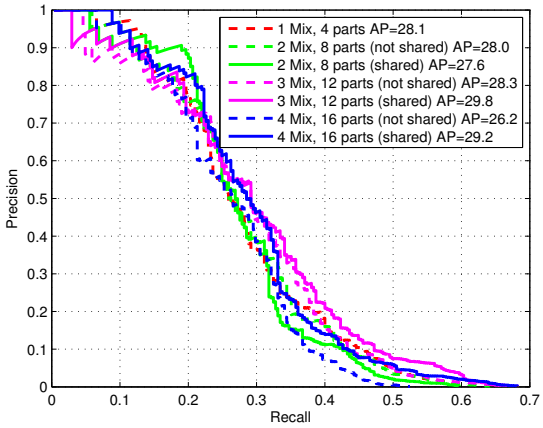
The results for these experiments are presented in Figure 3.12. Each coupled detector uses 12 parts and 3 mixture components *per object class*. This means that without part sharing each mixture component can make use of $2 = 12/(2 \times 3)$ parts . For the 'motorbike+car' detector an improvement in AP from 23.4% to 24.5% can be observed for the 'motorbike' class and 27.3% to 29.5% for the 'car' class when sharing parts among all mixture components of both object classes. AP values also improve for the 'horse+cow' detectors: AP for 'horse' class increases from 16.0% to 18.4% while for the 'cow' class it increases from 2.8% to 4.7% when parts are shared.



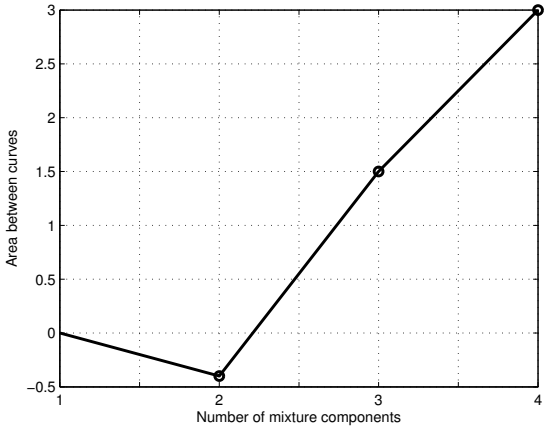
(a) car



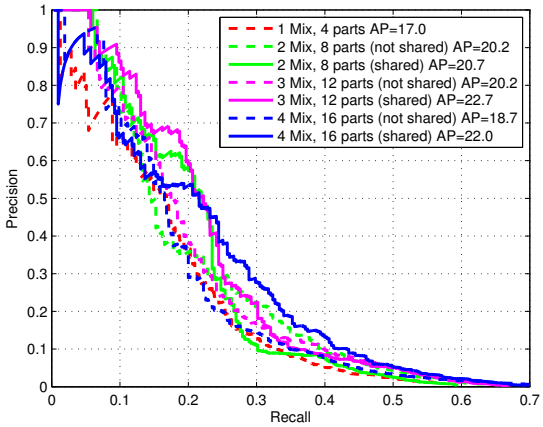
(b) car



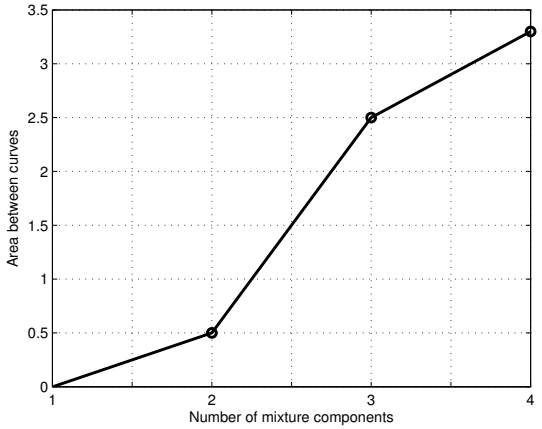
(d) motorbike



(e) motorbike

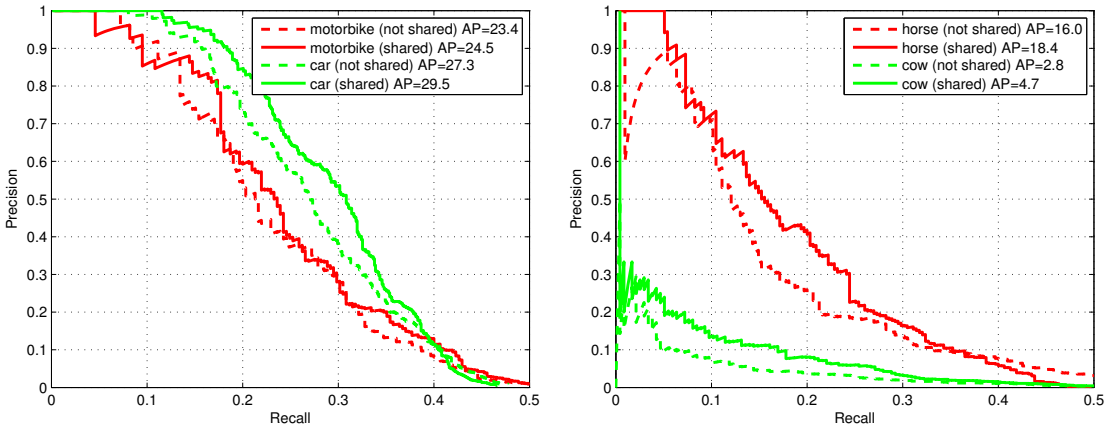


(g) horse



(h) horse

Figure 3.11: Results of part sharing for PASCAL VOC 2010. Precision/recall curves for models with 4 parts per mixture component are shown. AP usually improves with number of mixture components but 4 mixture components do not provide an additional gain over models with a smaller number of mixture components. Area between curves remains stable or grows with number of mixture components, indicating that part sharing is especially effective for models with many mixture components.



(a) motorbike, car (b) horse, cow

Figure 3.12: Results of part sharing for jointly trained PASCAL VOC 2010 classes. (a) Results of part sharing for jointly trained 'motorbike' and 'car' detectors. AP improves for both classes, indicating the applicability of part sharing to multiple object classes. (b) AP also improves for 'horse' and 'cow' classes when learned in combination with shared parts *c.f.* models without part sharing.

These results are substantial and suggest that it would be fruitful to further explore the possibility of joint learning of object detectors for multiple classes while sharing part responses among them.

3.6.3 Qualitative Results

Learned Features

In the following we visualize 'car', 'motorbike' and horse detectors to further establish the effects of part sharing in DPM. Figure 3.13 visualizes the positive detector weights of each orientation bin in each block as well as spatial priors and combination weights $\boldsymbol{\beta}$ of a car detector with 6 shared parts in 3 mixture components. To visualize positive detector weights each part filter is multiplied with its corresponding combination weight in $\boldsymbol{\beta}$. If multiple parts contribute to a HOG block in the visualization we accumulate their positive detector responses. Because root filters are coarse, their visualization is not very intuitive and we omit them here. Spatial priors of each part are visualized in the second row of the figure by plotting the cost of moving the respective part into a specific position. The color of the frame around each spatial prior represents the part number. The bottom row of Figure 3.13 shows the anchor positions of each part as well as the combination weights $\boldsymbol{\beta}$ of each part in each mixture component – warm colors represent that the part is strongly

used within a mixture component while cold colors represent that the part is weakly used. Note that some parts may share an anchor position and are therefore not visible in these plots.

Figure 3.13 clearly shows the effect of part sharing as most parts reappear across mixture components, *e.g*. the two roof parts (light blue and red outline in the spatial prior plots) are similar across all three mixture components. Note that while their appearance is identical across mixture components, anchor positions and spatial priors are different for each mixture component. This is to be expected given the definition of part sharing (Section 3.4). The part with the light green frame represents the wheel of a car and is used in mixture components 2 and 3. In mixture component 1 the location of this part is poorly initialized and hence the corresponding β-weight is low. This shows how DPMs with shared parts are able to overcome poor initialization by adapting the combination weights in order to decrease the importance of a specific part in a mixture component.

Figure 3.14 shows a car detector with 12 parts in 3 mixture components. Compared to the 6-part model (Figure 3.13), the appearance of the car is clearer because more parts are available to describe the object. The mixture components correspond to front/back-facing cars (component 1), side-facing cars (component 2) and three-quarter view cars (component 3).

A shared 12-part motorbike model is visualized in Figure 3.15. The first mixture component of this detector corresponds to front/back-facing motorbikes, the second models side-facing motorbikes the third three-quarter view object instances. Figure 3.16 visualizes a horse detector with 12 shared parts and 3 mixture components. The same correspondence of mixture components to viewpoints as in the previous example can be observed.

Part Locations

The proposed models with shared parts are able to efficiently reuse the appearance of object parts across multiple mixture components, which often correspond to individual viewpoints on an object class. Figure 3.17 shows detections of the 'car' class for a detector with 12 shared parts and 3 mixture components. In addition to each predicted bounding box, we show the location and extent of a single part, corresponding to the wheel of a car. The figure conveys that the proposed DPMs with part sharing are able to reuse the appearance of a part across multiple mixture components.

Figure 3.18 plots a similar set of detections for the 'motorbike' class. The part in this example represents the right wheel of the motorbike. We are able to recognize the shared

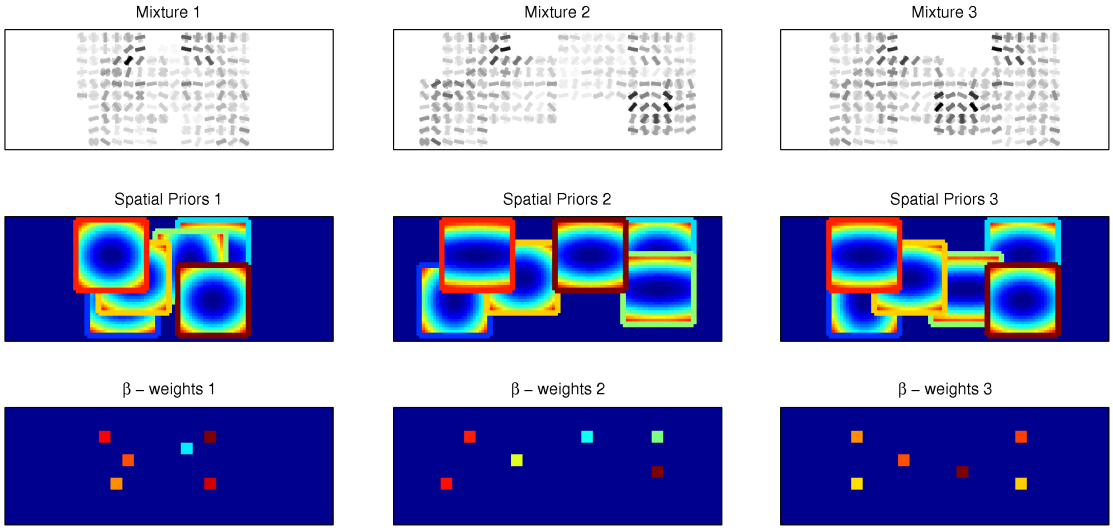


Figure 3.13: Car detector with 6 shared parts and 3 mixture components. Positive detector weights are shown in the top row of plots, spatial priors in the middle row and combination weights β in the bottom row. Positive detector weights of each part are individually weighted by the respective combination weight in β. The part number is color coded by the frame around each spatial prior. The effect of part sharing can clearly be observed – parts reappear across mixture components.

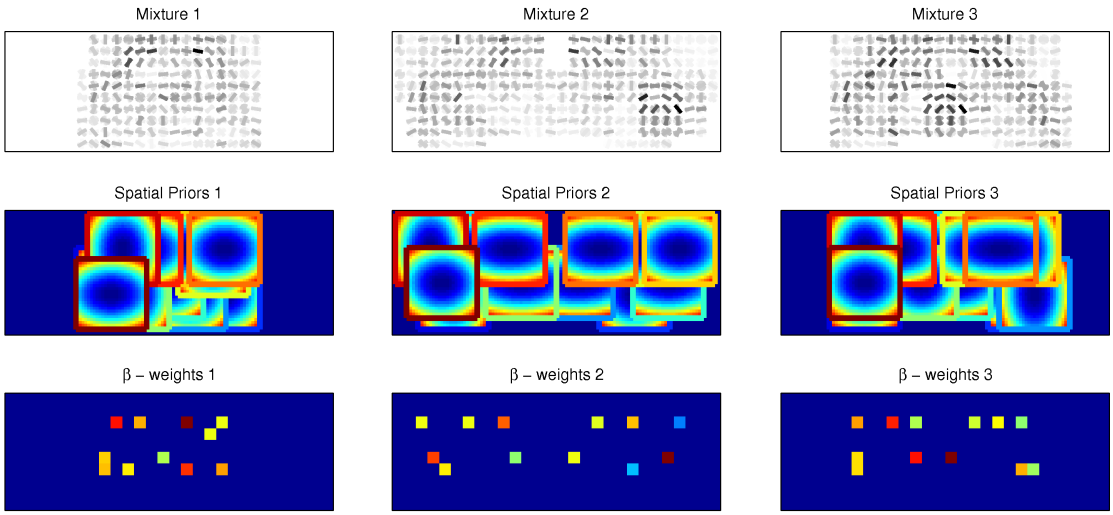


Figure 3.14: Car detector with 12 shared parts and 3 mixture components at $\lambda = 6$. Compared to the 6-part model of Figure 3.13 the appearance of the car is much clearer in these models due to the additional parts. Mixture components correspond to front/back-facing cars (component 1), side-facing cars (component 2) and three-quarter view cars (component 3). The part with the cyan frame in the spatial prior plots represents the wheel of a car and is shared between mixture components 2 and 3.

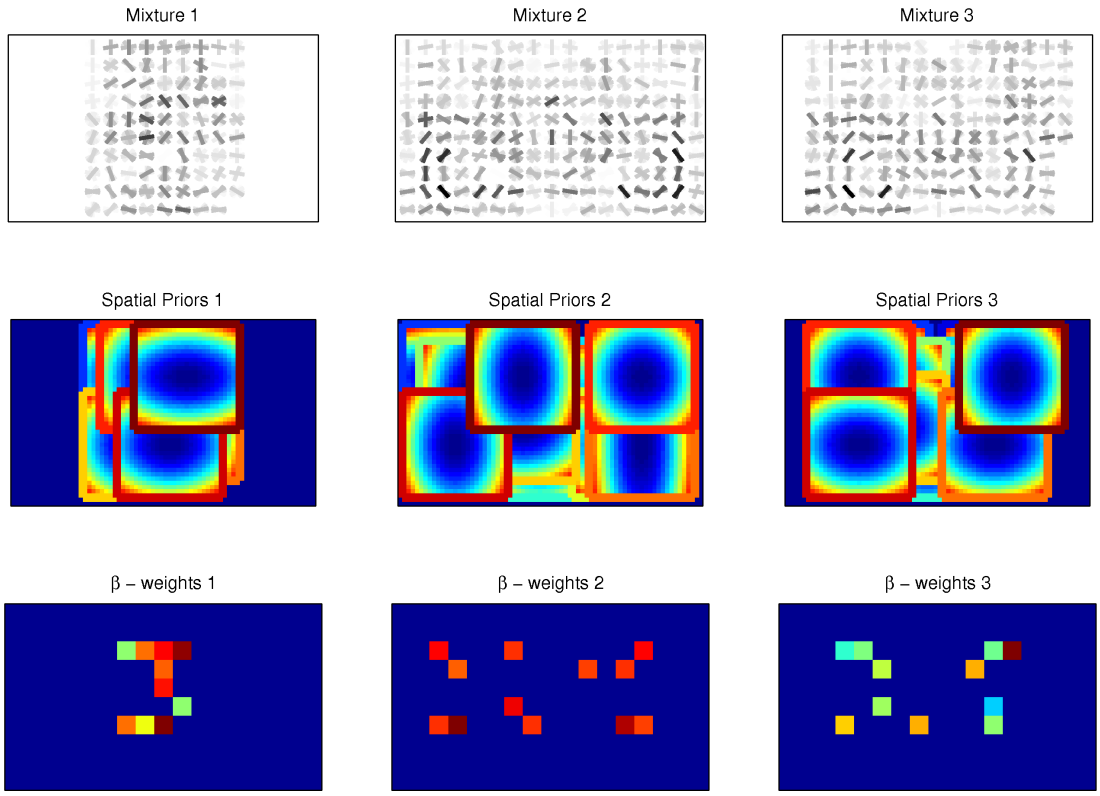


Figure 3.15: Motorbike detector with 12 shared parts and 3 mixture components. Mixture components correspond to front/back-facing motorbikes (component 1), side-facing motorbikes (component 2) and three-quarter view motorbikes (component 3). Many parts, such as the wheels of the motorbike, are shared across mixture components.

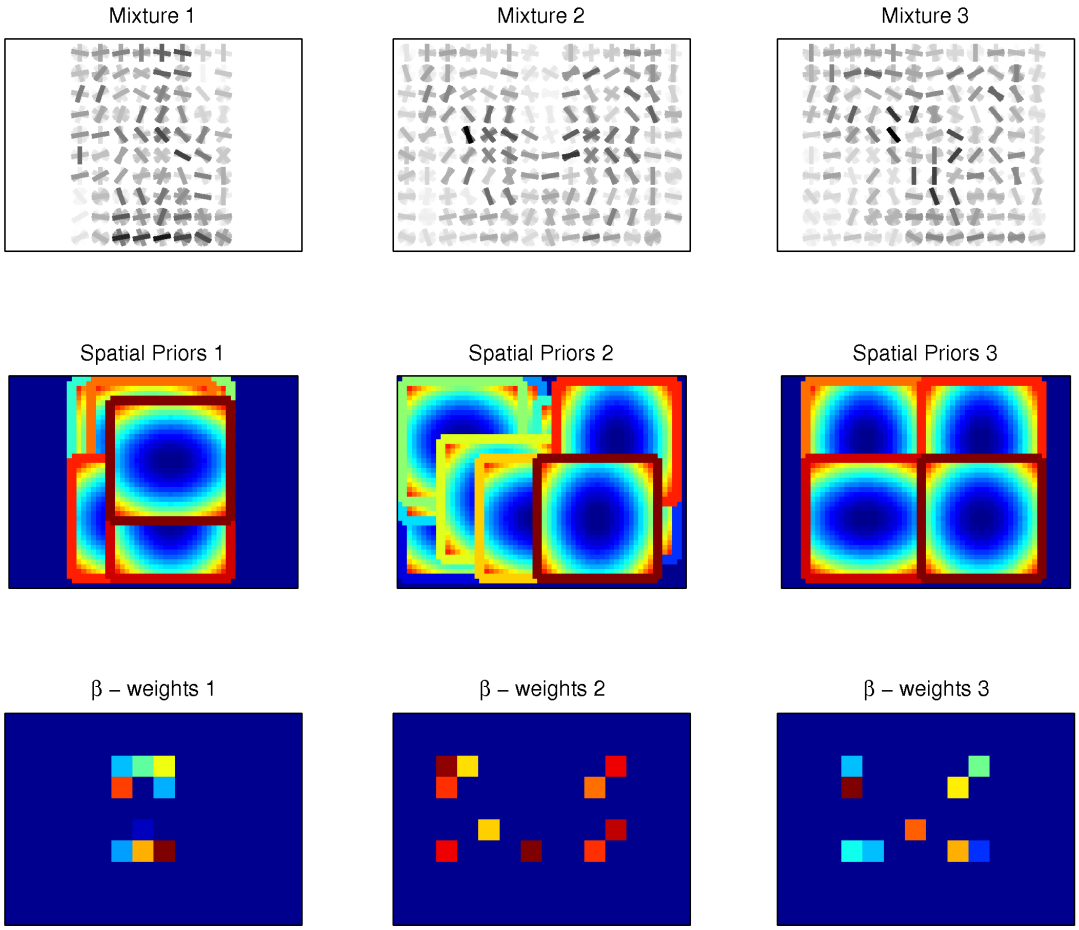


Figure 3.16: Horse detector with 12 shared parts and 3 mixture components . Spatial priors in this detector mainly appear homogeneous due to a high regularization parameter λ, which was set during cross validation.



Figure 3.17: Part locations for 'car' detections. We plot the location of a specific part in multiple detections of the 'car' class. The part, which corresponds to the wheel of a car, is recognized across viewpoints and efficiently reused in different mixture components. Note that this model does not have the ability to distinguish between left and right facing object instances but reliably recognizes the part either way.

part across viewpoints and in multiple mixture components. The part mainly appears in detections of three-quarter view and side-facing motorbikes. Sharing the part in these components is natural as the visual appearance of the part only changes modestly.

Figure 3.19 presents detections of the 'horse' class and tracks a part which corresponds to the neck/head portion of a horse. Similar to previous examples the part is successfully incorporated into mixture components representing different viewpoints, *e.g.* side-facing and front/back-facing horses.

3.7 Conclusions

In this chapter we have proposed two extensions to the deformable part-based model (DPM) of Felzenszwalb *et al.* [38]:

1. In the proposed framework part appearance is shared across mixture components while maintaining independent spatial configurations w.r.t. each mixture component. In comparison to the original DPM [38], this contribution allows us to reuse part appearance in different mixture components instead of learning the appearance of the same part for each mixture component separately.

2. We proposed an energy function to jointly learn part appearance $\mathcal{W}$, spatial configuration $\mathcal{S}$ and combination weights $\boldsymbol{\beta}$ for multiple mixture components. The combination weights $\boldsymbol{\beta}$ are used to set the individual importance of each part in each mixture component. In addition, the proposed energy function allows for parts to be shared across multiple object classes.

Both proposals result in a more efficient use of training data as it is shared between all mixture components. Empirically we have demonstrated the applicability of part sharing in the DPM by running a large set of experiments in Section 3.6.

Part sharing improves AP for most object classes and mean AP across all classes increases from 14.6 to 16.5 – see Table 2.2 and Figure 3.8. One positive effect of part sharing is that we can often reach better performance with models using half the number of parts compared to models using no part sharing – see Figure 3.9. In this sense part sharing indeed allows us to do more with less.

Qualitative results (Section 3.6.3) show the effects of part sharing. The proposed detectors are able to recognize object parts across different viewpoints as validated by Figures 3.17, 3.18 and 3.19.

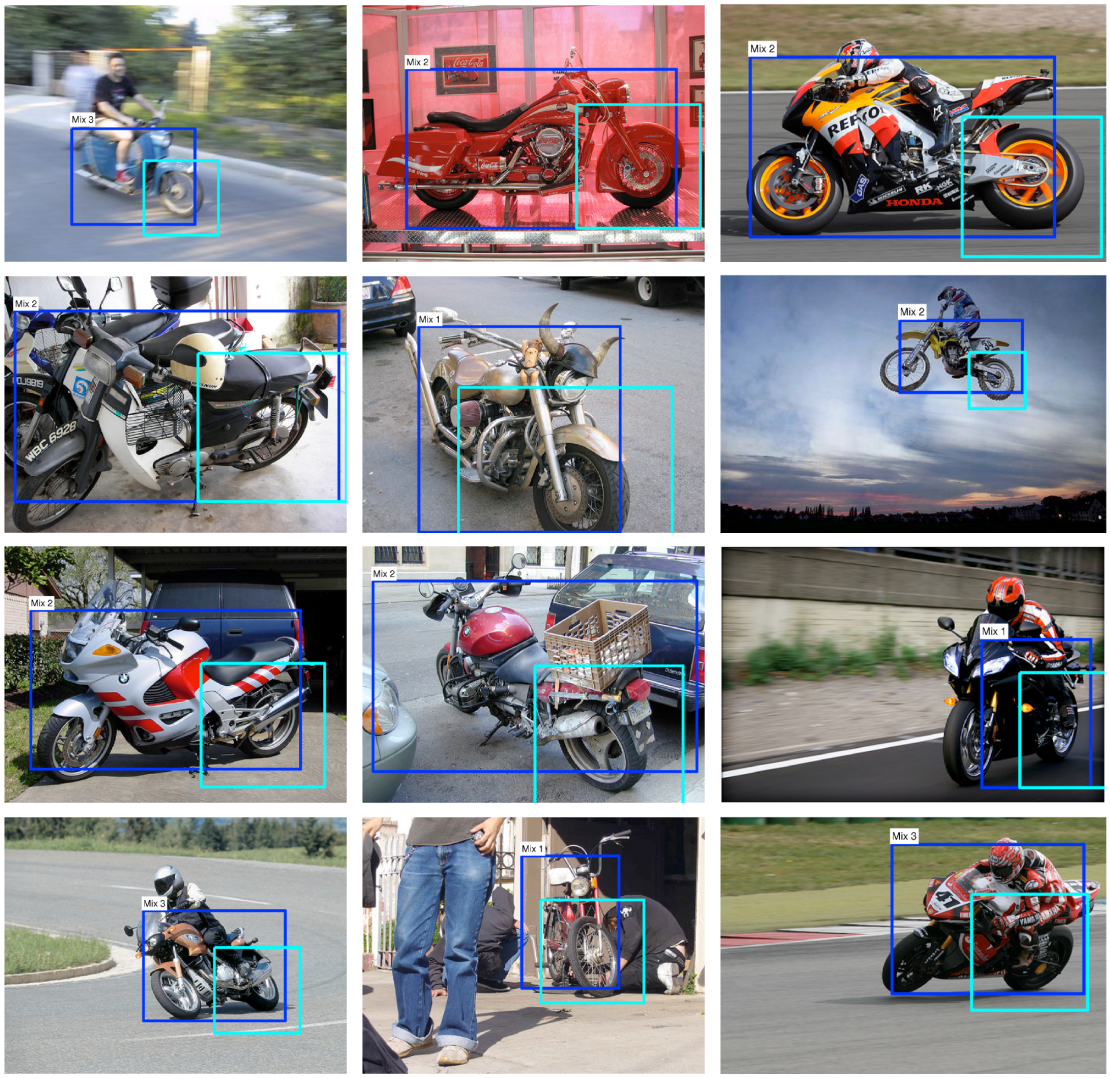


Figure 3.18: Part locations for 'motorbike' detections. We plot the location of a specific part in multiple detections of the 'motorbike' class. The part corresponds to the right wheel of a motorbike (dependent on viewpoint). We are able to use the part in different mixture components and it is reliably recognized across different mixture components.



Figure 3.19: Part locations for detections of the 'horse' class. We plot the location of a specific part in multiple detections. The part corresponds to the neck/head portion of a horse. As in previous examples allows the part to be used across different mixture components.

In summary, part sharing for DPMs offers possibilities to create better performing models with fewer parameters *c.f.* models without part sharing while utilizing the available training data more efficiently.

CHAPTER 4

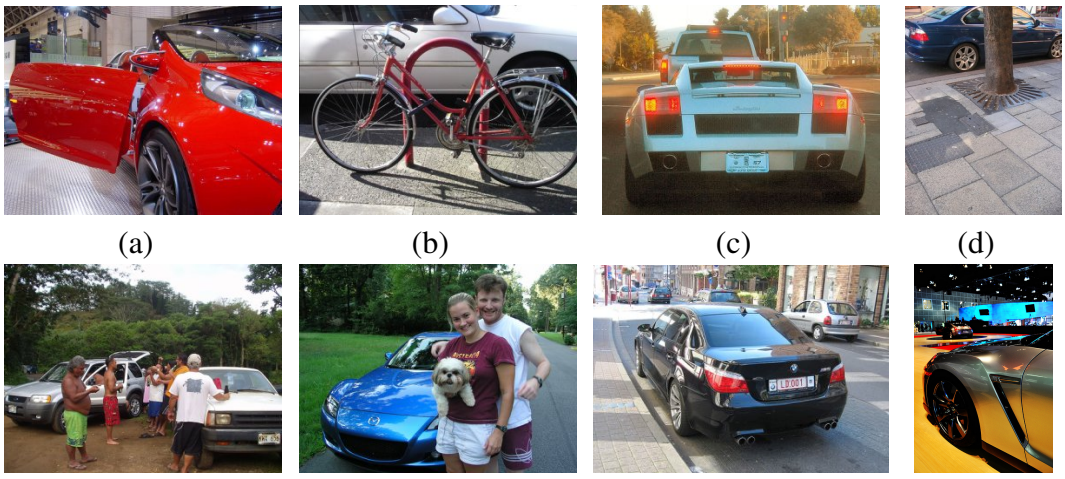
DETECTION OF PARTIALLY VISIBLE OBJECTS

4.1 Introduction

One aspect of object appearance which has thus far been largely overlooked, with a detrimental effect on detection results, is *partial visibility*. State-of-the-art methods [38, 134] generally assume that the entirety of an object instance is visible or at most offer a way to model truncation of an object by the image boundary [53, 135]. Moreover these approaches predict a bounding box for a partially visible object which 'hallucinates' the hidden parts of the object.

We investigate a model for detection which explicitly accounts for partially visible objects. Partial visibility of objects in a scene is commonplace, for example as shown in Figure 4.1 cars are routinely occluded by other objects such as other cars (c, g), people (e, f) or smaller objects such as trees (d), or may lie partially outside the image (a, b, h).

Current object detection methods typically neglect the possibility of partial visibility of an object instance, hoping that the feature representation of a partially visible object is sufficiently unaffected that the object/non-object classifier function will still exceed



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.1: Examples of partial visibility of the car class due to occlusion or truncation. Partial visibility in natural images in commonplace. For example, cars can be occluded by other cars (c, g), other objects (b, d, e, f) or truncated by the image boundary (a, b, h).

threshold *i.e.* the detector still 'fires'. While this is a reasonable assumption for small occlusions, for example a tree in front of a car, it is clearly unrealistic when a larger part of the object is hidden, see *e.g.* Figure 4.1 (b, c, f) and (h).

In practice, trained object detectors usually place high weight on a few key regions of the object that are characteristic of the object category, *e.g.* wheels for a car. When these regions are occluded (*e.g.* Figure 4.1 (f)), the corresponding positive contribution to the 'detection score' is lost, and given sufficient missing regions, the image is misclassified as 'non-object'. The problem is compounded by the use of discriminative classifiers, since the occluders will themselves typically contribute *negative* evidence to the presence of the object to be detected.

4.1.1 Chapter Contributions

In this chapter three key contributions are made: (i) we propose a method which explicitly accounts for partial visibility at both training and test time, by modeling the visibility as a *latent variable*. (ii) The proposed method gives a more accurate interpretation of the image scene by not just reporting the localization of an object but also by inferring which parts of the object are visible and which are not. (iii) We show that the modeling of partial visibility results in a performance improvement in object detection. Figure 4.2 provides an overview of the proposed framework.

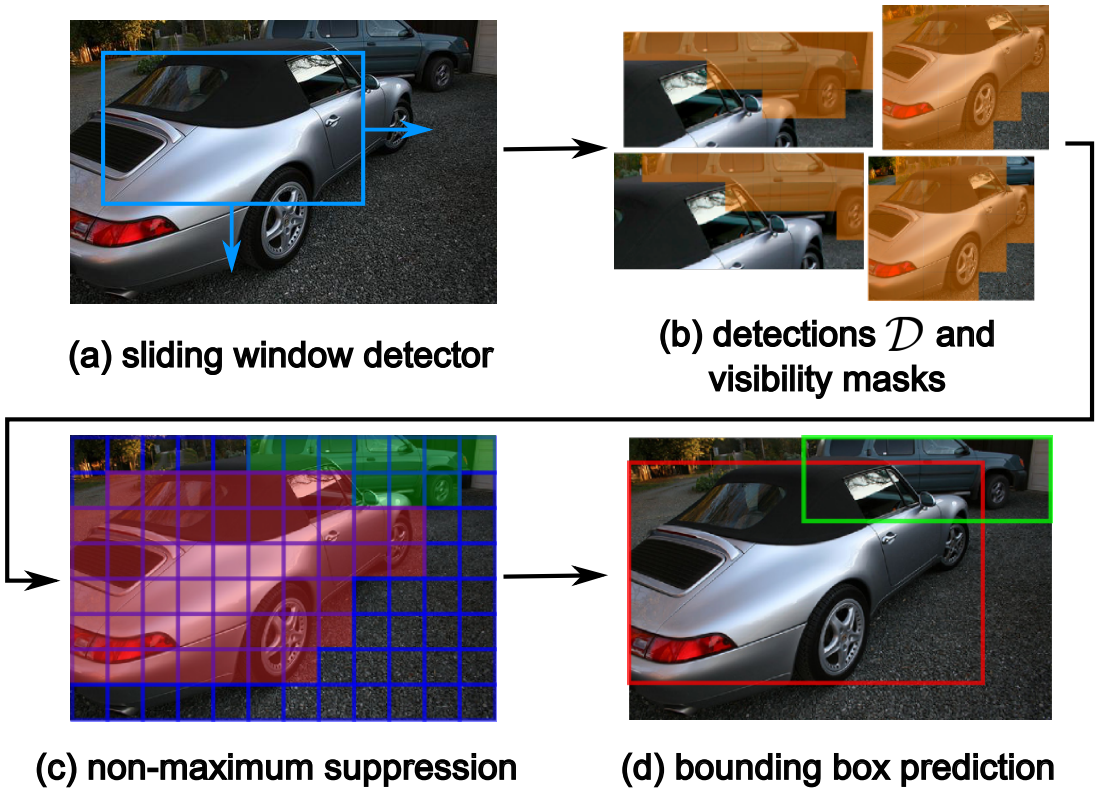


Figure 4.2: Pipeline of proposed detection framework: A sliding-window detector (a) generates candidate detections (b) each comprising a detection score and a block-wise mask of visibility inferred using an MRF model. We show two groups of closely located detections, which are commonly output by a sliding window detector. Non-maximum suppression uses the additional cues of partial visibility for each window to infer a scene interpretation (c), from which accurate object bounding boxes (d) are predicted. Different to common object detection schemes, our method is able to report on the visible extent of each object instance (c).

As in Chapter 2 the proposed method is based on a discriminative sliding window approach (Figure 4.2 (a)), taking advantage of a holistic description of the object, but *explicitly* modeling the possibility of partial visibility. For a given window we treat partial visibility as a latent variable which specifies which regions of the hypothesized object are visible or hidden. A prior over plausible values of the variables is defined in the form of an homogeneous Markov Random Field (MRF), capturing two properties: (i) the effect of partial visibility of an object class on the classification score if a region of the window, *e.g.* a wheel of car, is considered occluded; (ii) spatial continuity of partial visibility patters within a window – neighboring regions are often occluded together.

At test time we infer partial visibility while simultaneously computing a score representing the confidence of the object class being present – Figure 4.2 (b) shows some example detections with inferred visibility. Similar to other object detectors [23, 38], the proposed method gives multiple detections at the true location of an object instance. Consequently, we propose a non-maximum suppression (NMS) scheme which makes use of the pattern of visibility inferred for each window (Figure 4.2 (c)). In this way our approach can predict the correct visible extent of an object taking into account hidden parts, and correctly resolve detections of neighboring object instances. Such object instances can be discarded by conventional NMS schemes, which often consider the overlap of the bounding boxes of two object instances as a measure describing their spatial similarity [38]. Common object detection methods however assume objects to have a fixed 'average' extent and predict a bounding box which 'hallucinates' the visible extent of an object. In contrast, our proposed method results in an accurate bounding box for partially visible objects – see Figure 4.2 (d).

Chapter Outline

Related work in the area of partial visibility modeling is introduced in Section 4.2. We describe the proposed methods in Sections 4.3, 4.4 and 4.5. Section 4.6 presents qualitative and quantitative results on the PASCAL VOC 2010 dataset [30]. Conclusions are offered in Section 4.7.

4.2 Related Work

Most state-of-the-art approaches to object detection are based on a sliding window framework [138, 23, 38, 134] – see Section 1.3 for an introduction. Two methods using this approach, which stand out for their success on recent PASCAL VOC challenges [32, 30]

include that of Felzenszwalb *et al*. [38] and Vedaldi *et al*. [134]. The deformable part-based model of Felzenszwalb *et al*. [38] combines global window descriptors with a set of moveable parts embedded in a mixtures-of-experts framework [63] and has been discussed in Chapter 3.

The detection cascade of Vedaldi *et al*. [134] applies a sequence of descriptors and classifiers of increasing complexity to a window in a cascade manner [138]. This setup couples fast execution at test time (due to the cascade architecture) with state-of-the-art detection accuracy due to the use of non-linear classifiers.

While both of these methods do offer state-of-the-art detection performance, neither explicitly considers the partial visibility of an object instance *e.g*. arising from intra and inter class occlusion in an image.

4.2.1 Approaches to Occlusion Modeling

Rather little work has explicitly considered the problem of partial visibility in general object detection. Wang *et al*. [140] compute a patch-wise response map from a global detector by independently evaluating a window-based linear classifier for each patch. The classifier is learned considering the entire window while the bias term of the classifier is independently distributed to each image patch. Given the patch-wise responses, Wang *et al*. then use the mean shift algorithm [20] to infer occluded regions in a window exploiting the fact that patch-wise responses in occluded regions are generally low (*c.f*. visible regions) and contiguous. In the case that the resulting "segmentation map" is inconsistent, *i.e*. there are occluded regions, individually learned part-detectors are invoked and the maximum over part scores and global classifier score serves as the window's confidence.

All part-based approaches discussed earlier in Section 3.2 have some potential for tolerance to occlusion which greatly depends on how part scores are combined in the models. Voting-based approaches, *e.g*. Implicit Shape Models [79], and generative part-based approaches [44, 123, 146] are potentially quite robust to occlusion given that the lack of image evidence in occluded regions will not significantly upset the overall detection scheme. Other methods, such as the deformable part-based models (DPMs) [38] or methods building on conditional random fields [142, 114], can suffer from their discriminative learning capabilities as previously discussed: occluders often contribute negative evidence to the presence of an object. Some part-based models [41, 71, 142], however, explicitly consider the possibility of occlusion of an object class.

The "constellation of parts"-model of Fergus *et al*. [41], which was earlier discussed

in Section 3.2.5 defines a joint probability density over shape, appearance, scale *and* part-level occlusion. The model assigns part labels to a collection of affine image regions [67] by marginalizing over all possible combinations of such assignments. The method can consider individual object parts as occluded and uses a probability table to model all possible patterns of occlusion given a configuration of parts. The table is treated as a model parameter and inferred during training.

Kushal *et al*. [71] represent objects as groups of features, which correspond to larger parts of an object *e.g*. the back of a car. The representation of the object parts is locally adaptive, *i.e*. parts can adapt to the image content according to a learned geometric configuration. At test time parts are assigned to object instances by optimizing a joint probability density which considers the number of objects in the scene, the appearance and geometric configuration as well as the probability of occlusion of each part. The occlusion probability is treated as an independent variable and statistically estimated from a validation dataset.

Winn *et al*. propose the 'layout consistent random field' (LayoutCRF) model [142], which has already been mentioned in the previous chapter. The model combines local part detectors with a conditional random field [73] of part configuration, placing pairwise constraints on neighboring parts *e.g*. the middle of a car should be next to the front. The method offers an elegant formulation for patch-wise scene labeling accounting for the possibility of intra as well inter-class occlusion in the pairwise terms of the CRF. In their model, the cost of two neighboring image patches belonging to two different object instances, *i.e*. one occludes the other, is higher than two neighboring patches being assigned to the same object or background, forcing the method only to consider occlusion if significant support from individually evaluated image patches is available. An additional contrast term in the pairwise potentials encourages object edges to align with the image boundaries. As mentioned in the previous chapter, the LayoutCRF is limited to single-scale objects due to an expensive inference procedure and relies on somewhat weak part detectors. In addition, in their experimental evaluation Winn *et al*. [142] have only considered the possibility of intra-class occlusion. In contrast, the method presented in this chapter adopts a holistic object representation, detects objects at multiple scales, offers significantly faster inference *c.f*. the LayoutCRF and we successfully model inter as well as intra-class occlusion.

Some of the presented approaches [142, 79, 71] share a common representation, which is to represent an object by a set of independently trained part (or patch) detectors, building inference of partial visibility on top of the part responses. This assumes that parts can be reliably detected. However, in practice it often proves difficult to build reliable part



Figure 4.3: Patterns of occlusion considered by Lin *et al*. [81], who train specialized boosted classifiers for each pattern in addition to a boosted classifier which considers no occlusion. Figure reproduced with permission from Lin *et al*. [81].

detectors – there is too little salient visual information available on a local feature level. In contrast, the method proposed in this chapter tackles detection at a more global level, taking advantage of all of the visible object regions.

Girshick *et al*. [54] proposed grammar models for object detection, which are DPMs with a tree-hierarchy of parts. The approach has already been discussed in Section 3.2.1 of this thesis. Here we specifically focus on the ability of the method to compensate for the occluded parts of an object class. The model uses the mixture components of the DPM to represent various degrees of occlusion of the 'person' class of the PASCAL VOC 2010 dataset [30]. In addition, the method employs an occluder part in the DPM, which models the 'stuff' that occludes object instances of the 'person' class and must be placed in case not all the parts of the regular model are placed. The grammar model is hand-crafted, *i.e*. the degree of occlusion in each mixture component is defined by hand. This stands in contrast to the method presented in this chapter which learns the degree of occlusion for an object class and has the ability to infer many different occlusion patterns.

4.2.2 Face Detection

In the related area of face detection a number of authors have investigated models of occlusion, but the applicability of these approaches to general object detection has not been established. Williams *et al*. [141] use a variational Ising classifier to model contamination (such as occlusion) by a binary mask, and extend a common kernel classifier for 'clean' data to one that can tolerate contamination. Similar to the approach presented in this chapter, Williams *et al*. [141] define an MRF prior over plausible occlusion patterns. To compute the confidence whether an image depicts a face or not, Williams *et al*. marginalize over all possible occlusion patterns. In contrast to our method, which considers occlusion at training and test time, the method proposed by Williams *et al*. works as an extension to an already trained face detector, which is not aware of occlusion.

Lin *et al*. [81] adapt a boosted cascade for face detection to handle occlusion by 'hard-

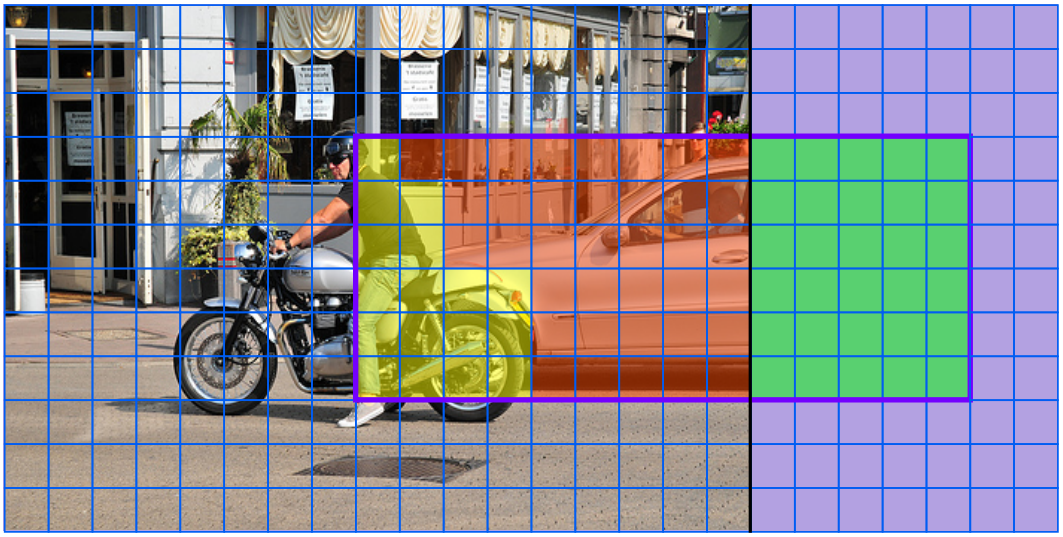


Figure 4.4: An example detection. Our approach is motivated by the method of Vedaldi & Zisserman [135] and Girshick *et al*. [53] who pad the image representation with additional blocks (purple area) to detect truncated object instances. During classification Vedaldi & Zisserman [135] 'count' the number of feature blocks that are outside an image (green area) and learn a classifier weight for this number of blocks. Instead of learning a global weight, Girshick *et al*. learn individual bias terms for each block. In case of truncation of a block, these bias terms are added to the classification score of a window. We adopt the approach of Vedaldi & Zisserman [135] but in addition to modeling truncation by the image boundary, our proposed method also infers the visible parts of an object inside the image (orange region) as well as the occluded parts of the object instance (yellow region).

coding' eight different types of partial visibility and testing for them in addition to testing for fully visible faces. The method first trains a boosted cascade using Haar-like features [138], assuming no occlusion. In a second step, a separate cascade for each of the eight occlusion types illustrated in Figure 4.3 is trained, only considering Haar-like features which lie inside the visible region of the respective occlusion pattern. At test time an example is first evaluated by the cascade which assumes no occlusion and, if it is rejected, it is furthermore analyzed by the eight specialized cascades. In the case that multiple cascades return a detection, the one with the largest score is chosen. The method of Lin *et al*. does not generalize beyond the patterns shown in Figure 4.3. This is in contrast to our method presented in this chapter, which *learns* the patterns of occlusion.

4.2.3 Motivation

Our approach is related to the work of Vedaldi & Zisserman [135] and Girshick *et al*. [53]. Both methods propose part-based models for object detection and have been described in detail in Section 3.2.1. The models account for alignment, multiple aspects and additionally for truncation by the image boundary of an object class. In the following we concentrate only on their ability to model truncation by the image boundary. Figure 4.4 provides an overview of the approaches and a comparison to the method presented in this chapter.

Both methods pad the image representation with additional blocks (purple area in Figure 4.4) to detect truncated objects. Vedaldi & Zisserman [135] 'count' the number of feature blocks that are outside an image and learn a classifier weight for this number of blocks. This is visualized in Figure 4.4 – green blocks indicate patches outside the image, which are covered by the window. Given a sufficient amount of occluded blocks, this weight can compensate for the missing responses of these blocks. Girshick *et al*. represent truncation as a block-wise mask defined by the overlap of a bounding box with the image boundary [53]. In case of occlusion of a block, individually learned bias terms are added to the window score to compensate for the missing block responses.

We have found that for the sliding window detectors used in this chapter both methods give comparable results, while the method of Vedaldi & Zisserman [135] requires the learning of fewer parameters (one additional variable over a regular sliding window detector). We therefore adopt the approach of Vedaldi & Zisserman [135] in this chapter, but in addition to modeling truncation by the image boundary our method also models partial visibility inside the image by treating it as a latent variable. This is illustrated in Figure 4.4 in which yellow blocks indicate the occluded parts of an object and orange blocks indicate the visible parts.

Very recently Gao *et al*. [50] presented a method which is closely related to the one in this chapter. The authors follow a similar approach to modeling partial visibility by treating visibility as a latent variable and defining a smoothness prior over possible occlusion patterns. However, different to our approach, Gao *et al*. [50] assume training data to be labeled with annotation regarding the visibility of each object instance. Another difference to the method proposed in this chapter is that their method uses a structured learning framework [129] to drive learning of the detectors, while our method uses the 'latent SVM' [38, 3] described in Chapter 3. Gao *et al*. employ a loss function for structured learning which is based on the overlap of the predicted bounding box with the ground-truth bounding box. However, different from the novel NMS scheme presented in

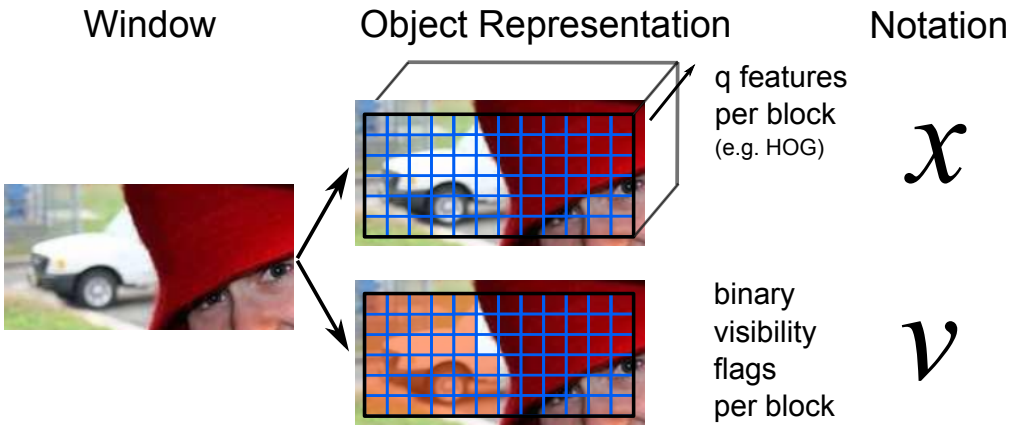


Figure 4.5: Modeling of partial visibility. A window is represented by a block structure, consisting of (i) a q-dimensional feature descriptor per block, concatenated into the feature descriptor for the entire window $\boldsymbol{x}$; (ii) a vector $\boldsymbol{v}$ of binary visibility flags, with one flag per block. The inferred visibility flags – visible blocks are shaded orange – control which of the corresponding features are used by the classifier, and are subject to a prior defining plausible patterns of visibility.

Section 4.5, this overlap measure does not take the partial visibility of an object instance into account.

4.3 Modeling Partial Visibility

We recall that sliding window detectors generally move a window of fixed size over an image (Figure 4.2 (a)). For each window, a feature vector is extracted and a classification function is applied to classify the window as object/non-object. As shown in Figure 4.5 we use features which can be organized in a structure of blocks. We model partial visibility at the level of blocks *i.e.* inference in our model yields a per-window labeling of which blocks of the object are visible or hidden. Throughout this chapter the features of the ith block are denoted $\boldsymbol{B}^i$ and the features of the entire window $\boldsymbol{x} = \langle \boldsymbol{B}^1, ..., \boldsymbol{B}^r \rangle$ for r blocks.

4.3.1 Partial Visibility as a Latent Variable

To represent partial visibility we append visibility flags $\boldsymbol{v}$ to the conventional feature representation $\boldsymbol{x}$ – see Figure 4.5. If $v_i = 0$, we consider the ith block to be not visible (occluded), if $v_i = 1$ we consider it visible.

The key to the method proposed in this chapter is to treat the visibility flags $\boldsymbol{v}$ as a latent variable to be inferred during classification of a window, allowing the classifier to operate on only the visible portion of the object, and enabling subsequent NMS processing to predict accurate bounding boxes for partially visible objects.

Furthermore, we apply inference to *training* examples to account for partially visible examples in the training data. This is closely related to the approach in Chapter 3, in which we have treated part placements as a latent variable at training and test time. Applying inference to the training examples allows our approach to operate without additional ground-truth annotation regarding the visibility of an object.

Figure 4.4 shows an example detection with inferred visibility flags: yellow blocks correspond to hidden parts of the object while orange blocks correspond to visible parts. We pad the image with additional blocks on each side to enable the detection of truncated objects. These blocks are always considered hidden ($v_i = 0$) and are shown in green in Figure 4.4.

Unary Bias

As noted in Section 4.1, the features of an occluded block typically contribute negatively to the classification score of a window because the occluder often resembles what the object class is supposed to *not* look like. Introducing visibility flags allows our method to correct this by inferring which blocks are in fact not visible and 'disabling' the part of the classifier operating on the corresponding features. This is achieved by replacing the partial classifier score for the block with a unary bias u, which compensates for the missing image evidence [135] in the case that the a block is inferred to be not visible ($v_i = 0$). This bias term is *learned* as part of the training process.

MRF Prior

Partial visibility of natural objects tends to be contiguous *e.g.* a car is often occluded by another car, resulting in a large and connected area of invisible blocks, whereas complex or sparse patterns of partial visibility are not common. To model this notion of contiguity we define an MRF prior over $\boldsymbol{v}$ in the form of an Ising model [84] as shown in Figure 4.6. The Ising model discourages assigning differing visible/invisible labels to neighboring blocks.

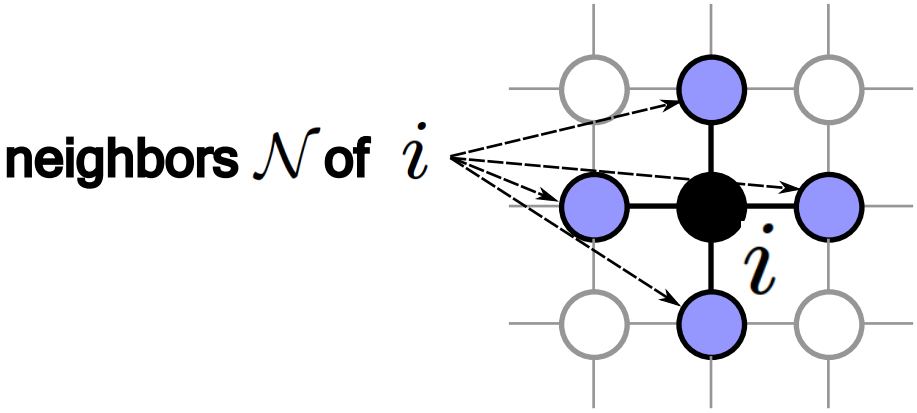


Figure 4.6: The imposed Ising model used for partial visibility modeling. To model the notion of contiguity in the visibility flags we define an MRF prior over $\boldsymbol{v}$ in the form of an Ising model [84]. We use a standard 4-neighborhood.

4.3.2 Window Responses

Given a block representation of a window with features $\boldsymbol{x}$, we define the task of assigning a classification (object present) confidence to the window as maximization of a joint *window response* function $g(\boldsymbol{x})$ with respect to the visibility flags $\boldsymbol{v}$ which define which blocks in the window are considered visible/hidden. The response function comprises (i) a classification function and (ii) a prior over visibility flags:

$$g\left(\boldsymbol{x}\right) = \max_{\boldsymbol{v}} \left[f\left(\boldsymbol{x}, \boldsymbol{v}\right) - \alpha \sum_{(i,j)\in\mathcal{N}} \rho\left(v_i, v_j\right) \right] \tag{4.1}$$

where the first term $f(\boldsymbol{x}, \boldsymbol{v})$ defines a 'classification score' for the window *given* the inferred pattern of visibility $\boldsymbol{v}$. We adopt a linear classification function, and later demonstrate the extension to a mixture of linear classifiers [38]:

$$f\left(\boldsymbol{x}, \boldsymbol{v}\right) = b + \sum_{i=1}^{r} v_i \boldsymbol{w}^i \cdot \boldsymbol{B}^i + \left(1 - v_i\right) u \tag{4.2}$$

The visibility flags v_i act as a selector/switch function – if the visibility flag for the ith block v_i is switched on, $\boldsymbol{w}^i \cdot \boldsymbol{B}^i$ is added to the score, where $\boldsymbol{B}^i$ represents the features of the ith block and $\boldsymbol{w}^i$ the learned appearance of that block. If v_i is switched off the *learned*

unary bias u is added to the score. b is the bias term of the linear classification function.

The second term of the window response (Equation 4.1) $\rho(v_i, v_j)$ acts as a penalty by defining an Ising prior over the field of visibility flags. A penalty of 1 is imposed if $v_i \neq v_j$ and 0 otherwise for all pairs of neighboring blocks $\mathcal{N}$. We use a conventional 4-neighborhood – see also Figure 4.6. The variable α defines the relative weight of the classification score and contiguity terms. If α is small, sparse patterns of visibility flags are allowed while for large values of α contiguous patterns are encouraged. As noted, this model of visibility can be interpreted as an MRF at the level of a single window, with nodes corresponding to blocks. The pairwise terms are homogeneous in that we assume occlusions have the same contiguity properties regardless of their location.

4.3.3 Inference

The maximization in Equation 4.1 can be solved efficiently using the well-known Graph-Cut approach [15, 14], since the problem corresponds to a standard binary-valued MRF, with sub-modular energy for positive α [70]. However, despite the efficiency of this method, an image comprises a large number of windows (> 10.000) such that solving a Graph-Cut problem for each window is still somewhat onerous. We therefore speed up the detection process by filtering each window using the following upper bound on $g(\boldsymbol{x})$:

$$\hat{g}(\boldsymbol{x}) = \max_{\boldsymbol{v}} f(\boldsymbol{x}, \boldsymbol{v}) \geq g(\boldsymbol{x}) \tag{4.3}$$

This upper bound is trivially derived from Equation 4.1 by removing the binary terms $\rho(v_i, v_j)$ since these are always nonnegative. Maximization of the upper bound is achieved in a simple block-wise fashion *i.e.* the values for v_i can be computed independently for each block simply by comparing the classifier term for the block to the bias term. If $\hat{g}(\boldsymbol{x})$ scores above a pre-defined threshold t, *e.g.* $t = -1$, we compute $g(\boldsymbol{x})$ by solving the Graph-Cut problem [14].

4.3.4 Mixture Model

To cope with different modes of appearance of an object class, for example viewpoints, we follow the approach of Chapter 3 and use a set of window classifiers in the manner of a mixture model [63].

For each mixture component, or cluster of appearance, a classifier is trained using

the corresponding training examples which are mutually exclusive to other mixture components. The final classification confidence $h(\boldsymbol{x})$ for a window with features $\boldsymbol{x}$ is then computed as the maximum over all the component classifiers $g^i(\boldsymbol{x})$:

$$h(\boldsymbol{x}) = \max_{i=1...d} g^i(\boldsymbol{x}) \tag{4.4}$$

where d is the number of mixture components. As before in this thesis we refer to function $h(\cdot)$ as the detector and to the functions $g^i(\cdot)$ as the mixture components. The particular mixture component m^* satisfying the above equation, i.e. $m^* = \arg\max_m \{g^m(\boldsymbol{x})\}$, is referred to as the *mixture assignment* of the example with features $\boldsymbol{x}$. We use individual unary bias terms for each mixture component. This is necessary because different mixture components might base their decision on a different set of key regions, requiring individual values of u to compensate for the occlusion of such regions.

4.4 Learning

Similar to the models of Chapter 3, we cast the learning problem as an energy minimization problem and adopt the optimization scheme proposed in Section 3.5.2. In the following we will introduce the necessary notation and define an energy function for model training in Section 4.4.1.

Notation

As in previous chapters a training dataset $\mathcal{T} = \{(\boldsymbol{x}^1, y^1), ..., (\boldsymbol{x}^n, y^n)\}$ is given where $y^k \in \{+1, -1\}$ is the class of the window (object/non-object) and $\boldsymbol{x}^k$ is the feature vector of the kth training example. Each training example is assigned latent variables which are the mixture assignment and visibility flags. We represent the latent mixture assignments by $\mathcal{M} = \{m_1, ..., m_n\}$ and the visibility flags of each example by $\mathcal{V} = \{\boldsymbol{v}^1, ..., \boldsymbol{v}^n\}$. They are combined into the set of latent variables $\mathcal{L} = \{\mathcal{M}, \mathcal{V}\}$. We recall that the mixture components of $h(\cdot)$ are essentially defined by the weight vector $\boldsymbol{w}$, modeling the appearance of an object class, and the unary bias term u, which replaces the classification score for a block in the case that it is not visible. Each mixture component is assigned its own unary bias term. Notation-wise we represent these as $\mathcal{W} = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^d\}$ and $\mathcal{U} = \{u^1, ..., u^d\}$ respectively. They are combined into the set of model variables $\mathcal{O} = \{\mathcal{W}, \mathcal{U}\}$.

We now present a learning scheme for estimating these model variables, while also inferring the latent visibility flags and mixture assignments for each training example. The learning scheme is motivated by the 'latent SVM' learning scheme used for part-based modeling in Chapter 3.

4.4.1 Energy Function

Similar to the previous chapter the energy function is dependent on the model variables $\mathcal{O} = \{\mathcal{W}, \mathcal{U}\}$ and assumes a training dataset $\mathcal{T}$ as given. It consists of two parts: (i) a regularization term, ensuring good generalization on unseen data and (ii) a loss term, determining how well the detector predicts the training labels:

$$E\left(\mathcal{O}|\,\mathcal{T}\right) = \frac{\lambda}{2}\sum_{j=1}^{d}\left[\left\|\boldsymbol{w}^j\right\|^2 + c_j\left(u^j\right)^2\right] + \sum_{k=1}^{n} L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) \tag{4.5}$$

where $\|\cdot\|$ is the ℓ_2-norm. λ sets the relative importance of the regularization term to the loss term. It defines a tradeoff between generalization (low norm of model variables)in and correctly classifying the training labels (small loss term). We found only modest performance increases by introducing separate regularization for $\boldsymbol{w}^j$ and u^j. c_j is the number of blocks in the jth mixture component. The multiplication with u^j ensures that the unary bias of each mixture component is appropriately weighted in the regularization term.

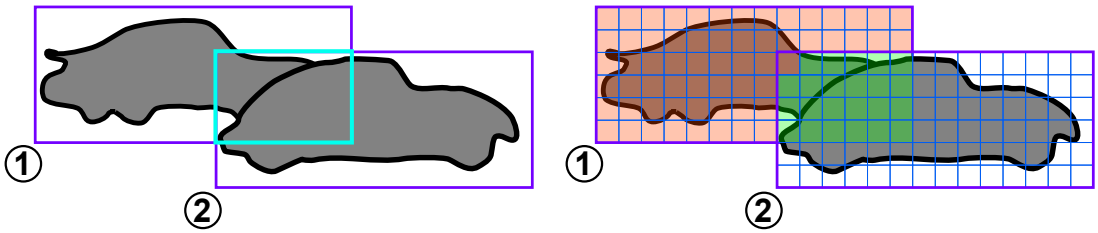
The loss function $L\left(\cdot\right)$ determines how deviations of the predictions of $h\left(\boldsymbol{x}^k\right)$ from the true target value y^k should be penalized. As in the previous chapter the hinge loss is used:

$$L\left(y^k, h\left(\boldsymbol{x}^k\right)\right) = \max\left\{0, 1 - y^k h\left(\boldsymbol{x}^k\right)\right\} \tag{4.6}$$

4.4.2 Optimization

We would like to minimize the energy function (Equation 4.5), *i.e.* reducing the accumulated losses of the second term while also ensuring low ℓ_2-norms of all weight vectors $\mathcal{W}$ and unary bias terms $\mathcal{U}$:

$$\min_{\mathcal{O}} E\left(\mathcal{O}|\,\mathcal{T}\right) = \min_{\mathcal{W},\mathcal{U}} E\left(\mathcal{W},\mathcal{U}|\,\mathcal{T}\right) \tag{4.7}$$



(a) Regular NMS (b) Occlusion-aware NMS

Figure 4.7: Occlusion-aware NMS *c.f.* regular NMS. Two instances of the 'car' object class are shown in (a) and (b), one car is occluding the other. A regular NMS scheme *e.g.* [38] (a) is not aware of this occlusion and considers the bounding box overlap as an overlap measure between the two detections. The regular NMS scheme would commonly remove the lower ranked detection and consequently 'lose' a true-positive. (b) The NMS scheme proposed in this chapter is aware of partial visibility (*e.g.* occlusion). For the sake of clarity we do not draw the visibility flags of detection 2 in (b) but consider it completely visible. The visibility flags of detection 1 are correctly inferred and can be used to define a new overlap measure between these detections, which takes occlusion into account. This section will introduce such an overlap measure and furthermore provide an optimization scheme which effectively uses this overlap measure and is guaranteed to converge to a global optimum.

To optimize this energy the method presented in Section 3.5.2 is used. This method alternates between updating latent variables $\mathcal{L}$ (outer loop) and optimizing model variables $\mathcal{O}$ while holding $\mathcal{L}$ fixed in an inner loop. As before we perform bootstrapping in the inner loop.

4.5 Fusion and Non-Maximum Suppression

As with conventional sliding window detectors, our proposed method defined so far will tend to give multiple 'candidate' detections for a single object due to invariance in the image descriptor. Given a set of such candidates $\mathcal{D}$ (Figure 4.2 (b)) we require a method for fusing detections into one per ground-truth object. This is essential for evaluation on the PASCAL VOC data [32], since the evaluation criterion for a true positive allows only one detection per ground-truth object – all additional detections are considered false positives. As in previous chapters we refer to this task as non-maximum suppression (NMS). The goal of NMS is to infer a subset $\mathcal{I} \subseteq \mathcal{D}$ of final detections, which we refer to as a *scene interpretation*.

The problem we are facing with NMS in the light of partially visible objects is that common methods for NMS usually assume the entirety of an object to be visible and

hence partially visible object instances can be suppressed and create an unfitting scene interpretation of an image. Figure 4.7 illustrates this problem and outlines the solution we present in this section. Rather than assuming full visibility of each object instance our approach makes use of the inferred visibility information, which is provided by the methods presented in the previous sections. We enable NMS which is aware of partial visibility by defining a novel overlap measure in Section 4.5.3 and an energy maximization framework to infer fitting scene interpretations in Section 4.5.4.

4.5.1 Previous Work

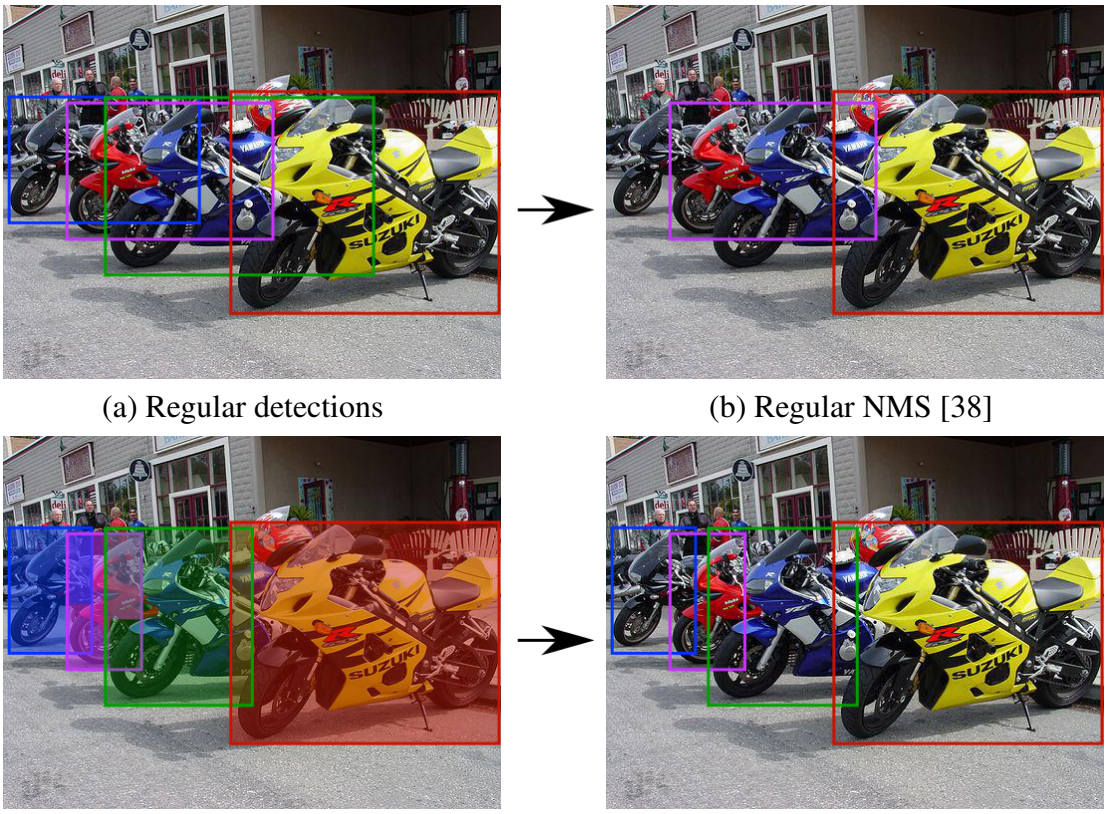
NMS in sliding window detectors has generally been treated as a process separate from the rest of the detection framework. Felzenszwalb *et al*. [38] predict bounding boxes using a linear regressor based on the positions of object parts in their framework and then employ a greedy selection scheme for NMS. This scheme, which has been used throughout the thesis so far, iteratively selects the highest ranking detection and removes all detections that overlap to a certain degree η.

We note that the overlap measure of Felzenszwalb *et al*. [38] is not the one used by the PASCAL VOC challenges [32] (see Equation 1.1). Felzenszwalb *et al*. [38] remove a detection if the division of the area of intersection of two bounding boxes by the area of the lower ranked bounding box exceeds some threshold η. That is, given two detections with bounding boxes B_l, B_k where the detection of B_l is the selected higher ranked detection, the overlap measure is given by

$$ov\left(B_l, B_k\right) = \frac{area\left(B_l \cap B_k\right)}{area\left(B_k\right)} \tag{4.8}$$

This overlap measure has the main effect that lower ranked detections which are fully contained within the highest ranked detection are removed as they always overlap 100% given the above measure of overlap.

Another popular approach to NMS includes mean-shift techniques [20], which treat detections as points in scale-space and use an iterative procedure to converge on modes of the 'distribution' of detections, *e.g*. Leibe & Schiele [80] and Dalal [22]. The use of mean-shift for NMS is not very principled, since it assumes a probability density on which's modes to converge but the scale-space can not directly be considered as a probability density. In addition, mean-shift techniques require the tuning of multiple parameters (smoothing parameters, choice of distance function, ...), which can be harmful to detec-



(a) Regular detections (b) Regular NMS [38]

(c) Detections with visibility flags (d) Visibility-aware NMS

Figure 4.8: Comparison of a regular sliding window detector to the proposed detectors with visibility-aware NMS. (a) Bounding boxes of detections of a regular detector 'hallucinate' the true extent of an object. (b) The NMS scheme consequently removes strongly overlapping detections. (c) Our proposed detection scheme provides visibility flags, indicating the visible extent of each detection. Bounding boxes are adapted to the visibility flags. (d) The proposed NMS scheme takes the visible extent of each object instance into account and does not suppress partially visible detections.

tion accuracy. Compared to the greedy selection strategy of Felzenszwalb *et al.* [38] we have found experimentally that the method of Dalal [22] performs slightly worse.

These schemes have in common that they do not acknowledge the possibility of partial visibility of an object class but instead 'hallucinate' the full extent of an object. Although the method of Felzenszwalb *et al.* [38] uses a regressor to predict the extent of a bounding box, it does not make use of cues regarding partial visibility. Figure 4.8 illustrates that this is problematic as the assumption of fully visible objects can cause suppression of partially visible objects. The figure shows a set of detections from a regular sliding window detector, which assumes fully visible objects and hallucinates the visible extent of the motorbikes. The greedy NMS scheme of Felzenszwalb *et al.* [38] consequently

removes strongly-overlapping detections, suppressing potential true-positive detections. In contrast, the detectors proposed in this chapter consider only the visible extent of each object instance as indicated by the visibility flags (Figure 4.8 (c)). Subsequent NMS takes the visible extent of each object instance into account and does not suppress partially visible detections – see Figure 4.8 (d).

In addition, the discussed NMS schemes are not guaranteed to converge to a globally optimal solution: the method of Felzenszwalb *et al*. [38] is greedy by nature while mean-shift methods are generally non-convex optimization problems. The method proposed in this section is guaranteed to converge to a globally optimal solution.

4.5.2 Preparation & Notation

The two key components of the proposed NMS scheme are (i) a novel overlap measure, which accounts for the possibility of partial visibility of an object and (ii) an energy function which offers a way to infer a good scene interpretation alongside an optimizer which is guaranteed to converge to the global optimum. The basic idea of our scheme is to find a set of bounding boxes which cover all detections while taking into account their inferred visibility.

The detection framework described in Section 4.3 outputs detections $\mathcal{D}$ at different scale levels. As a first step we ensure comparability of these detections by interpolating their visibility flags to a common scale level using nearest-neighbor interpolation. This common scale level is represented by a block grid of size $b_x \times b_y$ and we use the smallest scale level available (*i.e*. a scale factor of 1). As shown in Figure 4.9 (b) this grid describes the entire scene *c.f*. a single window. In addition, we normalize the scores of all detections in $\mathcal{D}$ by subtracting the threshold t that was used to obtain the detections. This step ensures that the minimal score will be 0.

To explain our NMS scheme two matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ are defined. The purpose of those matrices is to relate single detections of $\mathcal{D}$ to a scene. Both matrices are of size $b_x b_y \times |\mathcal{D}|$, where $|\mathcal{D}|$ is the number of detections and $b_x b_y$ is the total number of blocks in a scene. Please see Figure 4.9 for a visual explanation of the two matrices.

Elements in $\boldsymbol{H}$ describe whether a specific block of the scene is covered by a detection $d_j \in \mathcal{D}$. Element $H_{ij} = 1$ if the ith feature block *of the scene* is visible in the jth detection, *i.e*. the corresponding value in $\boldsymbol{v}^j \in \mathcal{V}$ is set to 1.

The second matrix $\boldsymbol{G}$ projects the individual detection scores (Equation 4.1) to the block grid of the scene – see Figure 4.9 (b). Elements in $\boldsymbol{G}$ corresponding to *all* blocks

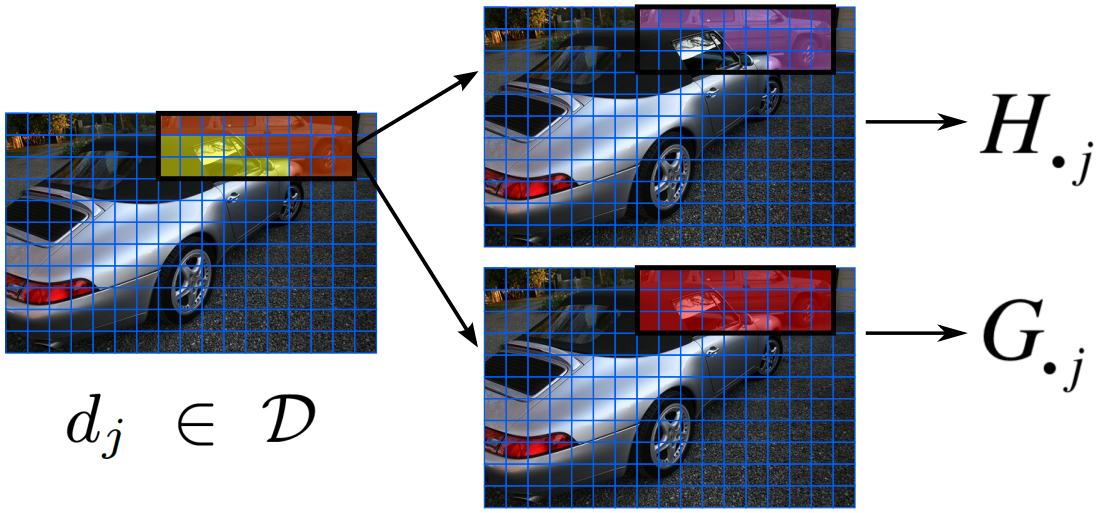


Figure 4.9: Matrices for the non-maximum suppression scheme. (a) a candidate detection $d_j \in \mathcal{D}$ (black bounding box) with inferred visible (orange) and occluded (yellow) regions; (b, top) the visible blocks of d_j are projected to the block grid of the scene. All purple blocks are set to 1, others to 0; (b, bottom) The score of d_j is projected to all blocks of the scene that are covered by the *entire* detection. Each red block is set to the score of d_j, others to 0. (c) The block grids of (b) are rasterized into the rows of matrices $\boldsymbol{G}$ and $\boldsymbol{H}$.

belonging to the ith detection – visible or not – are assigned the score (Equation 4.1) of that detection. The choice to set *all* blocks of $\boldsymbol{G}$ belonging to a detection to the score of this detection is an empirical one: only using visible blocks results in a NMS scheme with a lower detection accuracy *c.f.* using all blocks.

4.5.3 Overlap Measure

Let us define an overlap measure between two detections $d_l, d_k \in \mathcal{D}$ which captures the degree to which the detections overlap in terms of the number of shared *visible* blocks:

$$r\left(d_l, d_k\right) = \frac{\sum_{i=1}^{b_x b_y} H_{ik} \wedge H_{il}}{\sum_{i=1}^{b_x b_y} H_{ik} \vee H_{il}} \tag{4.9}$$

This overlap measure is related to existing methods using bounding box intersection [38] as an indicator of how spatially similar two detections are. However, given the availability of inferred partial visibility flags we consider this overlap measure more informative than simply measuring bounding box intersection because it takes into account the visible portion of the objects and does not hallucinate their full extent.

The overlap measure is also more principled than the one used in existing NMS schemes *e.g.* Felzenszwalb *et al.* [38] in that it considers the true overlap of two detections rather than defining an unprincipled measure (see Equation 4.8) which does not take the actual overlap between two detections, according to the PASCAL VOC criterion, into account (Equation 1.1).

4.5.4 Energy Maximization for NMS

To infer a good scene interpretation $\mathcal{I}$ we would like to maximize the accumulated 'score' of objects in the scene while compensating for multiple detections per ground-truth object. We therefore express NMS as a constrained energy maximization problem. The final and

optimal set of detections $\mathcal{I}^*$ is inferred by:

$$\begin{aligned} & & \mathcal{I}^* &= \arg\max_{\mathcal{I}} F(\mathcal{I}) \\ \text{with} & & F(\mathcal{I}) &= \sum_{i=1}^{b_x b_y} \max_{d_j \in \mathcal{I}} G_{ij} \\ \text{subject to} & & \forall d_k, d_l \in \mathcal{I} &: r(d_k, d_l) \leq \eta \end{aligned} \tag{4.10}$$

This means that for each block on the scene grid the algorithm aims to include the detection that provides the maximum score for that block according to matrix $\boldsymbol{G}$. However, to ensure that the final set of detections is plausible in the context of the scene, a constraint on the maximum amount of overlap between detections η is imposed. This can be considered similar to the overlap constraint imposed by the PASCAL VOC challenge [32] (Equation 1.1), but accounting for partial visibility.

Note that the optimization problem in Equation 4.10 can have multiple solutions. If a lower-ranked detection is fully contained within a higher-ranked detection the energy will be the same no matter whether the lower-ranked detection is considered an element of the scene interpretation $\mathcal{I}$ or not. The optimization scheme presented in the next section resolves this situation by not allowing detections which are fully contained within another detection as an element of the scene interpretation unless they have a higher score.

4.5.5 Optimization by Branch-and-Bound

The search space of the problem described in Equation 4.10 consists of all combinations of detections that satisfy the overlap constraint. It can naturally be thought of as a tree-structure, in which the root of the tree is an empty scene interpretation with no detections assigned to it. The scene interpretations on the first level of the tree consist of exactly one detection from $\mathcal{D}$. The number of different scene interpretations on this level is similar to the number of detections $|\mathcal{D}|$. Each consecutive level of the tree adds one detection of $\mathcal{D}$ to the scene interpretations of the previous level which satisfies the overlap constraint.

We explore this search space using a priority search, which starts with an empty scene interpretation (the root of the tree) having zero energy. We then iteratively pick the highest ranked interpretation and create new scene interpretations by adding one additional detection (satisfying the overlap constraint), recalculating their energy and adding them to the queue of already computed scene interpretations.

If the number of detections $|\mathcal{D}|$ is relatively small, a brute-force search through all

scene interpretations (the entire tree) is possible. However, in the case that the number of detections is large and many candidate detections overlap, a more efficient search procedure is necessary. To that end we propose a novel branch-and-bound procedure. We observe that for any given scene interpretation $\mathcal{I}$ we can compute the following upper bound for the remaining detections in that branch of the search tree, still satisfying the overlap constraint of Equation 4.10:

$$\hat{F}\left(\mathcal{I}\right) = \sum_{i=1}^{b_x b_y} \max_{d_j \in \mathcal{I} \cup \mathcal{C}} G_{ij} \geq F\left(\mathcal{I}\right) \tag{4.11}$$

where $\mathcal{C}$ is the set of detections that satisfies the constraint of Equation 4.10 given the already selected detections $\mathcal{I}$:

$$\mathcal{C} = \left\{ d \middle| \, d \in \mathcal{D} \wedge \forall s \in \mathcal{I} : r\left(d, s\right) \leq \eta \right\} \tag{4.12}$$

This upper bound answers the question of what energy could possibly be achieved if we continue adding detections to this scene interpretation. If it is smaller than the current best energy, we do not need to continue exploring this interpretation and can ignore this entire branch of the search tree. In contrast to other NMS schemes [38, 22] our energy function in combination with the proposed optimization scheme is guaranteed to converge to a global minimum.

Despite the proposed branch-and-bound procedure, optimization is still somewhat slow for a large set of detections $\mathcal{D}$. In our experiments using the proposed NMS scheme we therefore focus on detections which only score above a threshold of $t = -1$ to further prune the search space. The great majority of true-positive detections lies above this threshold while detections bellow this threshold are mainly dominated by false positives.

4.6 Emprical Results

In this section we report the empirical results in the form of a qualitative and quantitative experimental analysis of the proposed method. As in previous chapters, we first discuss the details of our implementation.

4.6.1 Implementation Details

Window Descriptor

We use the Histogram of Oriented Gradients (HOG) descriptor [23] – see Section 2.3 for a detailed explanation. As discussed earlier, the block structure of these features provides a suitable basis for building detectors allowing for partial visibility by inferring visibility at the block level. One layer of HOG features with cells of size 6×6 pixels is used. Blocks are of size 2×2 cells and overlap by one cell.

Parameter Estimation

We use cross-validation to set the parameters $\{\lambda, \alpha, \eta\}$ by learning detectors on the `train` subset of PASCAL VOC 2010 and determining the best parameters on `val` for a detector with $d = 2$ mixture components.

To set the regularization parameter λ of the baseline we learn regular sliding-window detectors without partial visibility modeling and consider possible values of λ in the range of 10^{-1} to 10^{2}. We adopt the value which gives the highest AP at $\eta = 0.5$ using the greedy NMS scheme of Felzenszwalb *et al*. [38]. We then fix this value of λ and use cross-validation to determine the best overlap value η for the greedy NMS scheme of Felzenszwalb *et al*. [38] in the range of possible values 0.2 to 1.0. The best set of parameters (λ^*, η^*) forms the baseline detector for each object class.

To determine the strength of the contiguity term in the MRF model we fix λ to λ^* and perform cross-validation in the range of 10^{-2} to 10^{-1}. Again, we pick the value α^* which yields best AP on `val` using the greedy NMS scheme of Felzenszwalb *et al*. [38]. Detectors with values close to 10^{-1} usually correspond to sliding window detectors which can only consider a detection completely visible or not visible at all and so we do not consider values larger than 10^{-1}.

The final step consists of determining the overlap threshold η for the NMS scheme presented in Section 4.5 by cross-validation considering possible values in the range of 0.2 to 1.0 while fixing other parameters to $\lambda = \lambda^*$ and $\alpha = \alpha^*$.

Truncated Objects

Similar to the sliding window detectors presented in Chapter 2 and Chapter 3 we pad the feature representation of training and test images on each side with 50% of the window size at each scale. These padded areas have their visibility flags $\boldsymbol{v}$ set to 0, *i.e*. we consider

these areas not visible and do not infer them [53]. For further explanation see Section 2.7.1 and Figure 4.4.

Training & Test Protocol

Following extraction of initial positive and negative training examples bootstrapping is performed. We use 6 updates on latent variables (outer loops). The inner loop performs a maximum of 20 bootstrapping rounds. Each round scans 200 randomly selected negative images for false positives. We select a maximum of 3 false positives per scale and per mixture component. Bootstrapping stops early if the number of false positives drops below 200 for each mixture component. As a final step, a maximum of 20 bootstrapping rounds is performed on 800 randomly selected negative images.

Predicted Bounding Box

Similar to the previous chapters we predict an average bounding box per mixture component. If the visibility flags indicate that the object is smaller than the average bounding box, we contract the bounding box around the visibility flags. All remaining implementation details stay unchanged to Chapter 3.

4.6.2 Quantitative Results

In this section we report the quantitative results of the proposed methods. We compare a baseline detector, which is a mixture-of-experts model without partial visibility modeling, to the proposed method in combination with the greedy NMS scheme of Felzenszwalb *et al*. [38]. In addition we compare these two detector setups to our proposed method when used in conjunction with the new NMS scheme proposed in Section 4.5.

For performance evaluation we use the PASCAL VOC 2010 [30] datasets and methodology as described Section 1.2.1 – precision/recall curve, reporting Average Precision (AP), with bounding box overlap of 50%.

Terminology

We compare different subsets of our methods. The *baseline (short: 'BL')* is a sliding-window detector with $d = 2$ mixture components but with no ability to infer visible regions of an object. The baseline is, however, able to model truncation of an object class as it makes use of the visibility flags of the padded area of each image (see Figure 4.4).

We use 2 mixture components because the empirical difference to 3 mixture components is only small for these regular sliding window detectors.

The *visibility modeling (short: 'VIS')* experiment utilizes a sliding-window detector with $d = 2$ mixture components and visibility modeling enabled, *i.e.* inference over visibility flags is performed. Both methods, *BL* and *VIS*, use the greedy selection scheme of Felzenszwalb *et al.* [38].

The final experiment *visibility modeling + new NMS (short: 'VIS+NMS')* additionally uses the proposed globally optimal non-maximum suppression (NMS) scheme introduced in Section 4.5 instead of the scheme of Felzenszwalb *et al.* [38].

Evaluation of Partial Visibility Modeling

Table 4.1 shows the Average Precision (AP) results for the 20 object classes of the PASCAL VOC 2010 challenge [30]. We compare the AP of the baseline (*BL*) to the use of visibility modeling (*VIS*). Visibility modeling improves AP for 16 of 20 classes and maintains AP for another 2 classes. Mean AP improves from 12.2% to 12.9% – a relative improvement of 5.7%.

To further explain our results we recall the roles of parameters α and λ in our detection scheme. The parameter α sets the strength of the penalty term which is subtracted from the window score in case two neighboring blocks have different visibility flags – see Equation 4.1. In that sense it influences the contiguity of the visibility flags: low values of α allow for sparse occlusion patterns while high values enforce contiguous patterns of visibility. Parameter λ controls the ℓ_2-norm of the weight vectors $\mathcal{W}$ and unary bias terms $\mathcal{U}$. Consequently, high values of λ encourage optimization to consider low values of the elements in $\mathcal{W}$ and $\mathcal{U}$. High values of λ result in low-norm solutions for elements in $\mathcal{W}$ and $\mathcal{U}$, resulting in the penalty term controlled by α (Equation 4.1) gaining greater influence on the detection score.

For our experiments we observe two types of results: First, for most object classes the proposed scheme infers all true-positive detections as fully visible *i.e.* all visibility flags are inferred to be 1, which is caused by high values of α and/or λ. Object classes exhibiting this behavior are for example 'aeroplane', 'cat', 'dog', or 'person'. While for some object classes, *e.g.* 'aeroplane', occlusion is not very common and this behavior is expected, for other object classes the underlying sliding window detectors are not successful in modeling the appearance of an object class *and* partial visibility.

Second, for some object classes ('bicycle', 'motorbike', 'car', 'chair' and to a certain degree 'bus' and 'horse') detections are allowed to have partial visibility. For five of these

AP (on `test`)	aero	bicycle	bird	boat	bottle	bus	car
BL	25.7	**28.8**	0.2	2.4	0.9	30.5	22.3
VIS	**28.0**	27.9	0.2	**3.4**	**1.3**	**32.3**	**23.5**
AP (on `test`)	cat	chair	cow	table	dog	horse	person
BL	2.5	5.1	3.9	5.2	3.7	13.4	**20.5**
VIS	**3.9**	**5.9**	**5.4**	**5.3**	**4.6**	**14.9**	17.6
AP (on `test`)	mbike	plant	sheep	sofa	train	tv	mean
BL	25.6	3.4	7.1	3.6	13.2	26.3	12.2
VIS	**27.3**	**3.7**	**8.1**	**3.8**	13.2	**27.3**	***12.9***

Table 4.1: PASCAL VOC 2010 results (`comp3`, `test`-set). We show Average Precision (AP) for a model with $d = 2$ mixture components with and without partial visibility modeling. The greedy NMS scheme of Felzenszwalb *et al*. [38] was used. AP improves for 16 of 20 object classes and mean AP improves slightly from 12.2 to 12.9.

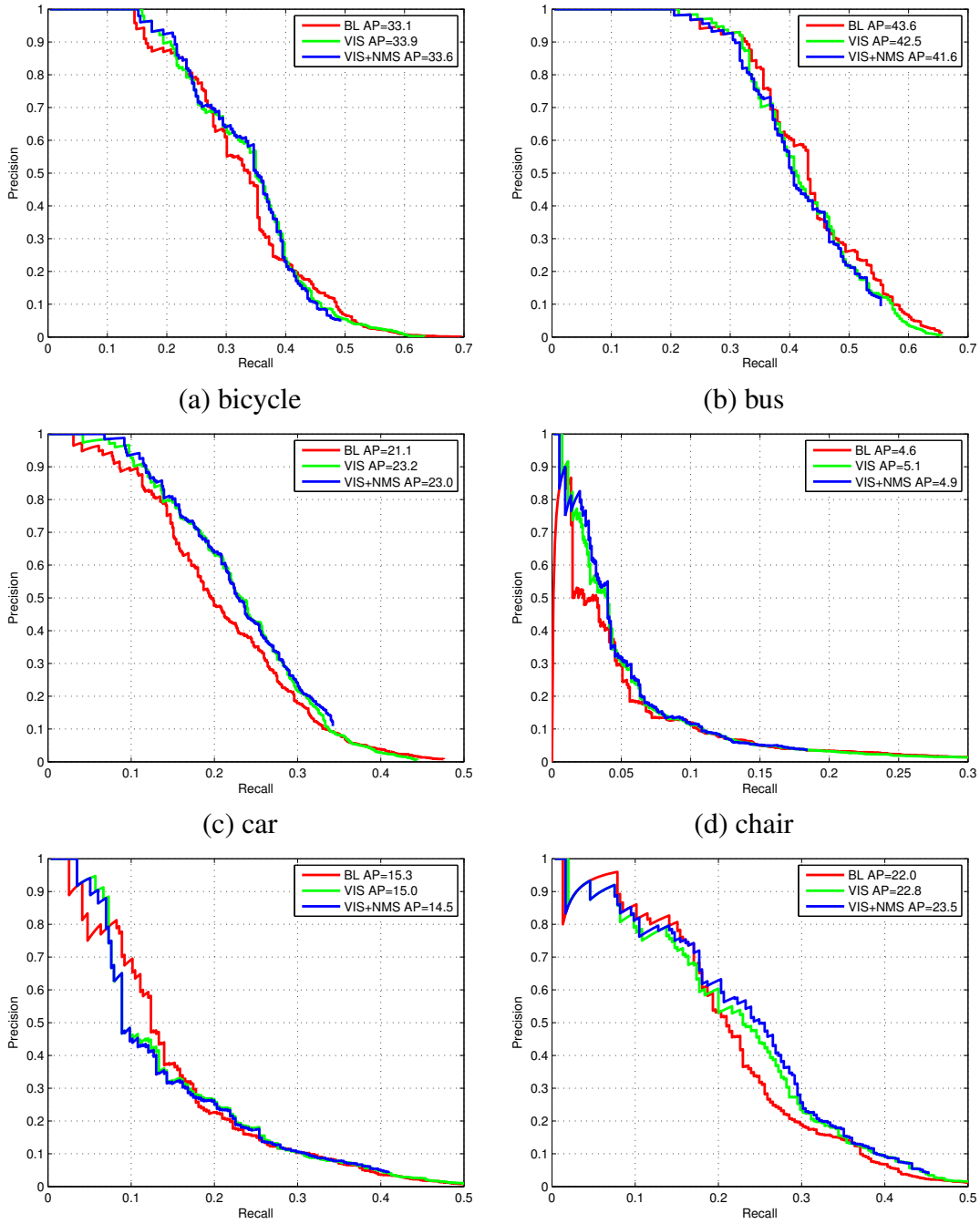
classes ('bus', 'car', 'chair', 'horse' and 'motorbike') AP improves, *e.g*. the 'car' class improves from 22.3% for a model without partial visibility modeling to 23.5% for a model with partial visibility modeling – a relative improvement of 5.4%. Another example is the 'horse' class, which improves from 13.4% to 14.9% – a relative improvement of 11.2%.

Figure 4.10 presents precision/recall curves for the six classes of the PASCAL VOC 2010 dataset for which we are able to model partial visibility. For these plots detectors were trained on the `train` of the PASCAL VOC 2010 dataset [30] and evaluated on the `val` dataset. Substantial AP improvements can be observed for the 'car' and 'chair' class when comparing *BL* to *VIS*. While the AP of the 'car' detector improves from 21.1% to 23.2% (a relative improvement of 10.0%), AP for the 'chair' class improves from 4.6% to 5.1% – a relative improvement of 10.9%. The curves show that recall increases for 'bicycle', 'car', 'chair' and 'motorbike' at almost all precision levels, *i.e*. we are able to detect more objects at comparable precision rates *c.f*. the baseline. Precision/recall curves for the remaining object classes can be found in Appendix A.3.

Globally Optimal NMS Scheme

The proposed NMS scheme gives comparable precision/recall curves to the greedy NMS scheme of Felzenszwalb *et al*. [38] – see blue curves in Figure 4.10. Note that the AP of *VIS* and *VIS+NMS* is not directly comparable due to *VIS* using detections above a threshold of $t = -1.5$ and *VIS+NMS* using $t = -1$ for efficiency reasons.

To allow for a fair comparison we also evaluate the *VIS* experiment at $t = -1$ on PASCAL VOC 2010 `test` and compare the results to *VIS+NMS* in Table 4.2. At this threshold



(a) bicycle (b) bus

(c) car (d) chair

(e) horse (f) motorbike

Figure 4.10: Precision/recall curves for selected PASCAL VOC 2010 classes, comparing the baseline (*BL*, red curves) to partial visibility modeling (*VIS*, green) and partial visibility modeling in combination with the proposed globally optimal NMS scheme (*VIS+NMS*, blue). Modeling of partial visibility gives an increased recall at comparable precision levels for 'bicycle', 'car', 'chair' and 'motorbike' (green & blue curves). AP improves for 4 of the 6 object classes.

AP (on `test`)	aero	bicycle	bird	boat	bottle	bus	car
VIS	**27.7**	27.7	0.1	3.2	1.0	31.9	22.9
VIS+NMS	26.3	**28.1**	0.1	3.2	1.0	**32.5**	**23.7**
AP (on `test`)	cat	chair	cow	table	dog	horse	person
VIS	3.0	5.6	5.2	**4.9**	3.9	**14.4**	**16.2**
VIS+NMS	3.0	**5.8**	**5.3**	4.3	3.9	13.7	15.8
AP (on `test`)	mbike	plant	sheep	sofa	train	tv	mean
VIS	27.1	3.5	**7.8**	**3.3**	**12.8**	**26.9**	***12.5***
VIS+NMS	**27.2**	3.5	7.5	3.2	12.2	25.9	12.3

Table 4.2: PASCAL VOC 2010 results (`comp3`, `test`-set). We show Average Precision (AP) for a models with $d = 2$ mixture components at a threshold of $t = -1$. *VIS* uses the greedy NMS scheme [38] while *VIS+NMS* uses the globally optimal NMS scheme presented in Section 4.5 used. AP improves for 6 of 20 object classes while mean AP drops slightly from 11.4% to 11.3% when comparing *VIS* to *VIS+NMS*.

the detectors will not reach the absolutely highest recall levels but the great majority of true-positive detections lie above that threshold. Since we only compare different NMS schemes these results can be interpreted as how the two different NMS methods perform when presented with the *same* set of detections $\mathcal{D}$ per image.

Table 4.2 compares the *VIS* experiment to the *VIS+NMS* experiment. Overall there is no clear improvement and mean AP decreases slightly from 12.5% to 12.3%. AP improves for 6 object classes and remains the same independent of the NMS scheme used for another 6 object classes. Out of the 6 classes that improve AP 4 detectors model partial visibility as discussed earlier. Improvements can be observed for the 'car' class for which AP improves from 22.9% to 23.7% (a relative improvement of 3.5%), the 'bus' class which improves from 31.9% to 32.5% or the 'bicycle' class with an improvement from 27.7% for *VIS* to 28.1% for *VIS+NMS*. While these results indicate that the new NMS scheme can not provide improved results for the majority of object classes, it is still general *i.e.* we believe it will prove beneficial with improved modeling of partial visibility.

Table 4.3 presents the maximum recall of detectors from the *VIS* and *VIS+NMS* experiments. Comparing maximum recall at a certain detection threshold enables us to determine whether a specific method is able to extract more true-positive detections of an object class. The mean of the maximum recall increases slightly from 31.3% to 31.4%. For some object classes the increase in maximum recall is relatively strong *e.g.* the 'car' object class increases from 33.3% for *VIS* to 34.2% for *VIS+NMS*.

max recall (`val`)	aero	bicycle	bird	boat	bottle	bus	car
VIS	46.9	**50.5**	4.1	7.3	12.2	**57.3**	33.3
VIS+NMS	**49.6**	49.2	4.1	**7.6**	12.2	55.3	**34.2**
max recall (`val`)	cat	chair	cow	table	dog	horse	person
VIS	35.7	18.4	24.2	**26.1**	**23.1**	40.3	**27.0**
VIS+NMS	**36.6**	18.4	**24.6**	24.8	22.1	**41.0**	26.8
max recall (`val`)	mbike	plant	sheep	sofa	train	tv	mean
VIS	44.9	20.3	30.8	33.0	36.5	53.1	31.3
VIS+NMS	**45.2**	**20.6**	**31.1**	**34.8**	**36.9**	**53.6**	**31.4**

Table 4.3: PASCAL VOC 2010 maximum recall results (`comp3`, `test`-set). We show maximum recall at a detection threshold of $t = -1$ and compare *VIS* to *VIS+NMS*. On average the new NMS scheme provides additional positive detections, i.e. recall increases.

4.6.3 Qualitative Results

In the following we establish the applicability of partial visibility modeling to object detection by presenting qualitative results in form of example detections comparing *BL* to *VIS* in Figure 4.11 and *VIS* to *VIS+NMS* in Figure 4.12.

Partial Visibility Modeling

Figure 4.11 shows example detections for four object classes. Compared to the baseline *BL* (dashed yellow bounding boxes), the proposed method *VIS* (solid blue bounding boxes) is able to infer partial visibility – active visibility flags are shown in cyan. Our method detects a wide range of occlusions for these different object classes. In example (f), the person occludes the middle-part of the car and these blocks are correctly considered not visible. Our method is also able to deal with larger degrees of occlusion, see *e.g.* example (j) in which the horse in the background is significantly occluded, or example (i) in which the bottle occludes large parts of the car.

Note that the inferred visibility flags are not absolutely precise, *e.g.* the person in Figure 4.11 (g) is partially considered visible as part of the car in the background. We note that these blocks might not offer significant negative evidence against the presence of the object, and hence are considered visible. This is also a consequence of the imposed discriminative learning scheme, which will sometimes be 'satisfied' if an occluder resembles part of the object class it occludes or at least does not contribute additional negative evidence. A generative model, which models what it expects to see rather than to discriminate between what it expects to see and what it does not expect to see, might be helpful



(a) bicycle (b) bicycle (c) bicycle

(d) car (e) car (f) car

(g) car (h) car (i) car

(j) horse (k) motorbike (l) motorbike

Figure 4.11: Example detections for the baseline and partial visibility modeling. We show detections from the *BL* and *VIS* experiments. If available, baseline detections are shown as bounding boxes with dashed yellow lines. Our method is able to infer the visible extent of an object (cyan regions) *c.f.* the baseline, which does not have this capability. To avoid visual clutter we only draw one detection per image.

for inferring precise visibility flags in these cases.

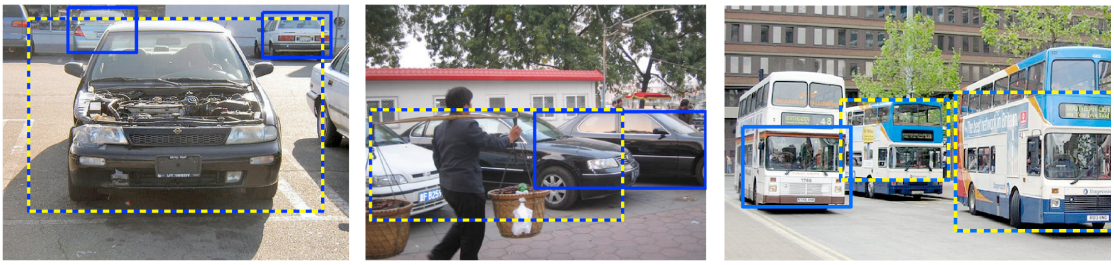
Globally Optimal NMS Scheme

In Figure 4.12 we compare detections to the greedy NMS scheme of Felzenszwalb *et al*. [38] with the globally optimal scheme proposed in Section 4.5. The new NMS scheme yields additional detections, including occluded objects. For example, in (a, b, d) additional occluded instances of the 'car' class are extracted. Examples (k, l) show additional instances of the 'person' class, which the new NMS scheme is able to select in contrast to the greedy NMS scheme of Felzenszwalb *et al*. [38]. These additional detections are mainly selected because the proposed NMS scheme employs a principled overlap measure which considers the overlap of the area the visible parts of two detections share w.r.t. the area both detections cover – see Equation 4.9. This is in contrast to the greedy NMS scheme [38] used by *VIS* which measures the overlap of two detections w.r.t. the area the lower-ranked detection covers and consequently suppresses lower-ranked detections which are largely contained within a higher-ranked detection. However, it should be noted that the majority of detections in Figure 4.12 infers an object instance to be fully visible and our principled NMS scheme is still able to select these detections *c.f*. the scheme of Felzenszwalb *et al*. [38] which suppresses them.

The new NMS scheme is also able to extract entirely new non-occluded detections in comparison to the scheme of Felzenszwalb *et al*. [38] – see *e.g*. Figure 4.12 (h). While the greedy scheme of Felzenszwalb *et al*. [38] first selects the highest ranked detection in a scene, our scheme maximizes the block-wise score – see Figure 4.9. This can consequently yield different detections.

In summary, the proposed NMS scheme is able to extract additional detections due to the principled overlap measure and the block-wise energy optimization. It does so at the cost of a slightly lower precision rate, *i.e*. it also extracts more false-positives on average. These additional false-positives are mainly a consequence of the imposed overlap measure. While the new overlap measure allows partially visible object instances to remain as an element of the scene interpretation, it may also allow other false-positives satisfying the overlap constraint of Equation 4.10 to remain as an element of the scene interpretation – see also Section 4.5.3 and Figure 4.7. The greedy NMS scheme of Felzenszwalb *et al*. [38] commonly removes such false-positive detection instances as it considers the overlap of bounding boxes rather than the overlap of visibility flags.

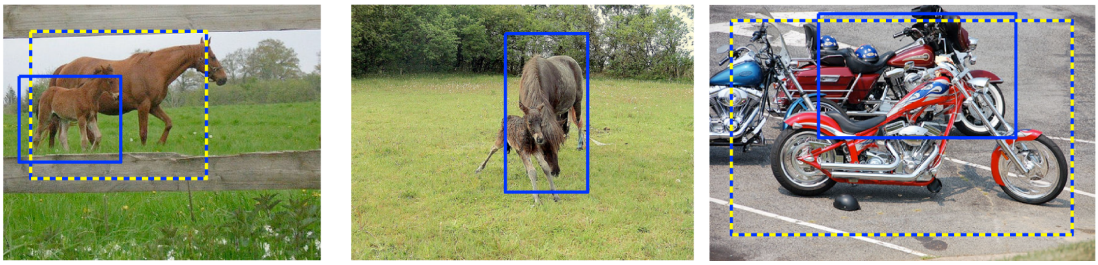
The positive aspects of the proposed NMS scheme remain: (i) it is more principled than the greedy NMS scheme [38] in that it takes partial visibility into account. (ii) It



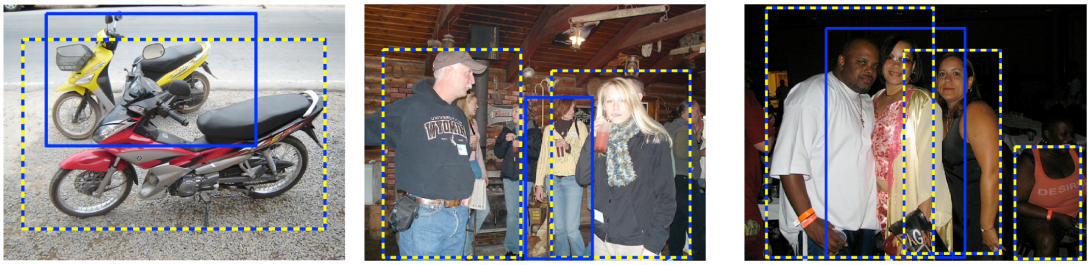
(a) car (b) car (c) bus



(d) car (e) chair (f) chair



(g) horse (h) horse (i) motorbike



(j) motorbike (k) person (l) person

Figure 4.12: Additional detections of the proposed NMS scheme (*VIS+NMS*). If available, *VIS* detections are drawn as yellow bounding boxes with dashed lines. The new NMS scheme yields additional occluded detections. We are also able to extract entirely new fully visibly detections *c.f*. *VIS* – see *e.g*. (g, h).

provides a globally optimal solution and (iii) it is general, *i.e*. we believe we can improve it with better modeling of partial visibility.

4.7 Conclusions

This chapter proposed a method for detection of partially visible objects in a sliding-window framework. Key contributions of the method are the treatment of partial visibility as latent variables and a non-maximum suppression scheme that makes use of additional cues in form of partial visibility.

Quantitatively the proposed method yields improved results for 16 of 20 object classes and mean AP increases from 12.2% to 12.9% – see Table 4.1. The majority of object classes does however not model partial visibility as true-positive detections are either considered completely visible or not visible at all. We plan to improve this situation in future work. The presented qualitative results are promising in that we are able to detect object instances exhibiting many types of occlusion – see Figure 4.11. In addition our method infers the visible extent of each detection and therefore provides additional information about the image scene *c.f*. common object detection methods [38, 134] which do not have this ability.

The novel NMS scheme takes partial visibility into account by defining a principled overlap measure between two detections (Equation 4.9). Quantitative results show that the method improves AP for some object classes while qualitative results indicate that more object instances are selected by the proposed NMS scheme compared to the greedy NMS scheme of Felzenszwalb *et al*. [38].

In summary, the proposed method offers possibilities to detect partially visible objects and on average performs better than a model without the ability to consider partial visibility while also providing promising qualitative results in that the proposed method is able to report on the visible extent of an object instance.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This thesis investigated the problem of object detection in still images and is separated into three different parts. Each part considers a different problem in the research area of object detection: (i) the development of a new segmentation-based feature descriptor; (ii) the extension of deformable part-based models to share parts and better handle finite training data, and (iii) the detection of partially visible objects.

In Chapter 2 we proposed a new feature descriptor, named CHOG, which builds on the popular HOG features [23] and takes advantage of potentially useful instance-specific object characteristics. These features implicitly incorporate color-based segmentation of an object into the feature extraction process and as a result offer the possibility to exploit potential foreground/background relationships in an image window. In addition to combining the new feature descriptor with a linear SVM, we also combined it with an SVM using an inhomogeneous quadratic kernel, which can model dependencies between pairs of elements of the feature descriptor.

Chapter 3 deals with deformable part-based models (DPMs) [38], the current state-of-the-art in object detection [32]. We extended the DPM with the ability to share parts across mixture components (*i.e.* subclasses, which often correspond to viewpoints) and object classes, resulting in a more efficient use of finite training data and a reduction in

the number of model parameters as parts can be reused across viewpoints/object classes. The proposed method allows, for example, the sharing of the appearance of a car wheel between two mixture components representing *e.g.* side-view and three-quarter-view cars.

In the first part of Chapter 4 we proposed a method to detect partially visible object instances by treating the visibility of an object class as a latent variable to be inferred at test *and* training time. The second part of Chapter 4 presented a novel non-maximum suppression scheme which takes the partial visibility of object instances into account by defining a principled overlap measure, considering *only* the overlap of the visible area of two detections.

5.1 Conclusions

5.1.1 Implicit Color Segmentation Features

The incorporation of instance specific segmentation cues into the feature extraction process has, to our knowledge, not been proposed before for the task of object detection. The ability to model consistent foreground/background relationships as a result of the imposed segmentation model is also novel and provides additional recognition cues *c.f.* HOG features [23] which cannot distinguish between foreground and background.

The proposed method gives significantly improved detection accuracy over HOG features [23] (see Sections 2.6.2 and 2.7.2) while qualitative results indicate that detectors using CHOG features are able to detect more object instances *c.f.* detectors which only use HOG features – see Section 2.7.3. These additional detections are often object instances which are imaged under poor lighting conditions, in which case the segmentation model can improve weak gradient responses, or uniformly colored object instances for which the segmentation models work especially well.

The computation of the proposed CHOG feature descriptor is fast due to the implicit nature of the feature extraction scheme, *i.e.* the gradients of the local segmentations can be computed by a simple linear combination of the original image gradients. However, due to the higher dimensionality of the feature descriptors, classification of the feature vector in the sliding window scheme is usually slower *c.f.* lower-dimensional HOG features.

In our experiments we have additionally established that the use of an SVM classifier with inhomogeneous quadratic kernel can further improve detection accuracy for both HOG and HOG+CHOG features – see Section 2.7.2. The qualitative results in Section 2.7.3 indicate that detectors using a kernel SVM are able to detect additional occluded

object instances as well as visual outliers, *i.e*. object instances which significantly differ from the common appearance of an object class. However, it should be noted that the usage of a kernel SVM is expensive due to the complexity of computing the kernel function for each window in the image.

5.1.2 Shared Parts for Deformable Part-based Models

In Section 1.1 we have identified the high variance of appearance of an object class induced by different viewpoints/poses as a major challenge of object detection. Most current methods either (i) ignore such issues, or hope that they can be implicitly modeled *e.g*. by using a non-linear classifier [97, 134, 115], or (ii) by dividing the data into different sub-classes *e.g*. corresponding to viewpoints [135, 38, 147]. In our work we adopt the second approach while also proposing to share elements of the model across sub-classes - an idea which has previously only been considered by a few other methods (see Section 3.2.1 for an overview).

Specifically our model shares part appearance in a deformable part-based model [38] (DPM) while maintaining individual geometric configurations for parts in different subclasses, *i.e*. while parts might appear similar across viewpoints they are commonly located differently. To the best of our knowledge, the work undertaken in Chapter 3 is the first that considers the sharing of parts across subclasses and object classes in a DPM.

In the course of describing our framework we have also proposed a modular reformulation of the DPM, which defines the detector as an accumulation of individual part responses. This modular reformulation as well as the employment of an energy function for discriminative learning made it possible to conveniently express the extension of the regular DPM to models which are able to share parts across subclasses and object classes.

The sharing of parts results in a more efficient use of finite training data while the number of model parameters is reduced as parts can be reused and do not have to be learned separately for each subclass. In addition, training and test time is usually significantly shortened due to the efficient use of parts and coupled training procedures.

In our experiments we found in Section 3.6.2 that DPMs with shared parts perform better compared to their counterparts without shared parts and often require fewer parts to reach a similar or better performance. In addition, the presented qualitative results show that the DPMs successfully use the learned part appearance across object viewpoints and different object classes – see Section 3.6.3.

5.1.3 Detection of Partially visible Objects

Models based on sliding window detectors which consider the partial visibility of an object instance in a principled way are very rare [50] and often only consider occlusion caused by the truncation of an object by the image boundary [53, 135]. Chapter 4 proposes a method for partial visibility modeling, in which visibility is treated as a latent variable to be inferred at test and training time. A novel aspect of our method is the inference of latent visibility at training time, eliminating the requirement of additional training data annotation indicating the visible extent of each object instance.

In contrast to common object detection schemes, which only report on object location and hallucinate the visible extent of an object instance by predicting a bounding box which assumes the entirety of an object instance to be visible, the proposed method provides additional scene information by indicating the visible extent of an object instance.

The hallucination of the visible extent of an object instance by common object detection schemes also poses a significant problem for non-maximum suppression (NMS) in that partially visible object instances might be suppressed. In contrast, the proposed NMS scheme (Section 4.5) takes the partial visibility of an object class into account by defining a principled overlap measure which considers the visible area two object instances share and hence does not suppress detections which occlude each other.

The NMS scheme is accompanied by an optimization method which is guaranteed to converge to a globally optimal solution *c.f.* other methods such as mean-shift techniques [80, 22], which can arrive in poor local minima corresponding to an unfitting scene interpretation. To our knowledge the proposed NMS scheme has not been considered before and is therefore novel.

Quantitative results indicate that partial visibility modeling improves detection accuracy for most object classes (see Section 4.6.2) while qualitative results indicate that our approach works successfully in that we are able to report on the visible extent of many object instances – see Section 4.6.3. The proposed NMS scheme selects additional true-positive detections due to the use of a principled overlap measure at the cost of a slightly increased false-positive rate.

5.2 Future Work

5.2.1 Implicit Color Segmentation Features

In our work we have shown that, in contrast to previous methods which compute a feature descriptor which is not dependent on local image properties, adapting the descriptor to such local image properties gives improved detection accuracy as well as improved recall, *i.e*. we are able to detect more objects. The promising quantitative and qualitative results indicate that the development of such image descriptors is a promising research direction and deserves further work. Specifically, future work on the segmentation features proposed in Chapter 2 should focus on three directions: (i) improvement of the local segmentation models; (ii) improvement of the computational efficiency of the detection scheme when kernel SVMs are used for classification and (iii) the combination of the proposed feature descriptors with other descriptors.

Improved Segmentation Models

In the future we intend to investigate further the incorporation of segmentation cues in the form of instance-specific models of appearance. The proposed models of foreground and background can be improved by using additional information to build the local segmentation models. For example, a combination of color normalizations, a more advanced discriminative model for separating foreground and background *c.f*. the Fisher discriminant [10] used in our approach, or by building the segmentation model from more 'global' cues rather than basing it merely on small image regions around a CHOG block. However, challenges remain in incorporating such more advanced models of segmentation without excessively compromising computational efficiency.

Improvement of Computational Efficiency

We have shown that using CHOG feature descriptors in combination with HOG feature descriptors can improve detection accuracy over detectors only using HOG features. We have further established that combining HOG+CHOG with an SVM using a quadratic kernel improves detection accuracy further. However, using such an SVM is computationally expensive due to the requirement to compute the kernel function for the feature vector of every window in the image. Further research into improving the computational efficiency of the detectors which use a kernel classifier is therefore worth pursuing. Possibilities for improving the efficiency of sliding window detectors were presented in Section 1.3.4 and

include the use of a cascade architecture [107, 138] or coarse-to-fine search [102, 103]. Furthermore the use of homogeneous feature maps to approximate kernels in a SVM classifier as proven successful in previous studies [136]. Another possibility is to apply dimensionality reduction techniques, such as Principal Component Analysis [10], to the feature vector of each window prior to the computation of the kernel function [115].

Combination of Feature Descriptors

The proposed method combines HOG features with the novel CHOG features in one joint descriptor per window. Future work should also focus on additional feature descriptors which could be added to this combination. Local Binary Patterns (LBP) are one option and have shown promising performance improvements in previous research when combined with HOG features [140].

5.2.2 Shared Parts for Deformable Part-based Models

Chapter 3 established that the sharing of parts in a deformable part-based model (DPM) is beneficial to detection accuracy and allows for a better handling of finite training data. Future work on the methods presented in Chapter 3 extends into four possible directions: (i) scaling of the proposed detectors to more classes; (ii) development of more sophisticated mixture models; (iii) reconsideration of the proposed learning scheme and (iv) the exploration of possibilities in transfer learning.

Scaling to more Classes

In future work we aim to show the applicability of part sharing in enabling scaling to many more classes while maintaining the accuracy achieved by the DPM for the few classes in existing studies [38, 147, 97]. Scaling to more classes is an important component of future research as it allows for the sharing of even more data and can potentially create a universal collection of parts for the modeling of a large number of object classes in a DPM. In addition, the learning and evaluation of detectors for many object classes is generally faster than learning and evaluating each detector separately due to the efficient use of parts and coupled training procedures.

Improved Mixture Models

In our experiments we were not able to improve detection accuracy further by simply increasing the number of mixture components in the mixture model independent of whether parts were shared or not. This suggests that the initialization strategy and the latent update scheme for the mixture components might be limiting factors. Girshick *et al*. [53] suggest that training separate mixture components for left and right facing object instances is beneficial to detection accuracy. Other initialization strategies than the one used in our experiments (k-means clustering on the width/height-ratio of bounding boxes) should be explored. A good starting point might be to arrange object instances of an object class into intuitive subclasses according to a human understanding and proceed from there. Such subclasses might correspond to different types of objects *e.g*. vans, limousines, compact cars and regular cars for the 'car' object class. While such an approach could be seen as "moving backwards", it can potentially lead to new insights into how to automatically organize mixture components for creating models with improved detection accuracy.

Reconsideration of the Learning Scheme

A significant challenge of the DPMs as well as the models considering partial visibility (Chapter 4) is how to efficiently train them. In our scheme we used a heuristic initialization strategy to avoid 'getting lost' in a region of the energy surface with badly generalizing solutions (local minima) at the beginning of training – see Section 3.6.1. The optimization problem itself is a very tough one, with unsatisfactory local minima caused by a non-convex and non-continuous energy surface. This situation will not improve in the future with more complex models capable of sharing information becoming available *e.g*. the grammar model of Girshick *et al*. [54]. Despite significant improvements in modeling capacity, this model only offered a small improvement in detection accuracy over the regular DPM [38] in the recent PASCAL VOC challenge [31]. A major challenge therefore remains in either (i) reformulating these more expressive models in a way which makes training more effective, or (ii) developing a new approach to optimization which should be more domain-specific than the generic approaches currently used.

Possibilities in Transfer Learning

While in previous implementations of the DPM [38, 147] parts were assigned to only one mixture component, the proposed models with part sharing assign a part to all mixture components. This raises the question whether it would be possible to learn detectors for

one object class and reuse their learned part appearances in the training of a detector for another object class. While in our implementation we have suggested the parallel learning of multiple detectors and sharing parts between them, we have not considered reusing the parts of an already learned detector to learn an entirely new detector. Such approaches are referred to as transfer learning [98] and would offer an elegant way to quickly learn object detectors for different classes using a DPM.

5.2.3 Detection of Partially Visible Objects

We have shown in Chapter 4 that partial visibility modeling can increase detection accuracy while providing a better scene understanding *c.f.* common object detection schemes by indicating the visible extent of an object instance. Future work on partial visibility modeling should focus on two main directions: The first direction focuses on how to better model partial visibility by (i) using better appearance models; (ii) improving the model of partial visibility and inference, and (iii) incorporating contextual cues into the detection scheme. The second direction focuses on the learning of models for partial visibility and should concentrate on (iv) the analysis of available datasets and (v) improving partial visibility modeling by providing additional training data annotation and/or segmentation cues.

Better Appearance Models

We believe that in our current approach partial visibility modeling can be improved and applied to more classes by using better detectors. Future work should therefore focus on building detectors which use better appearance models, such as more sophisticated feature descriptors [96, 140] than the ones used in our study. Another possibility is to incorporate partial visibility modeling into the deformable part-based model [38] (Chapter 3) by assigning a latent variable to each part in the DPM which indicates whether this part is visible or not. In case the part is not visible, a unary bias term can be added to the score – a similar approach has recently been presented by Girshick *et al.* [54].

Improved Models of Partial Visibility and Inference

From a broader point of view the question should be asked whether the way we model partial visibility, *i.e.* using a block-based Markov Random Field (MRF) at the window level, is sufficient. Further research should consider the extension to a pixel mask rather than a block-based scheme to allow for more detailed modeling of partial visibility. Challenges

remain in how to efficiently perform inference in such models: in our current implementation we use Graph-Cut [13] to infer a solution. While this approach is fast enough for the proposed method, it will most likely not scale to more precise and/or more sophisticated models of partial visibility. Fast approximations of inference as well as possible improvements to the sliding window detector *e.g*. better upper bounds on the detection score or the use of an cascade architecture (which is aware of partial visibility) should be explored. More sophisticated models of partial visibility should also consider the use of image cues to detect where occlusion is apparent in the image scene *e.g*. edges can provide information regarding the occlusion of an object.

Contextual Relationships

In our approach we have treated each object class separately by learning independent detectors. However, it is obvious that inter- as well as intra-class contextual relationships can be helpful in the modeling of partial visibility. Consider, for example, a car which is half occluded: it is likely that the occluder is another known object class, possibly another car, and can be reliably detected. Approaches to leverage contextual information have been presented in Section 1.3.5 of this thesis. An elegant approach which can be extended to use in combination with partial visibility modeling is that of Desai *et al*. [25] who model the 'geographical' relationships of object instances in an scene by learning how likely it is that one object instance appears *e.g*. on top of another or two object instances appear next to each other. This method could be extended to be used in combination with partial visibility modeling and could further be incorporated into the proposed NMS scheme. Challenges remain in how to model partial visibility in case the object in question is occluded by an unmodeled object *e.g*. a tree or a wall.

In addition, 3D information can be a valuable cue for partial visibility modeling and should be considered for future work. Our model is purely 2D, *i.e*. the model parameters (including those of the MRF) are not updated depending on scale, or based on any perspective information in the scene. This is a possible direction for future research: more advanced models which take 3D scene information into account can potentially improve the modeling of partial visibility.

Data Analysis

In our experiments we were able to successfully model partial visibility for 6 object classes of the PASCAL VOC 2010 dataset [30] . The detectors of all other object classes

considered true-positive detections to be fully visible. This raises the question whether the training data is sufficient and offers enough partially visible object instances to successfully model partial visibility. For the PASCAL VOC 2010 dataset [30] we can report that, for most object classes, many partially visible training instances are available while some object classes *e.g.* 'aeroplane' are naturally not occluded very often. However, it remains an unanswered question whether more data is required to successfully learn the proposed models for all object classes. This issue will become even more important with more complex models which may consider contextual information *e.g.* occlusion between object classes or 3D factors.

Additional Training Data Annotation and Segmentation Cues

One way to ease this problem might be to provide additional annotation regarding partial visibility for the object instances in the training dataset. In our approach we have treated the visibility of an object class as a latent variable to be inferred at training *and* test time, and consequently no additional annotation for the training data regarding the visible parts of object instances had to be provided. Loosening this restriction by allowing for additional information to be used during modeling of partial visibility can be beneficial to detection accuracy and successful learning. Partial annotation regarding the visible extent of an object instance is available for the PASCAL VOC datasets [32] and can be used during training to set the latent visibility flags for some training examples. Furthermore, methods which combine object detection and image segmentation (*e.g.* Ladicky *et al.* [72] or Ramanan [105]) should be considered for improving our models.

5.3 Outlook and Final Thoughts

The field of object detection is a challenging one and will remain so for quite some time. This thesis presented methods towards solutions of individual problems in the research area and we have also presented, sometimes significant, performance improvements compared to the relative baselines.

From a bigger viewpoint, current research mainly extends in two directions: (i) adding more prior knowledge to the detection scheme or (ii) learning major parts of the detection scheme. This thesis has mainly focused on the first direction and current research, *e.g.*, using grammar models for object detection [54], follows this direction as well. The grammar models proposed by Girshick *et al.* [54] explicitly assume an object to be decomposable into smaller parts, which themselves can be decomposed into smaller subparts.

Furthermore, feature descriptors (*e.g*. HOG, CHOG) incorporate various forms of invariance into the image representation *e.g*. translation invariance – an object can move by a small degree without changing the feature descriptor significantly. This choice of image representation and its tunable parameters are another example of prior knowledge which is explicitly incorporated into the detection scheme.

The other direction, which is commonly summarized under the label of "Deep Learning" [57, 8] and was shortly introduced in Section 2.2.4 *learns* how to recognize an object class. For example, a deep learning model might learn how to effectively recognize individual object parts, of which size these parts should be, how to combine them into larger parts, and how to model invariance as well as the degree of invariance required to model object appearance.

Both approaches have their advantages and disadvantages: hand-crafted models, which use a lot of prior knowledge in their design, are usually quite intuitive and fit well into 'common' machine learning schemes (*e.g*. discriminative learning or structural learning). Deep learning models, on the other hand, are attractive due to their enormous modeling capabilities but are often very hard to train. At the time of writing this thesis it is as yet undecided which of these directions, if either, will lead the way towards a solution to the task of object detection. Only one thing is for certain: both directions have to be explored further.

This is the end of the thesis. I hope you have enjoyed reading it and that you will feel inspired to think of ways of improving the methods presented or applying them to your own research. Object detection is a difficult yet beautiful research field, holding many mysteries which have yet to be solved. Much more research is required to work towards a general solution. Although, as I assume, he was unaware of object detection and computer vision in general, nobody could have expressed this situation better than Robert Frost:

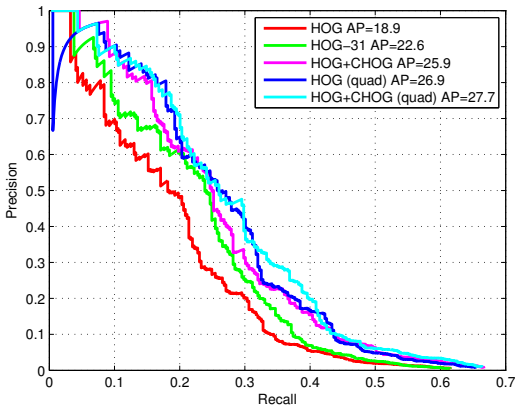
The woods are lovely, dark and deep,
But I have promises to keep,
And miles to go before I sleep,
And miles to go before I sleep.

APPENDIX A

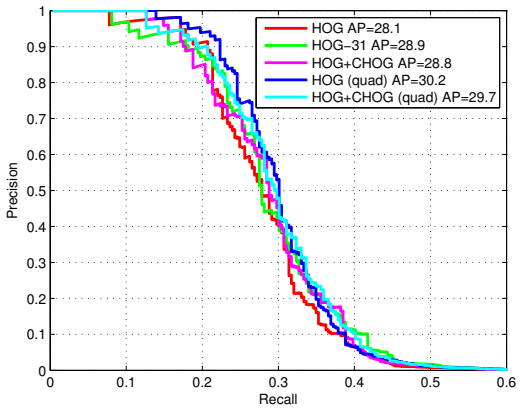
ADDITIONAL RESULTS

A.1 Implicit Color Segmentation Features

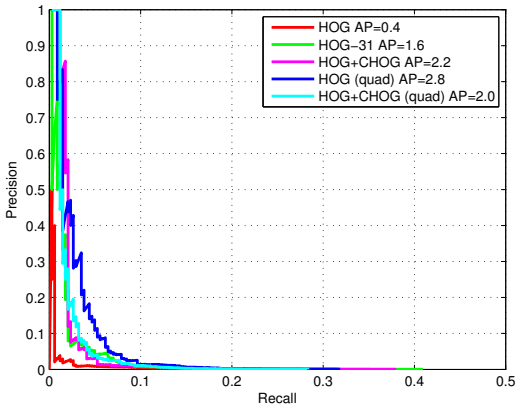
In Section 2.7.2 we presented precision/recall plots for a selection of object classes – see Figure 2.15. We present the remaining 14 plots in Figures A.1, A.2 and A.3. The presented curves compare HOG, HOG+CHOG with linear and kernel SVM respectively.



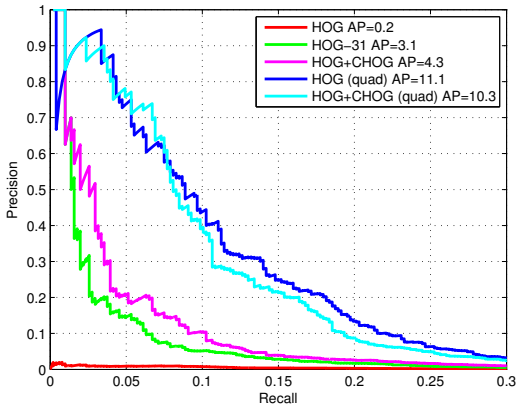
(a) aeroplane



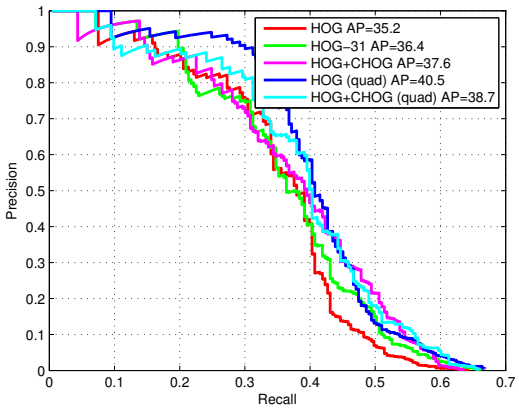
(b) bicycle



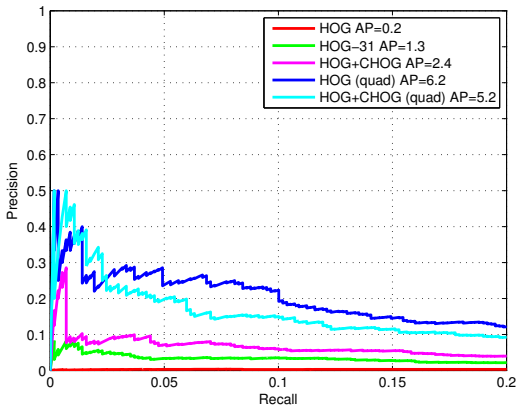
(c) boat



(d) bottle

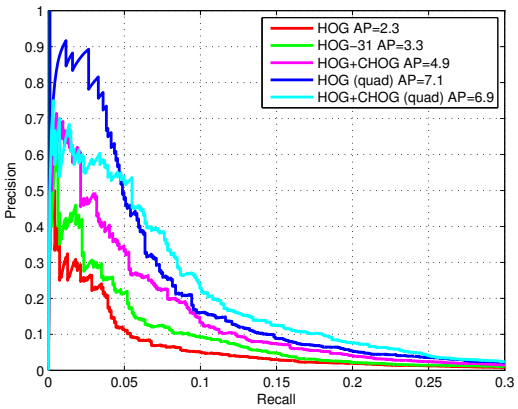


(e) bus

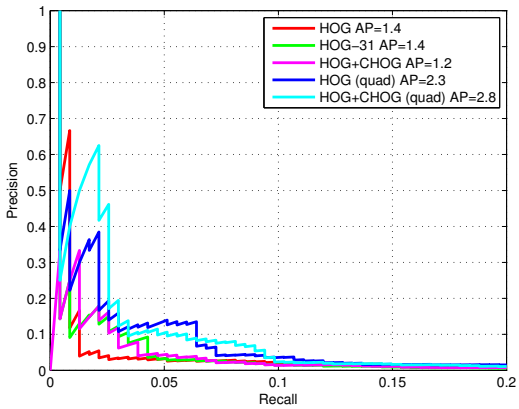


(f) cat

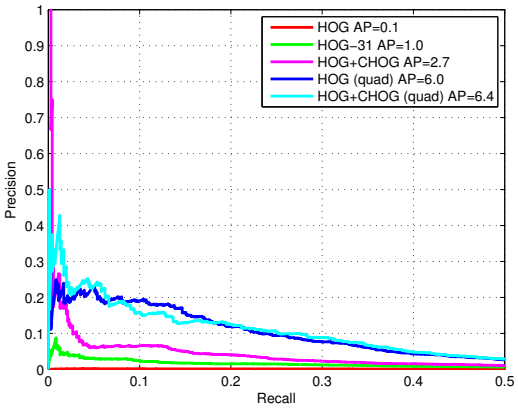
Figure A.1: Precision/recall curves for selected PASCAL VOC 2010 ‘aeroplane’, ‘bicycle’, ‘boat’, ‘bottle’, ‘bus’ and ‘cat’, comparing HOG to HOG+CHOG with linear and quadratic kernel SVM. In addition we also present results for HOG features using signed and unsigned gradients [38] (HOG-31) in combination with a linear SVM.



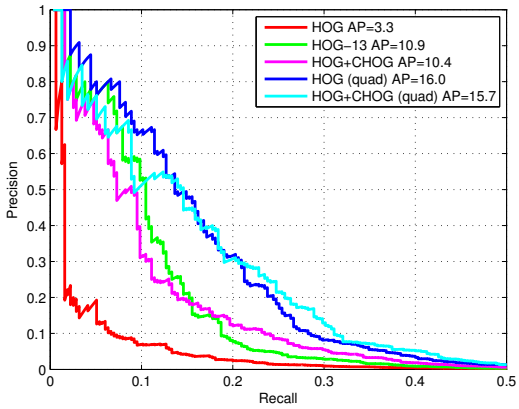
(a) chair



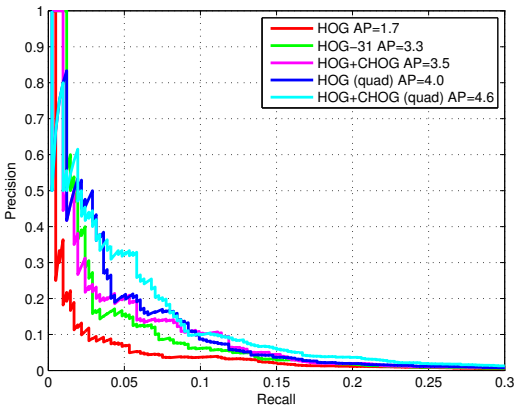
(b) diningtable



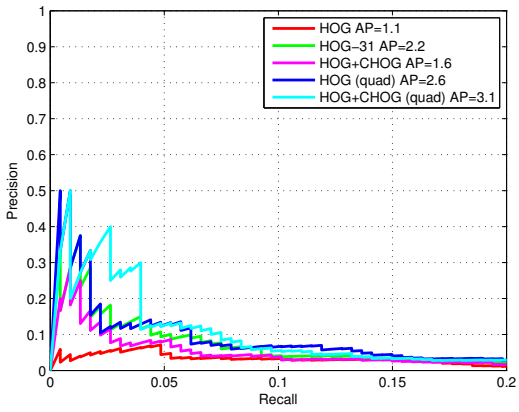
(c) dog



(d) horse

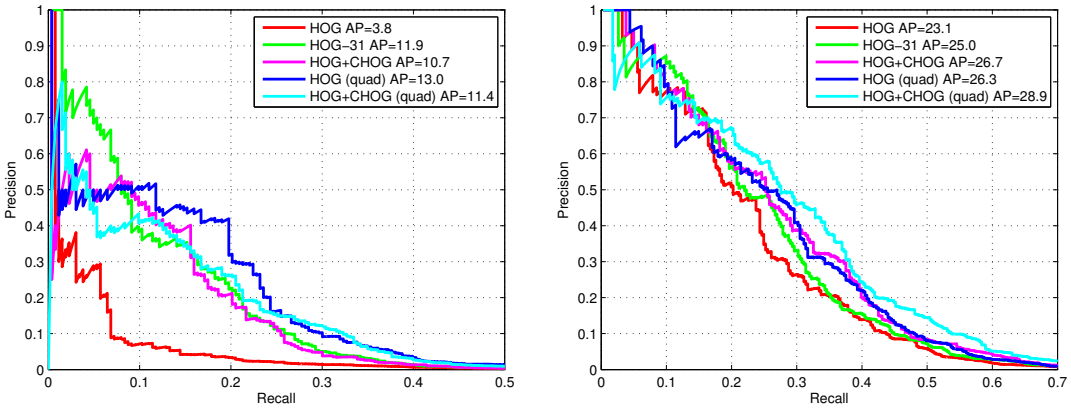


(e) pottedplant



(f) sofa

Figure A.2: Precision/recall curves for selected PASCAL VOC 2010 ‘chair’, ‘diningtable’, ‘dog’, ‘horse’, ‘pottedplant’ and ‘sofa’, comparing HOG to HOG+CHOG with linear and quadratic kernel SVM. In addition we also present results for HOG features using signed and unsigned gradients [38] (HOG-31) in combination with a linear SVM.



(a) train (b) tvmonitor

Figure A.3: Precision/recall curves for selected PASCAL VOC 2010 'train' and 'tvmonitor', comparing HOG to HOG+CHOG with linear and quadratic kernel SVM. In addition we also present results for HOG features using signed and unsigned gradients [38] (HOG-31) in combination with a linear SVM.

A.2 Shared Parts for Deformable Part-based Models

Figure 3.8 shows precision/recall plots for selected object classes of the PASCAL VOC 2010 dataset. The remaining plots are shown in in Figures A.4, A.5 and A.6 for models with and without part sharing.

A.3 Detection of Partially Visible Objects

We have evaluated partial visibility modeling on six selected object classes in Figure 4.10. The remaining precision/recall curves for the partial visibility modeling experiments are shown in Figures A.7, A.8 and A.9.

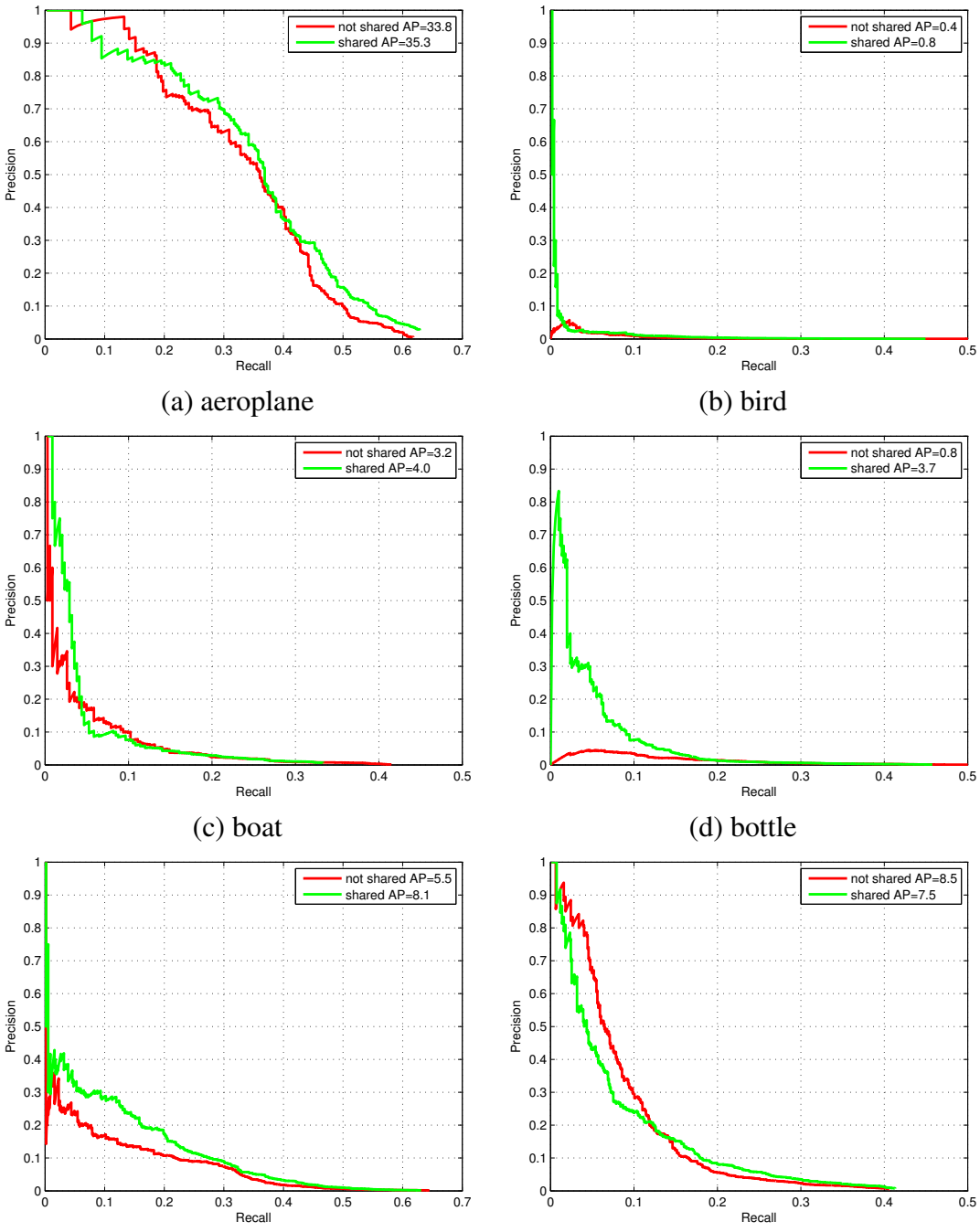
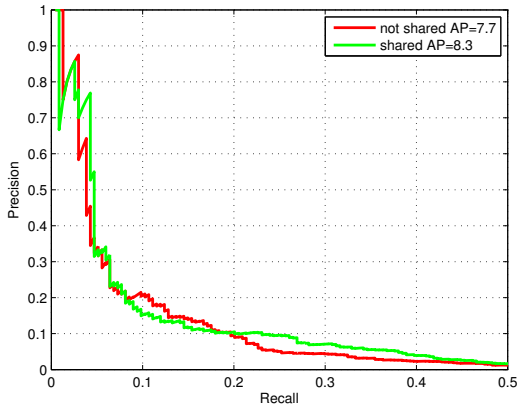
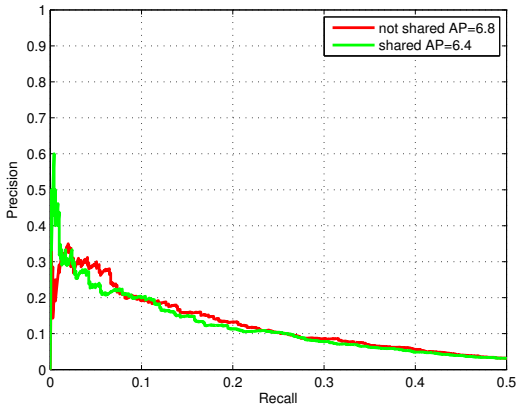


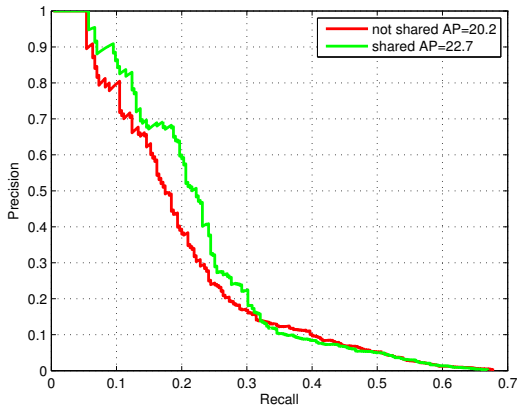
Figure A.4: Precision/recall curves for selected PASCAL VOC 2010 ‘aeroplane’, ‘bird’, ‘boat’, ‘bottle’, ‘cat’ and ‘chair’, comparing DPMs with and without shared parts.



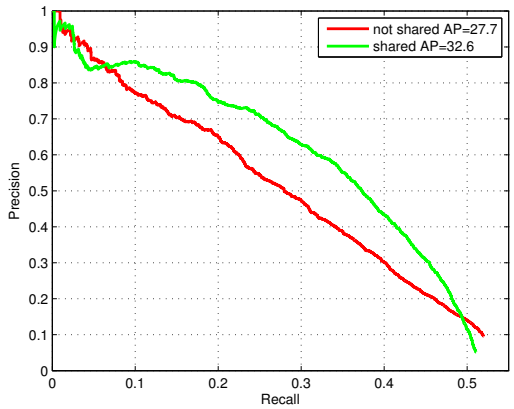
(a) diningtable



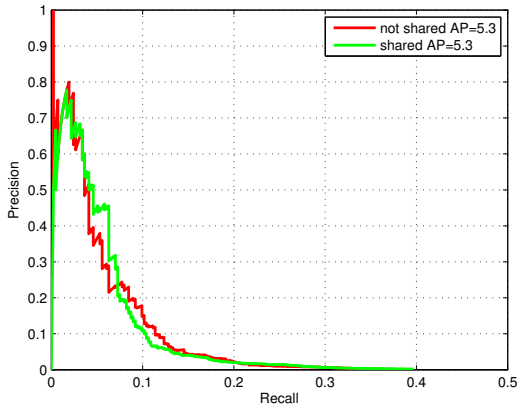
(b) dog



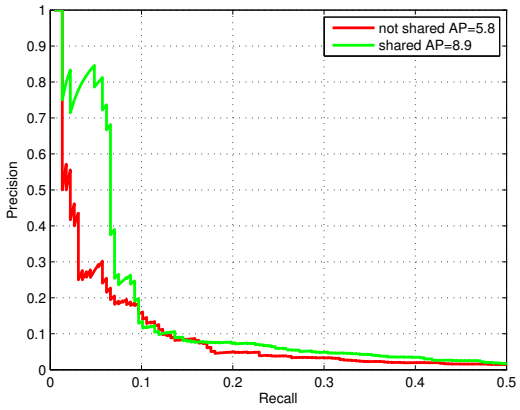
(c) horse



(d) person

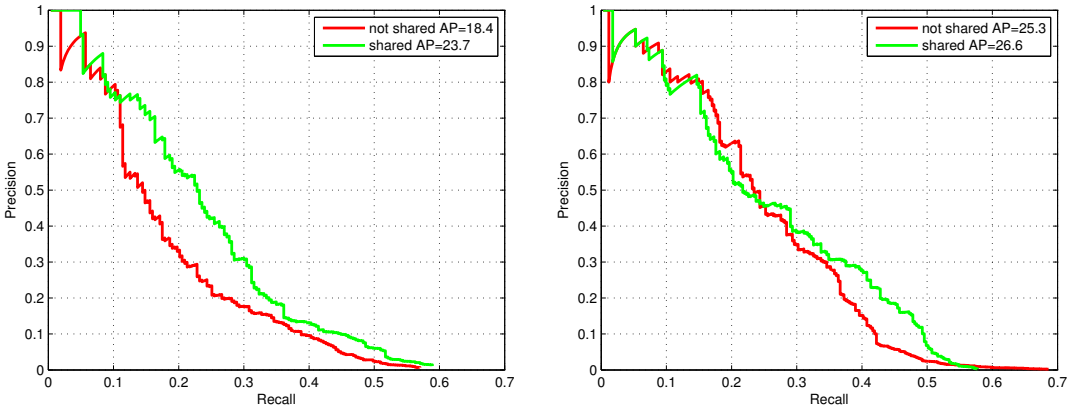


(e) pottedplant



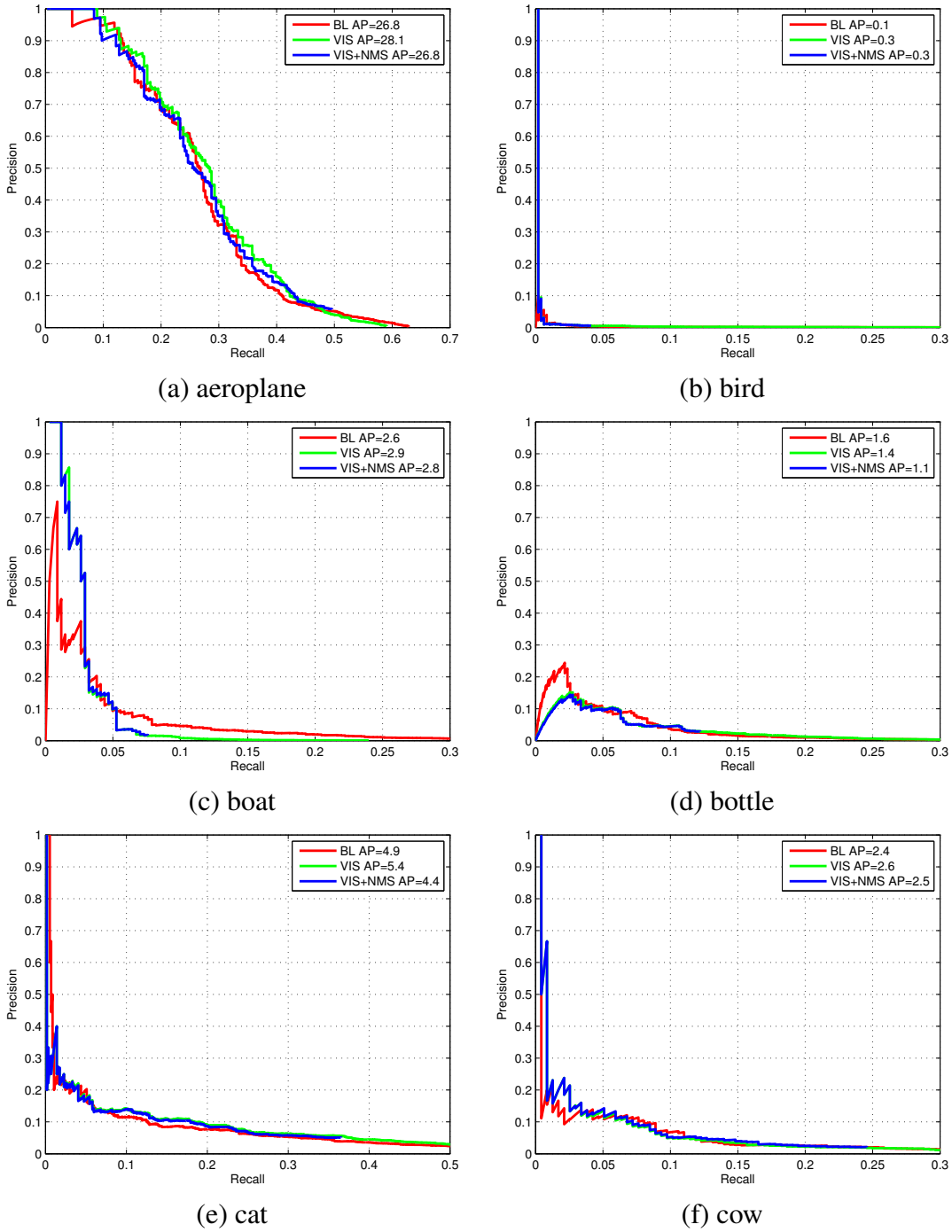
(f) sofa

Figure A.5: Precision/recall curves for selected PASCAL VOC 2010 ‘cow’, ‘diningtable’, ‘dog’, ‘motorbike’, ‘pottedplant’ and ‘sofa’, comparing DPMs with and without shared parts.



(a) train (b) tvmonitor

Figure A.6: Precision/recall curves for selected PASCAL VOC 2010 'train' and 'tvmonitor', comparing DPMs with and without shared parts.

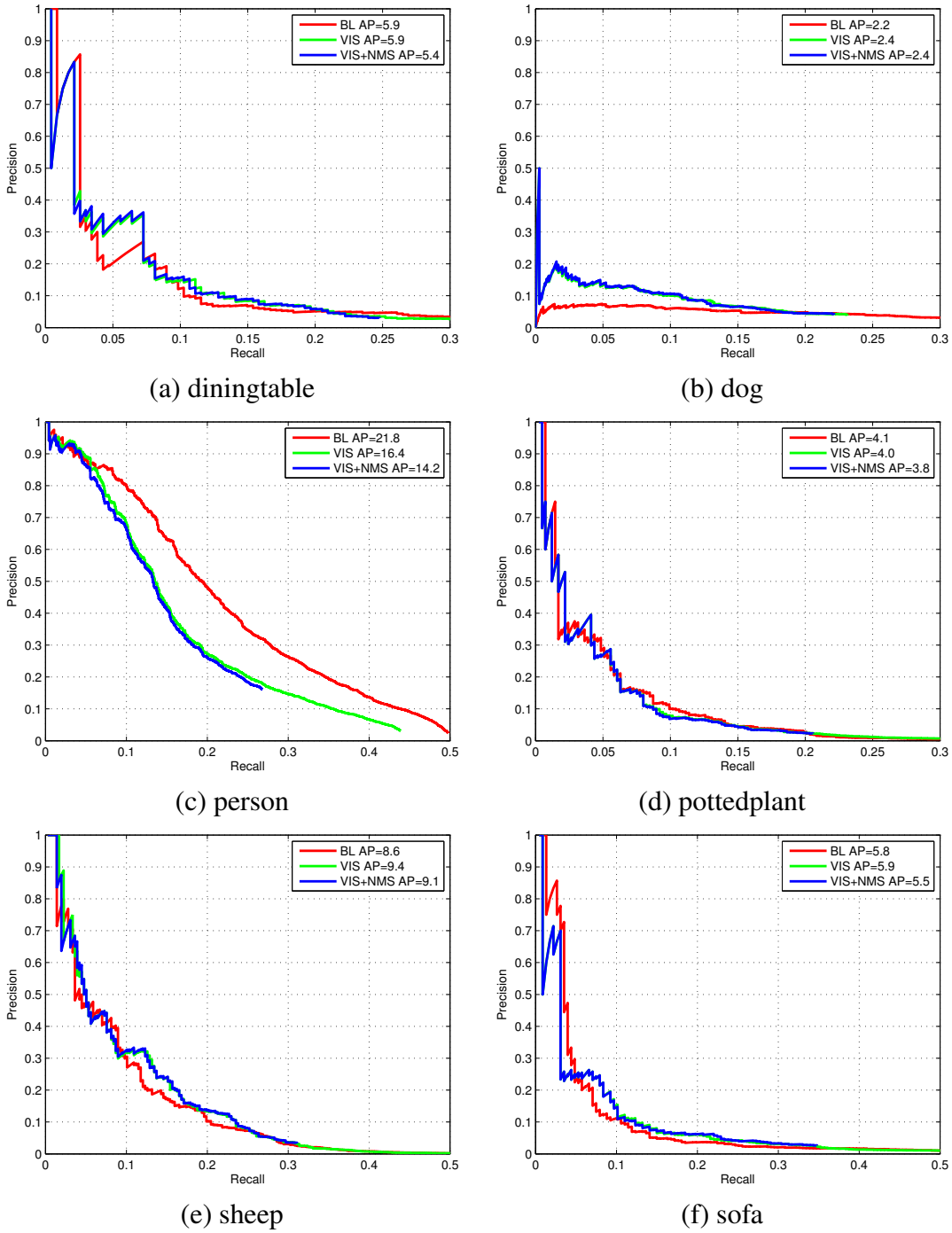


(a) aeroplane (b) bird

(c) boat (d) bottle

(e) cat (f) cow

Figure A.7: Precision/recall curves for selected PASCAL VOC 2010 'aeroplane', 'bird', 'boat', 'bottle', 'cat' and 'chair', comparing the baseline (*BL*), visibility modeling (*VIS*) and visibility modeling with novel NMS scheme (*VIS+NMS*).

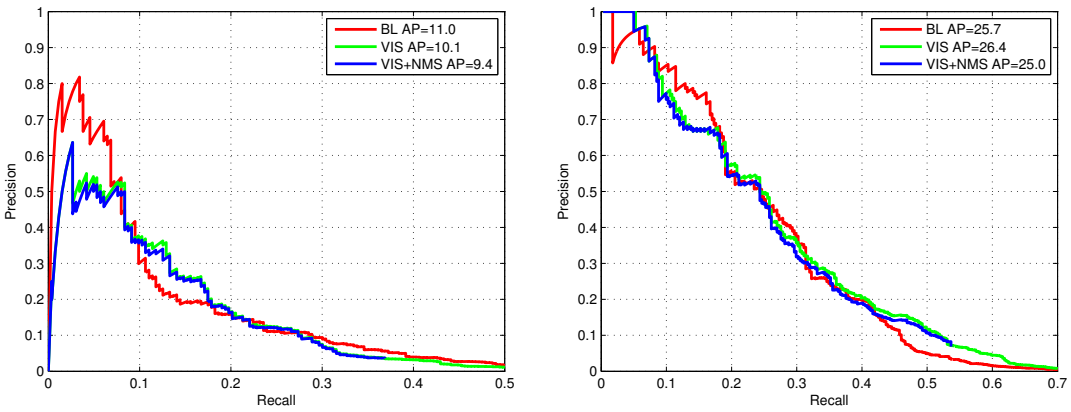


(a) diningtable (b) dog

(c) person (d) pottedplant

(e) sheep (f) sofa

Figure A.8: Precision/recall curves for selected PASCAL VOC 2010 'diningtable', 'dog', 'person', 'pottedplant', 'sheep' and 'sofa', comparing the baseline (*BL*), visibility modeling (*VIS*) and visibility modeling with novel NMS scheme (*VIS+NMS*).



(a) train (b) tvmonitor

Figure A.9: Precision/recall curves for selected PASCAL VOC 2010 'train' and 'tvmonitor', comparing the baseline (*BL*), visibility modeling (*VIS*) and visibility modeling with novel NMS scheme (*VIS+NMS*).

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of the European Conference on Computer Vision*, 2002.

[3] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Proceedings of Neural Information Processing Systems*, 2003.

[4] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[5] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[6] D. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2), 1981.

[7] O. Barinova, V. Lempitsky, and P. Kohli. On the detection of multiple object instances using Hough transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[8] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

[9] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[10] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, 2001.

[14] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[15] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 2001.

[16] R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited-memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5), 1995.

[17] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[18] G. Chunhui, L. Joseph, A. Pablo, and J. Malik. Recognition using regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[19] R. Collins, Y. Liu, and M. Leordeanu. On-line selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), 2005.

[20] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 2002.

[21] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge University Press, 2000.

[22] N. Dalal. *Finding people in images and videos*. PhD thesis, INRIA, 2006.

[23] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[24] V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *Proceedings of the British Machine Vision Conference*, 2010.

[25] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[26] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[27] P. Dollar, C. Wojek, C. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99, 2011.

[28] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. Robust multiperson tracking from a mobile platform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31, 2009.

[29] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[30] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[31] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[32] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2), 2010.

[33] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[34] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, Farquhar J., M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang. The 2005 PASCAL Visual Object Classes Challenge. In *Selected Proceedings of the First PASCAL Challenges Workshop*, 2006.

[35] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[36] F. Felzenszwalb and P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61, 2005.

[37] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[38] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), 2010.

[39] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[40] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[41] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[42] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[43] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Advances in Neural Information Processing Systems 22*, 2009.

[44] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[45] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1), 1973.

[46] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory (EuroCOLT)*, 1995.

[47] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2), 2000.

[48] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, 2011.

[49] C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[50] T. Gao, B. Packer, and D. Koller. A segmentation-aware object detection model with occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[51] D. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73, 2007.

[52] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[53] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[54] R. Girshick, P. Felzenszwalb, and D. McAllester. Object detection with grammar models. In *Proceedings of Neural Information Processing Systems*, 2011.

[55] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 1988.

[56] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[57] G. Hinton, S. Osindero, M. Welling, and Y. Teh. Unsupervised discovery of non-linear structure using contrastive backpropagation. *Cognitive Science*, 30(4), 2006.

[58] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006.

[59] J. Hiriart-Urruty and C. Lemarechal. *Fundamentals of Convex Analysis*. Springer, 2001.

[60] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1), 2008.

[61] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for multi-view object class recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[62] S. Hussain and B. Triggs. Feature sets and dimensionality reduction for visual object detection. In *Proceedings of the British Machine Vision Conference*, 2010.

[63] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3, 1991.

[64] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, 1999.

[65] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010.

[66] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[67] T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2), 2000.

[68] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *On-line Learning for Computer Vision Workshop*, 2009.

[69] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Proceedings of Neural Information Processing Systems*, 2010.

[70] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[71] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[72] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. Torr. What, where & how many? combining object detectors and crfs. In *Proceedings of the European Conference on Computer Vision*, 2010.

[73] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[74] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31, 2009.

[75] Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[76] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 86(11), 1998.

[77] H. Lee, P. Pham, Y. Largman, and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proceedings of Neural Information Processing Systems*, 2009.

[78] A. Lehmann, B. Leibe, and Van Gool. L. PRISM: PRincipled Implicit Shape Model. In *Proceedings of the British Machine Vision Conference*, 2009.

[79] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77, 2008.

[80] B. Leibe and B. Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In *Proceedings of the Annual Symposium of the German Association for Pattern Recognition (DAGM)*, 2004.

[81] Y. Lin and C. Fuh. Fast object detection with occlusions. In *Proceedings of the European Conference on Computer Vision*, 2004.

[82] Z. Lin and L. Davis. A pose-invariant descriptor for human detection and segmentation. In *Proceedings of the European Conference on Computer Vision*, 2008.

[83] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.

[84] J. MacCormick and A. Blake. Spatial dependence in the observation of visual contours. In *Proceedings of the European Conference on Computer Vision*, 1998.

[85] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[86] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[87] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH)*, 1997.

[88] David Meger, Marius Muja, Scott Helmer, Ankur Gupta, Catherine Gamroth, Tomas Hoffman, Matthew A. Baumann, Tristram Southey, Pooyan Fazli, Walter Wohlkinger, Pooja Viswanathan, James J. Little, David G. Lowe, and James Orwell. Curious george: An integrated visual search platform. In *Canadian Conference on Computer and Robot Vision*, 2010.

[89] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[90] S. Munder and D. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11), 2006.

[91] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1), 1996.

[92] R. Okada and S. Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. In *Proceedings of the European Conference on Computer Vision*, 2008.

[93] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the European Conference on Computer Vision*, 2006.

[94] A. Opelt, A. Pinz, and A. Zisserman. Learning an alphabet of shape and appearance for multi-class object detection. *International Journal of Computer Vision*, 80(1), 2008.

[95] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical report, MIT, 1997.

[96] P. Ott and M. Everingham. Implicit color segmentation features for pedestrian and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[97] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[98] S. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22, 2010.

[99] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proc. IEEE Intelligent Vehicles Symposium*, 1998.

[100] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *Proceedings of the IEEE International Conference on Image Processing*, 1999.

[101] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2, 1901.

[102] M. Pedersoli, J. Gonzàlez, A. Bagdanov, and J. Villanueva. Recursive coarse-to-fine localization for fast object detection. In *Proceedings of the European Conference on Computer Vision*, 2010.

[103] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[104] D. Ramanan. Learning to parse images of articulated bodies. In *Proceedings of Neural Information Processing Systems*, 2006.

[105] D. Ramanan. Using segmentation to verify object hypotheses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[106] M. Ranzato, Y-Lan Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Proceedings of Neural Information Processing Systems*, 2007.

[107] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Efficient face detection by a cascaded support-vector machine expansion. *Proceedings of The Royal Society of London*, 460(2501), 2004.

[108] D. Rumelhart, G. Hintont, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986.

[109] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[110] M. Sadeghi and A. Farhadi. Recognition using visual phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[111] R. Schapire. The strength of weak learnability. *Machine Learning*, 5, 1990.

[112] M. Schmidt, K. Murphy, G. Fung, and R. Rosales. Structure learning in random fields for heart motion abnormality detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[113] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3), 2004.

[114] P. Schnitzspan, S. Roth, and B. Schiele. Automatic discovery of meaningful object parts with latent CRFs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[115] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[116] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[117] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of the European Conference on Computer Vision*, 2006.

[118] J. Sivic. *Efficient Visual Search of Images and Videos*. PhD thesis, University of Oxford, 2006.

[119] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.

[120] S. Smith and M. Lewicki. Learning efficient auditory codes using spikes predicts cochlear filters. In *Proceedings of Neural Information Processing Systems*, 2004.

[121] K. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the Sixth International Workshop on Machine Learning (ML)*, 1989.

[122] M. Sridhar, A. Cohn, and D. Hogg. Unsupervised learning of event classes from video. In *National Conference on Artificial Intelligence (AAAI)*, 2010.

[123] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[124] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[125] A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2), 2003.

[126] A. Torralba and A. Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[127] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 2007.

[128] D. Tran and D. Forsyth. Configuration estimates improve pedestrian finding. In *Proceedings of Neural Information Processing Systems*, 2007.

[129] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[130] T. Tuytelaars, C. Lampert, M. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *International Journal of Computer Vision*, 2010.

[131] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on Riemannian manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[132] C. Van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[133] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[134] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[135] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial truncation. In *Proceedings of Neural Information Processing Systems*, 2009.

[136] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[137] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.

[138] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[139] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63, 2005.

[140] X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *Proceedings of the IEEE International Conference on Computer Vision*, 2009.

[141] O. Williams, A. Blake, and R. Cipolla. The variational Ising classifier (VIC) algorithm for coherently contaminated data. In *Proceedings of Neural Information Processing Systems*, 2005.

[142] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[143] H. Wold. Partial least squares. *Encyclopedia of Statistical Sciences*, 6, 1985.

[144] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[145] C. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

[146] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille. Part and appearance sharing: Recursive compositional models for multi-view. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[147] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[148] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.