HAL open science

Machine Learning Methods for Visual Object Detection

Sabit Ul Hussain

▶ To cite this version:

Sabit Ul Hussain. Machine Learning Methods for Visual Object Detection. General Mathematics [math.GM]. Université de Grenoble, 2011. English. NNT : 2011GRENM070 . tel-00722632v2

HAL Id: tel-00722632

https://theses.hal.science/tel-00722632v2

Submitted on 17 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Imagerie, Vision et Robotique**

Arrêté ministériel

Présentée par

Sibt ul Hussain

Thèse dirigée par **Bill Triggs**

préparée au sein du **Laboratoire Jean Kuntzmann**
et de l'École Doctorale **Mathématiques, Sciences et Technologies de l'Information, Informatique**

Machine Learning Methods for Visual Object Detection

Thèse soutenue publiquement le **07 Décembre, 2011**
devant le jury composé de:

Mr. Andrew Zisserman
University of Oxford, England, Rapporteur
Mr. Jiri Matas
Czech Technical University, Czech Republic, Rapporteur
Mr. James Crowley
Institut Polytechnique de Grenoble, France, Examinateur
Mr. Fréderic Jurie
Université de Caen, France , Examinateur
Mr. Bill Triggs
Laboratoire Jean Kuntzmann (CNRS), Directeur de thèse

Dedicated to my beloved parents and grandparents.

Abstract

The goal of this thesis is to develop better practical methods for detecting common object classes in real world images. We present a family of object detectors that combine Histogram of Oriented Gradient (HOG), Local Binary Pattern (LBP) and Local Ternary Pattern (LTP) features with efficient Latent SVM classifiers and effective dimensionality reduction and sparsification schemes to give state-of-the-art performance on several important datasets including PASCAL VOC2006 and VOC2007, INRIA Person and ETHZ. The three main contributions are as follows.

Firstly, we pioneer the use of Local Ternary Pattern features for object detection, showing that LTP gives better overall performance than HOG and LBP, because it captures both rich local texture and object shape information while being resistant to variations in lighting conditions. It thus works well both for classes that are recognized mainly by their structure and ones that are recognized mainly by their textures. We also show that HOG, LBP and LTP complement one another, so that an extended feature set that incorporates all three of them gives further improvements in performance.

Secondly, in order to tackle the speed and memory usage problems associated with high-dimensional modern feature sets, we propose two effective dimensionality reduction techniques. The first, feature projection using Partial Least Squares, allows detectors to be trained more rapidly with negligible loss of accuracy and no loss of run time speed for linear detectors. The second, feature selection using SVM weight truncation, allows active feature sets to be reduced in size by almost an order of magnitude with little or no loss, and often a small gain, in detector accuracy. Despite its simplicity, this feature selection scheme outperforms all of the other sparsity enforcing methods that we have tested.

Lastly, we describe work in progress on Local Quantized Patterns (LQP), a generalized form of local pattern features that uses lookup table based vector quantization to provide local pattern style pixel neighbourhood codings that have the speed of LBP/LTP and some of the flexibility and power of traditional visual word representations. Our experiments show that LQP outperforms all of the other feature sets tested including HOG, LBP and LTP.

Résumé

Le but de cette thèse est de développer des méthodes pratiques plus performantes pour la détection d'instances de classes d'objets de la vie quotidienne dans les images. Nous présentons une famille de détecteurs qui incorporent trois types d'indices visuelles performantes – histogrammes de gradients orientés (Histograms of Oriented Gradients, HOG), motifs locaux binaires (Local Binary Patterns, LBP) et motifs locaux ternaires (Local Ternary Patterns, LTP) – dans des méthodes de discrimination efficaces de type machine à vecteur de support latent (Latent SVM), sous deux régimes de réduction de dimension – moindres carrées partielles (Partial Least Squares, PLS) et sélection de variables par élagage de poids SVM (SVM Weight Truncation). Sur plusieurs jeux de données importantes, notamment ceux du PASCAL VOC2006 et VOC2007, INRIA Person et ETH Zurich, nous démontrons que nos méthodes améliorent l'état de l'art du domaine.

Nos contributions principales sont :

Nous étudions l'indice visuelle LTP pour la détection d'objets. Nous démontrons que sa performance est globalement mieux que celle des indices bien établies HOG et LBP parce qu'elle permet d'encoder à la fois la texture locale de l'objet et sa forme globale, tout en étant résistante aux variations d'éclairage. Grâce à ces atouts, LTP fonctionne aussi bien pour les classes qui sont caractérisées principalement par leurs structures que pour celles qui sont caractérisées par leurs textures. En plus, nous démontrons que les indices HOG, LBP et LTP sont bien complémentaires, de sorte qu'un jeux d'indices étendu qui intègre tous les trois améliore encore la performance.

Les jeux d'indices visuelles performantes étant de dimension assez élevée, nous proposons deux méthodes de réduction de dimension afin d'améliorer leur vitesse et réduire leur utilisation de mémoire. La première, basée sur la projection moindres carrés partielles, diminue significativement le temps de formation des détecteurs linéaires, sans réduction de précision ni perte de vitesse d'exécution. La seconde, fondée sur la sélection de variables par l'élagage des poids du SVM, nous permet de réduire le nombre d'indices actives par un ordre de grandeur avec une réduction minime, voire même une petite augmentation, de la précision du détecteur. Malgré sa simplicité, cette méthode de sélection de variables surpasse toutes les autres approches que nous avons mis à l'essai.

Enfin, nous décrivons notre travail en cours sur une nouvelle variété d'indice visuelle – les « motifs locaux quantifiées » (Local Quantized Patterns, LQP). LQP généralise les indices existantes LBP/LTP en introduisant une étape de quantification vectorielle – ce qui permet une souplesse et une puissance analogue aux celles des approches de reconnaissance visuelle « sac de mots », qui sont basées sur la quantification des régions locales d'image considérablement plus grandes – sans perdre la simplicité et la rapidité qui caractérisent les approches motifs locales actuelles parce que les résultats de la quantification puissent être pré-compilés et stockés dans un tableau. LQP permet une augmentation considérable de la taille du support local de l'indice, et donc de sa puissance discriminatoire. Nos expériences indiquent qu'elle a la meilleure performance de toutes les indices visuelles testés, y compris HOG, LBP et LTP.

Acknowledgments

Plans go wrong for lack of advice; many advisers bring success.

This thesis would not have been possible without the continuous support of my supervisor, Dr. Bill Triggs. His enthusiasm, motivation and patience and his efforts to explain things simply and clearly helped me to develop a sound understanding of the subject area and to develop the results in the thesis. While writing the manuscript, he provided me with sound advice and encouraged me to write in a simple, logical and understandable way. I was fortunate to have a great mentor and adviser.

I would like to thank my thesis committee Professors Andrew Zisserman, Jiri Matas, Frédérice Jurie and James Crowley for their encouragement and insightful comments. I would also like to express my gratitude to all of my colleagues and friends with whom I had the opportunity of working during this period, and to my teachers for helping me to grasp the basics, for having confidence in me and motivating me through the difficult times, and for persuading me to keep on advancing. I would also like to acknowledge the broader research community for sharing their ideas so openly and the open-source community for providing many useful programs that aided the work.

The Higher Education Commission of Pakistan and the European Commission (via the research project CLASS) provided the financial support for this research. Without this, my thesis would not have been possible.

I am indebted to my Pakistani friends and colleagues in Grenoble, who supported me through thick and thin, sharing many joyful moments via different sports and events and making my stay in Grenoble a time that I will always cherish.

Finally, and most importantly, my parents and grandparents deserve a special mention for their continuous and unconditional love, support, sacrifices and prayers. They deserve all of the credit for lighting the candle and keeping it alight without thought for themselves. Nor can I forget the love of my dear sisters, who have always motivated and inspired me and comforted me with their sweet gossip and laughter.

Contents

Abstract iii

Résumé v

Acknowledgments vii

1 Introduction 1
1.1 Challenges . . . 2
1.2 General Approach . . . 4
1.3 Main Contributions . . . 6
1.4 Thesis Organization . . . 7

2 State of the Art 9
2.1 Classifier Architecture . . . 10
2.1.1 Bottom-Up Approaches . . . 10
2.1.2 Top-Down Approaches . . . 11
2.2 Descriptors . . . 12
2.3 Classifiers . . . 14
2.4 Post-Processing . . . 17
2.5 Computational Efficiency . . . 18
2.6 Relationship to Past Work . . . 19

3 Training Methods and Evaluation Metrics 21
3.1 Training Mechanisms . . . 21
3.2 Non-latent Training . . . 22
3.2.1 Hard negatives . . . 23
3.2.2 SVM Algorithm . . . 23
3.3 Latent Training . . . 25
3.3.1 Latent Support Vector Machines . . . 25
3.3.2 Stages of Latent SVM Training . . . 26
3.3.3 Discussion . . . 27
3.4 Bilateral Symmetry . . . 29
3.4.1 Folding . . . 29
3.4.2 Mirrored Pairs . . . 30
3.5 Post-Processing . . . 31
3.5.1 Non-Maximum Suppression (NMS) . . . 31
3.5.2 Bounding Box Prediction. . . . 32
3.6 Evaluation Metrics . . . 32

3.6.1 Detection Error Tradeoff (DET) Plots . . . 33
3.6.2 Precision-Recall Plots . . . 33
3.6.3 Recall-False Positive Per Image (Recall-FPPI) Plots . . . 35

4 Feature Sets **37**
4.1 Histograms of Oriented Gradients (HOG) . . . 38
4.1.1 Implementation Details . . . 38
4.1.1.1 HOG Dimensionality Reduction . . . 39
4.1.2 Results and Discussion for HOG . . . 40
4.2 Local Pattern Features . . . 41
4.2.1 Local Binary Patterns (LBP) . . . 42
4.2.2 Local Ternary Patterns (LTP) . . . 44
4.2.3 Center-Symmetric Local Patterns (CS-LBP/LTP) . . . 46
4.2.4 Parameter Settings for Local Pattern Features . . . 47
4.2.4.1 Image Preprocessing . . . 47
4.2.4.2 Color Space . . . 48
4.2.4.3 Feature map structure . . . 49
4.2.4.4 Cell Normalization . . . 50
4.2.4.5 LTP Threshold . . . 52
4.3 Combinations of Features . . . 52
4.4 Speed . . . 54
4.5 Summary . . . 55

5 Dimensionality Reduction and Classifiers **57**
5.1 Dimensionality Reduction . . . 57
5.1.1 Discriminant Subspace Projection . . . 57
5.1.1.1 Partial Least Squares (PLS) . . . 58
5.1.2 Feature Selection and Sparsity . . . 64
5.1.2.1 Boosting For Feature Selection . . . 64
5.1.2.2 L_1 Feature Selection . . . 65
5.1.2.3 Weight Truncation Based Sparse Classifiers . . . 68
5.2 Nonlinear Classifiers . . . 74
5.2.1 Nonlinear Feature Extension . . . 74
5.2.1.1 Piecewise Feature Extension . . . 75
5.2.1.2 Componentwise Quadratic Classifier . . . 76
5.2.2 PLS Quadratic Classifier . . . 78
5.2.3 Cascade of Linear and Nonlinear Detectors . . . 79
5.2.4 Discussion . . . 79
5.3 Summary . . . 80

6 Experiments **81**
6.1 Parameter Settings and Detector Configurations . . . 81
6.2 Results on the INRIA Person Dataset . . . 83
6.2.1 Non-Latent Detectors . . . 84
6.2.2 Latent Detectors . . . 86
6.2.3 Discussion . . . 86
6.3 The PASCAL Visual Object Challenge Datasets . . . 92

6.4 PASCAL VOC2006 93
6.4.1 Feature Sets 94
6.4.2 Single versus Multiple Roots and Parts 95
6.4.3 Partial Least Squares Dimensionality Reduction 97
6.4.4 Sparsification 98
6.5 PASCAL VOC2007 102
6.6 PASCAL VOC2010 104
6.7 ETHZ Dataset 105
6.8 Summary 106

7 Conclusions and Perspectives 109
7.1 Key Contributions 109
7.2 Future Work 111

A Local Quantized Pattern Feature Sets 115
A.1 Local Quantized Patterns 116
A.1.1 Implementation Details 117
A.1.2 Local Pattern Geometry and Sampling 118
A.1.3 Code Book Learning Method 119
A.2 Results and Discussions 120
A.2.1 INRIA Person Dataset 120
A.2.2 PASCAL VOC2006 124
A.2.3 Discussion 125
A.3 Summary 126

References 129

Chapter 1: Introduction

The most incomprehensible thing about the world is that it is comprehensible.
Albert Einstein

We are living in a digital age. Digital cameras, internet connected computers and smart phones are an integral and growing part of our lives. Image and video collections are growing daily: besides personal collections, more than 5 billion images are hosted on Flickr[1], and around 2.1 billion videos on YouTube[2]. There is a common need for intelligent machines that can summarize visual content and present it in readily accessible forms, perform reasoning based on it, and augment it in intelligent ways. This applies both to commercial media (images, film, television) and to images and video of events from everyday life.

Although current computers and smart phones can handle computationally intensive tasks, they are not yet able to perform many of the basic cognitive activities that a normal person performs unconsciously many times throughout the day, and that form the basis of his or her intelligent high level decisions. A case in point is the human ability to interpret visual information, for instance to differentiate between similar object categories like horse and mule or motorbike and bicycle, and to identify objects of the same class despite very different shapes or appearances. For example, cruiser, sports, touring, scooter, dirtbike, *etc*. are all recognized as kinds of motorbikes. Moreover, this is possible despite clutter, occlusions and variations in appearance due to changes in size, position, viewpoint, color, texture, *etc*.

One of the main goals of computer vision is to equip computers with artificial visual systems having human-like image understanding capabilities so that the above goals can be reached. One fundamental task of such systems will be the interpretation and labeling of scene content. Such interpretation can occur at several levels within an image:

[1] Flickr official blog.
[2] Number of returned search results for the video query '*'.

- *Image classification* is the task of annotating entire images according to the elements present in them. It says what is present without necessarily saying where[3].
- *Object detection/localization* is the task of identifying the presence, location and extent of any instances of a given object class that are present in the image. Extent can be indicated by, *e.g.*, bounding boxes or pixel-level masks. Multiple categories can be also detected, however identifying all instances of all of the object classes that are present in everyday images is beyond current technology.
- *Semantic segmentation* is the task of labeling each image pixel with the object class that it was generated by.

In this thesis we will be concerned exclusively with the detection/localization task. We will use the words detection and localization interchangeably. Moreover, although all of the above recognition tasks can also be performed at the level of specific individuals (a given person or building), here we will focus on generic class level detection ("person", "car", "horse", ...).

Reliable practical object detectors would have many applications. Current image content management systems are based mainly on manually supplied meta-data provided either by the uploading user or by a specially employed workforce. Annotation is tedious, costly and error prone, and even at the best it seldom provides very complete coverage (many instances are missed). Multi-category object detectors would allow images to be labelled automatically based on their content, thus facilitating content-based browsing, search and retrieval. Object detection would be also useful for intelligent environments, for example surveillance systems could automatically identify intruders or aged people in need of assistance, and smarts cars could use object/human detection coupled with other cues to avoid collisions. Further application domains include gaming, robotics, entertainment, advertising and manufacturing – indeed any application where intelligent systems need to observe or interact with humans, objects or animals.

1.1 Challenges

We still have only a very rudimentary understanding of the higher levels of human visual perception, but we know that it occupies a substantial portion of the brain – more computation than is readily available with current desktop computers[4] – takes several

[3] http://www.google.com/imghp

[4] Currently, the combined computers of the planet can perform "only" $\sim 6.4 \times 10^{18}$ instructions per second – similar to the maximum number of nerve impulses sent in a human brain per second – but in the next 10-20 years individual computers will reach the same computational capacity. Also, although computers are still quite weak at visual scene understanding, they are already stronger than humans at many tasks including games such as chess and even at general knowledge quizzes – an IBM Watson computer recently beat the best human champions in the quiz show Jeopardy.

Figure 1.1: *Some examples of images of man-made and natural object classes. These illustrate some of the challenges faced by object category detectors.*

years to acquire, and is much richer than current artificial vision systems. Psychophysical studies suggest that a 6 year old child can recognize 10-30 thousand object categories [Biederman 1987]. In contrast, current artificial detectors are based on "comparatively simple" statistical pattern recognition techniques for individual classes, and despite substantial progress made during the past decade, the detection of visual object classes remains a challenging problem that receives a great deal of attention in the vision community [Everingham et al. 2010b,a]. To be practical, any system must address the following issues:

- **Imaging factors:** Real world objects appear under a wide range of illumination conditions in both indoor and outdoor settings. The images may be locally under- or over-exposed and specularities, shadows, *etc.*, are common. Object images are also highly viewpoint dependent: the depth dimension is lost; image scale varies

due to the relative location of the sensor and object; and the appearance of complex object classes varies significantly with viewing angle.

- **Intra-class Variance:** Within a given class, objects can exhibit widely varying shape, color and texture. For example the car category includes convertibles, coupés, hatchbacks, limousines, *etc*. Natural classes such as person, cat, dog, *etc*., also have a wide range of articulated poses and non-rigid deformations, resulting in highly variable object layout and appearance.
- **Background Clutter:** Objects appear against a wide range of backgrounds, often in close proximity to or direct interaction with neighbouring objects.
- **Occlusion/Truncation:** Objects are often occluded by other objects that lie in front of them or truncated by the borders of the image, so that only a portion of the object is visible.

Image categorization methods also have to cope with these challenges, but object detection remains somewhat harder because: (i) it has to recover the object locations and sizes despite pose and appearance variations, occlusions, truncations, *etc*.; (ii) it has to detect each object in the image, not just say whether at least one object of the class is present; (iii) the criteria for success are often stricter – for example, a small person in the background might be irrelevant for scene-type classification but highly relevant for a military surveillance application. These difficulties are apparent in performance figures. For example, in the PASCAL VOC2010 visual object classification and detection challenges [Everingham et al. 2010a], the best object localization methods have precision scores $\sim 2\times$ lower than the best image classification methods. Figure 1.1 shows some examples of challenging real world images containing common object categories.

1.2 General Approach

Given the above-mentioned challenges, the object detection problem appears to be too complex to model analytically, so we resort to a learning-based approach in which a diverse and representative set of training examples is used as a surrogate for a model. Object detection is thus cast as a problem of classifying potential candidates proposed by an underlying object position hypothesis generator, and machine learning is used to learn a decision rule for these hypotheses from a representative training set. This formulation allows advances in machine learning to be leveraged and as with other machine learning problems, many variants have been tested based on supervised, semi-supervised or unsupervised learning of one class, binary or multiple class models. However it must be emphasized that machine learning is only part of the solution. Successful object detection requires a well chosen combination of effective visual representation, domain

modeling and flexible classification, so a typical object detector must make choices in three areas: the object position hypothesis generator; the set of visual descriptors used to capture object shape, color and texture characteristics; and the object/non-object classifier based on these features.

Broadly speaking, current systems can be divided into two main categories. In the first, descriptors are computed only sparsely at locations given by some local feature detector, and these positions are used to generate possible object hypotheses. In the vision community such methods are generally known as local feature or constellation models [Fergus et al. 2003; Leibe et al. 2004]. The local features provide a relatively sparse set of hypotheses and hence a computationally efficient detector, and they also allow voting over a relatively large and diverse set of object parts (local features), but at present the underlying features that they are based on tend to be too local and too generic to allow state-of-the-art levels of object discrimination. Models with many inter-connected parts also tend to have high combinatorial complexities, although star- or tree-structured topologies are typically adopted to reduce this.

In the second approach, a 'sliding window' detector is swept across the image at multiple positions and scales, robust visual features are extracted at each window position, and a window-level object/non-object classifier is evaluated on these, often followed by post processing to merge overlapping duplicate detections. The dense scan of the possible object locations in 3D scale-space makes the detector resistant to scale and position changes, but it is computationally intensive and the final results are critically dependent on the quality of the underlying classifier (particularly its false positive rate, as most of windows tested are negatives). At present, sliding window detectors give higher accuracies than local feature based approaches for most classes [Everingham et al. 2010a], although the differences tend to be less pronounced on highly deformable object classes like cat, dog, *etc*. and on classes with high degrees of intra-class or viewpoint variability such as chair, boat, *etc*. The coarseness of the sampling steps during the scale and position sweep also influences the performance significantly, as do details of the training regime.

Both generative and discriminative approaches have been used to learn the underlying classifiers. Generative classifiers define prior and likelihood models for the appearance of class and non-class instances, deriving the output function indirectly from the likelihood ratio of these, while discriminative models directly model the class/non-class decision given the input features. Although generative approaches have considerable long-term potential for deeper image understanding, the best current object detectors are trained discriminatively.

The overall performance of any detector depends critically on three elements: the feature set; the classifier and learning method; and the training set. Although some detectors based on a single type of feature (*e.g*. Histograms of Oriented Gradients [Dalal

2006]) give good performance, experience suggests that better results can usually be achieved by incorporating several kinds ("channels") of features. In particular, almost all of the best-performing methods in recent PASCAL VOC detection challenges have used several feature channels, combining these with detectors that incorporate multiple aspects (object appearance models) and displaceable parts. Furthermore, although we will not investigate this here, it has recently become popular to incorporate segmentation or contextual cues in either the feature set or in post-processing steps to further improve the performance. A sliding window detector must typically scan tens of thousands of windows per image, so the use of any method that is computationally expensive at the window level makes the complete scanning of the image extremely slow. This is particularly true during training, where multiple iterations over the training set are typically required to include selected "hard" negatives (there being too many negative windows to include all of them explicitly in the initial training set) and resolve aspect choices and part filters and positions.

1.3 Main Contributions

The goal of this thesis is to develop better practical methods for detecting common object classes in real world images. The focuses are feature sets, dimensionality reduction and sparsity. Our sliding window detector combines three powerful feature sets, Local Binary Patterns (LBP) [Ojala et al. 1996], Local Ternary Patterns (LTP) [Tan and Triggs 2010], and Histograms of Oriented Gradients (HOG) [Dalal and Triggs 2005], with [Felzenszwalb et al. 2009]'s Latent SVM approach incorporating multiple aspects and parts. Our combined feature set gives excellent accuracy: our root-only detectors often outperform state-of-the art root plus parts detectors with competing feature sets and our root plus parts detectors have even higher accuracies. However feature sets of such high dimension are slow to train and use so we also study two discriminative dimensionality reduction techniques. The first, Partial Least Squares (PLS), allows detectors to be trained more rapidly with no corresponding loss of run time speed or accuracy. The second, a feature selection method based on SVM weight truncation, allows the size of the active feature set to be reduced by an order of magnitude with a corresponding gain in speed and little or no loss in accuracy. We also study several methods for including a limited set of nonlinearities in the detectors without the prohibitive loss of speed associated with kernel methods. Finally, in an appendix we describe work in progress on a generalization of LBP/LTP features that we call Local Quantized Patterns (LQP). This uses lookup table based vector quantization to provide local pattern style micro-local appearance codings that have the speed of LBP/LTP and some of the flexibility and power of traditional visual word representations.

Our main contributions can be summarized as follows:

- We pioneer the use of Local Ternary Pattern (LTP) [Tan and Triggs 2010] features for object detection and show that they give better performance than either HOG or LBP alone. Moreover, we show that HOG, LBP and LTP complement one another, so that an extended feature set incorporating all three of them gives very good performance.
- We study Partial Least Squares (PLS) as a discriminative dimensionality reduction method to speed the training of linear detectors. The resulting method is as fast and accurate as a standard linear detector at run time, but much faster to train. We also present some nonlinear classifiers based on PLS that have slightly better accuracies than the corresponding linear ones without much loss of speed.
- We show that for our detectors, a simple SVM weight pruning heuristic allows the active feature set to be reduced in size by an order of magnitude with little or no loss in accuracy, and sometimes even a gain. We perform a detailed comparison that shows the advantages of our approach relative to other existing feature selection methods.
- We show that all of the above techniques integrate well with the state of the art multi-aspect/multi-part Latent SVM approach of Felzenszwalb et al. [2009].
- We present an extensive experimental evaluation of the influence of the various components and parameters of these methods, and study their performance relative to other existing approaches on a selection of challenging publicly available datasets.
- We present Local Quantized Patterns (LQP), an enriched form of Local Pattern features that combines some of the advantages and flexibility of the local pattern and visual codebook frameworks.

Overall, these contributions lead to significant improvements in detector performance, giving state-of-the-art results on several benchmark datasets.

1.4 Thesis Organization

The remainder of the thesis is organized as follows:

- **Chapter 2** reviews the state of the art on object detection. After a brief survey of the different types of object detectors, we present each main component of the sliding window approach in more detail.
- **Chapter 3** presents more details on the SVM training mechanisms, and the protocols that we use for the detector evaluation.

- **Chapter 4** details the three feature sets that we use for object detection HOG, LBP and LTP, and provides a study of the parameter settings used for both the individual descriptor channels and combinations of descriptors.
- **Chapter 5** describes our feature projection and selection methods, and our approaches to developing fast nonlinear classifiers.
- **Chapter 6** presents the results of a detailed experimental study of our detectors on each of the datasets that we use for detector training and evaluation, comparing them to other state of the art methods.
- **Chapter 7** presents our main conclusions and discusses some possible directions for further work.
- **Appendix A** presents LQP, our fast vector quantized version of local pattern features, and gives some preliminary experimental results on this.

Chapter 2: State of the Art

Research in visual object detection has gathered considerable momentum in the past decade because it has finally reached the stage where many practical applications appear to be within reach. Object detectors have to cope with the effects of varying illumination, cluttered backgrounds, and widely varying image positions and scales. To adapt to these difficulties, researchers have developed increasingly informative descriptors and powerful classifiers and training mechanisms. Currently, the major factors driving research are, how to further improve the accuracy, reliability and computational efficiency of the detectors (both during training and evaluation), and how to learn deeper image models from less reliable or more weakly annotated training data. Improved accuracy requires more powerful feature sets, classifier functions and training mechanisms, while improved speed requires features and classifiers that are more efficient to evaluate and methods to ensure that they are only evaluated at the necessary places. Although these goals overlap, they can still be addressed somewhat independently.

Current object detection systems can be divided into two broad classes based on the way that they approach the problem:

- **Bottom-up** approaches see objects as collections of parts or local appearance fragments. They first use relatively generic visual features to find possible locations for object parts, then combine these using geometric or statistical information about part layout to hypothesize and evaluate possible object locations in the image. Highly object specific and/or structural information is thus evaluated only at a relatively sparse set of possible object locations.

- **Top-down** approaches view an object as a complete body, perhaps with parts as sub-elements but always with a single large-scale "part" representing the entire body. Such approaches typically use dense multiscale scanning of one or more highly class specific templates across an image pyramid to detect object instances and their extents.

The distinction between bottom-up and top-down approaches is not sharp. In particular, recent top-down approaches often include subparts to better model object shape deformations and partial occlusions, and on the other hand recent bottom-up approaches

use coordinates relative to an object-level detection window to train and evaluate their part detectors.

Irrespective of the approach employed, object detectors are critically dependent on their visual descriptors and classifiers: the descriptors must distill the information needed to detect the class into a readily usable form, which the classifier uses to make the actual decision at the window level. Post-processing is often also included, *e.g.* to remove multiple overlapping detections of the same object or to enforce contextual constraints. Detectors can thus be classified according to their choices of feature set, classifier architecture and post-processor.

This chapter provides a brief overview of current object detection approaches, particularly focusing on the various components of top-down detectors including feature sets, classifiers and training, and post-processing methods, for increasing the computational efficiency of the detector.

2.1 Classifier Architecture

2.1.1 Bottom-Up Approaches

Bottom-up approaches represent objects as collections of visually detectable parts or fragments. They typically use an array of detectors to find sparse sets of candidate positions for a diverse set of atomic object parts. Here "part" simply means a distinctive and recurring visual pattern that is closely associated with the object, not necessarily a semantic body part like leg, torso, forearm, *etc.* Given the set of candidate part detections, the detector attempts to use information about the plausible relative locations of parts to assemble the part detections into coherent object detections. To detect the parts, either purpose-built detectors are developed for specific classes of parts (*e.g.* for human limbs, arms, torsos, *etc.*) [Mohan et al. 2001; Ioffe and Forsyth 2001; Ronfard et al. 2002; Mikolajczyk et al. 2004; Felzenszwalb and Huttenlocher 2005; Dalal 2006; Wu and Nevatia 2007; Dollàr et al. 2008; Gall and Lempitsky 2009; Lin et al. 2009; Schnitzspan et al. 2009, 2010] or a sparse set of distinctive visual patterns that are statistically associated with the object is found, *e.g.* by some form of data mining [Agarwal et al. 2004; Shotton et al. 2005; Opelt et al. 2006; Leibe et al. 2008; Fergus et al. 2003, 2005; Bouchard and Triggs 2005; Maji and Malik 2009; Vedaldi et al. 2009; Razavi et al. 2011]. Many of the distinctive pattern based detectors use descriptors over generic interest regions [Lowe 2004; Kadir and Brady 2001; Förstner 1987] or edge/boundary fragments [Shotton et al. 2005; Opelt et al. 2006] within a visual codebook learning framework to find discriminative patterns for use as parts, subsequently learning statistical parts models for their relative locations to produce a constellation style object model [Agarwal et al. 2004; Shotton et al. 2005; Opelt et al. 2006; Leibe et al. 2008]. Using salient patterns

of this kind provides invariance to small deformations, scale, orientation and viewpoint changes, thus increasing the robustness of the detector. Conversely, most of the detectors based on specific classes of parts use either template based detectors [Mohan et al. 2001; Dalal 2006; Wu and Nevatia 2009; Schnitzspan et al. 2010] or densely sample patches from the scale-space pyramid [Gall and Lempitsky 2009; Barinova et al. 2010] to locate the positions and scales of the parts.

In addition to the differences in their parts detectors, bottom-up approaches adopt several different methods of combining parts. The Constellation model of Fergus et al. [2003] uses a joint Gaussian distribution over fully-connected parts. Many authors, including [Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2000; Ronfard et al. 2002], use articulated kinematic tree structures. [Shotton et al. 2005; Opelt et al. 2006; Leibe et al. 2008; Maji and Malik 2009; Gall and Lempitsky 2009] use star structured graphical models, and [Agarwal et al. 2004; Schnitzspan et al. 2010] use pairwise connections among part end points. Finally, some methods simply use the responses of the part detectors as input to a second layer of training that infers the object location without directly enforcing any within-parts relationships [Mohan et al. 2001; Dalal 2006].

Bottom-up detectors are popular because they potentially allow partial occlusions and unexpected variations in object pose to be handled. One might also hope that they gain in robustness and reliability because they combine multiple sources of evidence to generate the final object hypothesis. Unfortunately, their focus on local parts tends to give them a rather myopic view of the image and they turn out to be sensitive to the individual part detector responses and often combinatorially more complex than top-down approaches. As a result, they are less accurate than top-down approaches in current detector evaluations. Their main weaknesses are the relative unreliability of small part detectors as compared to larger whole-object ones, and the complexity of the spatial reasoning that is needed to efficiently combine sets of unreliable part detections that include many misses and false alarms.

2.1.2 Top-Down Approaches

Top-down approaches make detection decisions mainly on the basis of the appearance of the complete object, although recent methods often use features associated with implicit parts as a component of this. Whole objects are typically too complex to generate reliable responses from generic feature detectors (interest point, *etc.*), so top-down approaches usually scan a dedicated object detection window densely across the image at multiple positions and scales [Papageorgiou and Poggio 2000; Viola and Jones 2004; Dalal and Triggs 2005; Zhu et al. 2006; Felzenszwalb et al. 2009; Zhu et al. 2010]. The simplest methods of this type are essentially variants (with modern feature sets and classifiers) of

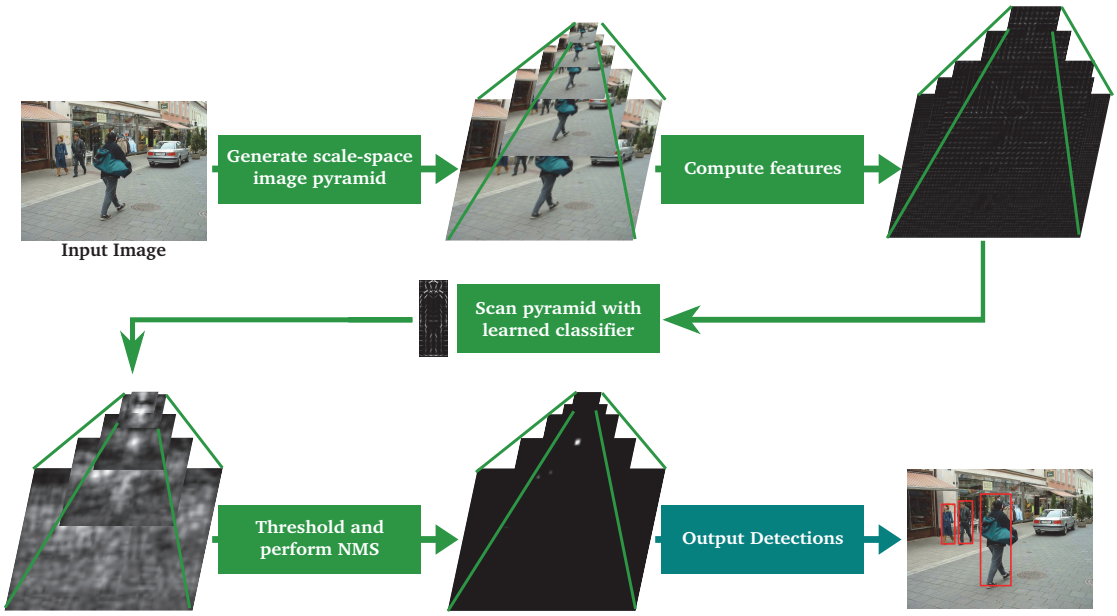


Figure 2.1: *An overview of the sliding window approach.*

the old idea of scanning a rigid object-shaped template over the image and hence are commonly known as sliding window detectors.

Figure 2.1 sketches the architecture of a typical basic sliding window detector. A vector of highly discriminative local visual features is calculated at each location of an image pyramid over the input image. A rectangular detection window is then scanned more or less densely over the feature pyramid, at each location evaluating an object/non-object classifier using the features in the window. The resulting pyramid of classifier scores is then post-processed with thresholding and non-maximum suppression to produce the final detections. The window level classifier is essentially a modern variant of the traditional rigid object template.

Although basic top-down detectors have much simpler architectures than bottom-up approaches, they give better performances in many realistic scenarios where the object class being detected is comparatively "rigid" and unoccluded. More complex top-down detectors implicitly encode spatial layout using dynamic programing style best-candidate search for object parts or sub-regions [Felzenszwalb et al. 2009; Zhu et al. 2010; Ott and Everingham 2011].

2.2 Descriptors

Visual descriptors are a basic component of any object detection system. They must capture sufficient information to distinguish class instances from non-class ones while being resistant to photometric and geometric changes. People have tried many different kinds of descriptors including: edges and gradients [Gavrila and Philomin 1999; Dalal

and Triggs 2005; Felzenszwalb et al. 2009], convolutional net filters [LeCun et al. 1998; Garcia and Delakis 2004], wavelets and Haar rectangles [Mohan et al. 2001; Viola and Jones 2004; Papageorgiou and Poggio 2000; Schneiderman and Kanade 2004; Dollàr et al. 2008], Bag-of-Words descriptors [Blaschko and Lampert 2008; Harzallah et al. 2009], feature covariance matrices [Tuzel et al. 2006, 2008], local pattern texture features [Wang et al. 2009], *etc*.

Early approaches were often based on rigid object-level templates – or in the case of Gavrila and Philomin [1999], hierarchies of templates – that directly used pixel-level features, but more recent approaches typically interpose some form of local spatial pooling between the pixels and the template matching to provide greater resistance to small spatial deformations of the object. For example, Scale Invariant Feature Transform (SIFT) [Lowe 2004] and Histogram of Oriented Gradient (HOG) descriptors [Dalal and Triggs 2005] are based on histogramming pixel-level oriented gradient descriptors into the cells of a regular spatial grid, and recent Local Binary Pattern (LBP) descriptors [Wang et al. 2009] do the same with pixel-level local pattern texture codes. HOG and LBP are currently among the most popular descriptors for object recognition. Their main advantage over simpler features like Haar wavelets is the high discriminative power produced by locally-invariant pooling of features with high spatial resolution, fine orientation resolution and strong resistance to illumination variations. We will present HOG and LBP in detail in Chapter 4.

Another common family of visual representations is Bag of Words (BOW) approaches, where local appearance is characterized by histograms of vector quantized local image patch descriptors ("visual words"). For example, [Blaschko and Lampert 2008; Lampert et al. 2008] use vector quantized Speeded Up Robust Feature (SURF) descriptors [Bay et al. 2008] computed at points sampled from a dense regular grid, and/or from randomly selected or salient image locations. Other approaches such as [Harzallah et al. 2009; Vedaldi et al. 2009] use BOW enriched with multi-resolution spatial information [Lazebnik et al. 2006; Bosch et al. 2007] to capture both the local object appearance and its spatial layout.

Many recent methods enhance their performance by combining several feature sets using either simple concatenation of window-level descriptor vectors, or combinations of set-level kernels with weights chosen using multiple kernel learning. For example, Schwartz et al. [2009] combine multi-level HOG [Zhu et al. 2006], color histogram and texture co-occurrence [Haralick 1979] features. Wang et al. [2009] use HOG and LBP features. Harzallah et al. [2009] combine HOG with SIFT based Bag-of-Word histograms computed over a 3-level spatial pyramid [Lazebnik et al. 2006]. Vedaldi et al. [2009] use multiple kernel learning to combine six feature sets including Bags-of-Words over SIFT descriptors sampled both densely and at interest points, self-similarity descriptors [Shechtman and Irani 2007] and edge based descriptors.

2.3 Classifiers

The window-level classifier uses the given visual descriptors to decide whether an object class instance is present in the given input region or window. Typically, classifiers need to be trained using a large set of manually annotated training examples. Various factors govern the choice of the classifier architecture including the nature and dimensionality of the input feature set, the size of the training set, speed considerations, the kind of output needed (binary or multiclass classification) and the discriminative power required for the task. The classifiers used for object detection range from decision stumps over simple rectangular features [Viola and Jones 2004] to convolutional neural networks [LeCun et al. 1998; Garcia and Delakis 2004] and Support Vector Machines (SVMs) [Papageorgiou and Poggio 2000; Dalal and Triggs 2005; Felzenszwalb et al. 2009].

Bottom-up approaches can be trained either discriminatively or generatively. For example, Opelt et al. [2006] use groups of boundary fragments as weak classifiers in a discriminant AdaBoost learning framework while [Dollàr et al. 2008; Lin et al. 2009] use boosting in a Multiple Instance Learning framework to train both their component part detectors and their complete object detectors. Conversely, Fergus et al. [2003] use a generative model where the part appearance, shape and location parameters are estimated using Expectation-Minimization, and likelihood ratios are used for discrimination. Likewise, Schnitzspan et al. [2010] model part locations, extents, *etc*. as hidden variables in a graphical model and use EM to estimate maximum likelihood part configurations.

Current top-down approaches tend to be trained discriminatively. The two most popular choices are SVMs [Cortes and Vapnik 1995; Joachims 1999; Schölkopf and Smola 2002] – which can use either linear or kernel representations – and Boosting [Freund and Schapire 1996; Schapire and Singer 1999]. However even in these approaches some generative and bottom-up ideas are used. In particular, the discriminatively trained latent part positions of Latent SVM [Felzenszwalb et al. 2009] are analogous to the maximum likelihood part positions obtained when instantiating generative approaches.

Support Vector Machines (SVM)

Support Vector Machines are a form of linear discriminant classifier that provides good generalization by maximizing the margin – the separation between the boundaries of positive and negative classes – under a norm constraint on the weight vector. Like most other linear classifiers, they can either be used directly in linear form (the classifier score is an affine function of the descriptor vector), or kernelized. Kernel SVMs usually give somewhat better accuracy than linear ones, but they are much slower to run, and training is often even slower if it requires the tuning of kernel hyperparameters via

cross-validation.

Papageorgiou and Poggio [2000] and Mohan et al. [2001] use overcomplete dictionaries of Haar wavelets to train quadratic kernel SVM based object detectors. However modern detectors typically use linear SVMs for speed, compensating for the loss of accuracy by adopting extended feature sets. Dalal and Triggs [2005] use linear SVM over HOG features. Similarly, Wang et al. [2009] use linear SVM over combined HOG and LBP features in person detectors that also include explicit occlusion modelling. Blaschko and Lampert [2008] cast the object detection problem as a structured learning one, learning a mapping from the 2D image to 4D object localization box coordinates using a generalized SVM algorithm.

Felzenszwalb et al. [2009] introduced Latent SVM, an enhanced form of linear SVM that incorporates latent variables representing the exact positions and scales of the detector root template (which is no longer required to coincide exactly with the training annotation) and its subparts (if present), and that also has the ability to choose the best of several alternative representations ('aspects') for each window. The main advantages of this framework are that it allows sharper individual templates to be learned despite small variations of the annotation windows in the training set, and that it provides a natural extension to more flexible object models that include latent aspects and parts. We will describe the Latent SVM architecture and training process in more detail in Chapter 3. Similarly Zhu et al. [2010] use latent structural SVM learning to propose an efficient method for training three layer hierarchical models.

Other recent methods either use cascades of linear and nonlinear classifiers for improved speed and performance, or propose other optimizations that allow nonlinear SVM detectors to be speeded up. For example, Harzallah et al. [2009] use a cascade of linear and Chi-squared kernel SVMs for their object detectors. Vedaldi et al. [2009] use Multiple Kernel Learning (MKL) to train cascades of linear, quasi-linear and nonlinear classifiers. The MKL algorithm finds the optimal linear combination of an underlying set of simple kernels for the given problem. Vedaldi and Zisserman [2010] expand additive kernels (Intersection, Hellinger's, and Chi-squared) in terms of dot products of nonlinear basis functions of the features, thus allowing them to approximate the kernels more efficiently. Similar approximations for additive kernels were proposed by Maji et al. [2008]; Maji and Berg [2009].

Boosted Classifiers

Boosting is a stepwise greedy method for building a strong ensemble classifier from a pool of much weaker candidate classifiers. At each step it uses an appropriately chosen reweighting of the input data to choose a suitable new weak classifier to add to the ensemble. In practice the weak classifier is often chosen from a comparatively small

random sample of the available candidates. Boosting succeeds whenever weak classifiers can be found that are always at least a little bit better than random. There are a number of different variants, but the original AdaBoost [Freund and Schapire 1996; Schapire and Singer 1999] is probably still the most well-known and often-used variant despite its known sensitivity to outliers.

In object recognition, boosting is often used in association with: (i) decision stumps (thresholded features) over extremely large feature sets – where its greedy stepwise learning process allows the most relevant features/weak classifiers to be selected efficiently; (ii) cascade-style detectors that prune out negatives progressively while descending a cascade of increasingly complex and competent window-level classifiers [Viola and Jones 2004] – where efficient pruning requires that the classifiers early in the cascade should be as fast as possible and hence use very few features/weak classifiers. Since Viola and Jones, AdaBoost and its variants have been used extensively, both to select subsets of discriminant features and to train detectors. For instance, Laptev [2009] uses Fisher Linear Discriminants over vector valued HOG features as weak classifiers in his AdaBoost object detector. Torralba et al. [2007] use GentleBoost over regression stumps in their multiclass object detection framework, which tailors the classifiers to share features that are useful across several object categories.

Although boosting is well suited to both feature selection and classifier training over high cardinality feature sets such as overcomplete Haar wavelets, it rarely equals the performance of linear SVM over a given, manageably-sized set of features/weak classifiers, and it is also much slower to train in this case. For this reason, for feature sets of large but manageable dimensionality such as ours, SVM and its nonlinear variants remain the methods of choice for training object detectors. It is true that – as is also the case with some SVM variants such as Intersection Kernel SVM [Maji et al. 2008] – boosting thresholded component-wise decision stumps over a given set of input features implicitly learns nonlinear classifier – specifically a weighted sum of nonlinear univariate functions of the input features – whereas linear SVMs are restricted to using linear functions. However our experiments below suggest that linear functions are close to optimal in our applications and in practice we have never found an instance where boosting outperforms linear SVM using our features.

Other Classifiers

Some detectors use other classifiers than SVM and Boosting. For instance, [Garcia and Delakis 2004; Osadchy et al. 2007] use convolutional nets to train their face detector, and [Gall and Lempitsky 2009; Barinova et al. 2010] use random forests over densely sampled patches for their part detectors. Schwartz et al. [2009] use Partial Least Squares (PLS) to project out a small number of discriminant dimensions from their extremely

high-dimensional feature vector, then train a Quadratic Discriminant Analysis (QDA) classifier on these reduced dimensions.

2.4 Post-Processing

Integration of Multiple Detections

During the detection process, the classifier is scanned across the image at multiple scales and positions. For a give real class instance, this often produces several candidate detections that overlap closely in scale and space. The goal of the detection fusion or Non-Maximum Suppression (NMS) stage is to integrate or cluster these candidates to produce a single final detection at the appropriate position and scale without suppressing detections associated with other nearby objects. Several different methods of merging detections have been developed. Viola and Jones [2004] group candidate detections into disjoint subsets using an area overlap criterion, then select a single representative detection per subset. The locations and dimensions of the resulting bounding boxes are estimated as the averages over the subsets. Dalal [2006] developed a more robust approach that uses Gaussian kernel mean shift clustering to estimate the true position and scale of the detection. This performs well but it is rather slow and it requires the tuning of multiple parameters. Felzenszwalb et al. [2009] adapted the Viola and Jones [2004] approach to their latent learning framework. Instead of assigning every pair of detection candidates that overlap to the same subset, they only merge candidates that overlap significantly, forming separate subsets for any remaining ones. The candidate with the maximum confidence value in the subset is selected as the final detection.

Contextual Reasoning

A number of recent detectors use cues derived from the region surrounding the object to filter out implausible detections, or even to reduce the number of detector evaluations in some multiclass problems. For instance, rather than fusing the candidate detections for each object class independently using Non-Maximum Suppression, Desai et al. [2009] use the empirical relationships of the classes to themselves, the other classes and the background to learn NMS parameters for each class in a structured learning framework. In the same spirit, Lampert and Blaschko [2008] take detectors for different categories and use learned relationships between the categories to perform post-processing. This generates a dependency graph between the classes that is used to decide which detectors to use together during scanning. Ramanan [2007] uses pre-learned object segmentation models as a post-filter to reject false alarms. Harzallah et al. [2009] describe a probabilistic framework that includes contextual cues from a global scene-type classifier in their

detector.

2.5 Computational Efficiency

To be practical, a detector needs to be computationally efficient in both training and use. Many otherwise-promising methods fail this test. In a typical detection task where objects can occur in arbitrary poses at arbitrary image positions and scales, the detector must scan tens or even hundreds of thousands of candidate windows per image and check them for the presence of object instances, often running several component detectors in each window. Usually only a few of these windows actually contain the object class of interest, whereas detectors based on conventional machine learning classifiers typically consume a fixed amount of processing time per window scanned, irrespective of whether or not it contains the class of interest. To increase the computational efficiency, many methods adopt either a cascade architecture or a coarse-to-fine search strategy or even both.

Cascade approaches are based on the idea of attentional filtering or early rejection of unpromising patterns [Baker and Nayar 1996; Viola and Jones 2004]. A nested chain of classifiers is constructed in such a manner that relatively simple and efficient classifiers are used to reject the bulk of the negative cases in the first few stages of the cascade, while the later stages involve increasingly complex classifiers that are trained to handle the remaining difficult decisions. Besides the many deep boosting-based cascades [Viola and Jones 2004; Zhu et al. 2006], Harzallah et al. [2009] and Vedaldi et al. [2009] use shallow cascades of linear and nonlinear SVM classifiers to train their detectors. Felzenszwalb et al. [2010a] use a cascade architecture to speed up the evaluation of their parts-based multiple component detectors. Specifically, they evaluate the costly deformable parts in a learned hierarchy with a set of learned thresholds to allow the early pruning of implausible hypotheses.

Coarse-to-fine approaches start with coarse-scale features and gradually add finer scale ones, pruning as many of the surviving negatives as possible at each stage. For example, Pedersoli et al. [2010, 2011] increase the scanning speed of their detectors by propagating the partial evaluation score of each detector to the next level down in an image pyramid. The cascade and coarse-to-fine approaches can be combined to further reduce the computational cost, *e.g.* Pedersoli et al. [2011] combine their coarse-to-fine approach with the cascade of Felzenszwalb et al. [2010a].

One can also use a coarse-to-fine strategy for the detector position and scale scan (as opposed to the spatial resolutions of the features used). Lampert et al. [2008] formulate the scanning of the image pyramid as a branch and bound problem. They partition the set of candidate object locations in the pyramid into local subsets, deriving a detector score bound for each subset, and using this to prune unpromising subsets. This approach

is very effective when it can be used, but it requires the existence of good quality and computationally inexpensive bounds, which is not the case (as far as we know) for our features and detectors.

2.6 Relationship to Past Work

This thesis presents a family of sliding-window object detectors based on the frameworks developed by Dalal and Triggs [2005] and Felzenszwalb et al. [2009]. Our original contributions relate mainly to extensions of the feature set and to the introduction of effective dimensionality reduction and sparsification methods. Many recent approaches [Harzallah et al. 2009; Vedaldi et al. 2009; Wang et al. 2009; Schwartz et al. 2009] use extended feature sets that include several different types of features, but we will argue that our LBP+LTP+HOG set offers a particularly good trade-off between discriminative power and computational complexity. Both Dalal and Triggs [2005] and Felzenszwalb et al. [2009] use variants of HOG features for their detectors. Although HOG alone gives near state-of-the-art performance for structural classes like vehicles and people, it performs less well for classes that are characterized more by distinctive texture than by rigid shape, such as many kinds of animals. Following Ahonen and Pietikäinen [2007], Wang et al. [2009] used LBP texture features for human detection. Although this gives performance that is on par with that of HOG for some classes by itself, it does not give consistently good performance across a broader range of classes and datasets. One of the main problems is the fact that LBP is sensitive to noise in near-uniform regions. To counter this, Tan and Triggs [2010] introduced a more discriminant and robust variant of LBP called Local Ternary Patterns (LTP) for a face recognition task. We show that LTP also works well for object recognition (better than HOG in many cases), and that the combination of HOG, LBP and LTP is even stronger at the cost of increased descriptor dimensionality. Although the local pattern features (LBP/LTP) perform well, they remain very local and hence somewhat myopic. To tackle this we introduce Local Quantized Patterns (LQP), a generalization of local pattern features that allows the patterns to have many more pixels and/or quantization levels and a much wider range of geometries than standard hard-coded LBP/LTP. We show that LQP gives better performance than all of the other features that we have tested including HOG, LBP and LTP.

Regarding dimensionality reduction, Schwartz et al. [2009] used Partial Least Squares (PLS) [de Jong 1993; Wold et al. 2001] to find a reduced feature space for their nonlinear classifier. However this requires a multi-dimensional linear projection to the reduced subspace at run time, so it is relatively costly to run. We show something different: that when training linear SVM detectors on high-dimensional feature sets, using PLS to reduce the dimension before SVM training can substantially reduce the overall detector

training time with no loss of accuracy, and that pushing the reduced-space classifiers back through the PLS projection provides a standard linear classifier on the original feature space, so there is no loss of speed at run time.

The other main approach to dimensionality reduction is feature selection or sparse learning. There are many different feature selection methods. In particular, many recent machine learning algorithms use L_1 regularization to obtain sparsity – examples include the L_1 regularized logistic regression and L_2 loss SVM algorithms of [Fan et al. 2008] – but at present these methods prove to be less accurate than classical linear SVMs in our applications. Instead, we show that a very simple and efficient weight-thresholding heuristic gives results that are more accurate than the best current L_1 SVM algorithms. The resulting method greatly reduces the effective dimensionality of the detectors, thus enhancing their speed, and it also improves their accuracy slightly in many cases.

Chapter 3: Training Methods and Evaluation Metrics

We test various types of single- and multi-root and/or multi-part sliding-window detectors. All of these are built by adding a post-processor to a window-level classifier that is learned over a large set of labeled training examples. In addition to the feature set, classifier type and training data, the performance of the detector depends considerably on the underlying training mechanism. This chapter details our two non-latent and latent training mechanisms, our methods of enforcing bilateral symmetry and our post-processing module. These are all based heavily on Felzenszwalb et al. [2009] and Dalal [2006], so the details are given mainly for completeness. The chapter finishes with a discussion of the metrics that we will use to evaluate detector performance in subsequent chapters.

3.1 Training Mechanisms

There are two approaches to using training annotations, direct ("non-latent") and latent. The direct approach is the traditional one. It simply takes the annotation windows, rescales them all to a fixed size, and directly trains a classifier on the resulting window-level feature vectors. This approach is simple and fast to train, but it cannot handle classes whose annotation windows have significant variations in aspect ratio and it does not take account of two key facts: (i) the given training windows are supplied manually so they are unlikely to have the best possible alignment in position and scale with the underlying object, at least if "best" is measured in terms of the preferences of the final detector; (ii) in use, a practical detector evaluates image windows only at a fixed set of position and scale increments and with typical manual annotation procedures, the training windows are unlikely to correspond exactly to one of these. It would be better to use a training procedure that accounts for misalignments of these kinds. The latent approach addresses these issues by sampling potential classifier training windows only at the given increments and relaxing the assumption that the window used must coincide exactly with the given annotation, instead searching for the "best" training

window among all those that overlap the annotation by at least a certain amount. Here, "best" denotes the window that generates the strongest (most positive) response from the current detector. The exact training window positions and scales are thus latent variables that need to be found during training. Moreover, this approach can easily be extended to handle other latent variables such as which of several alternative root detectors to use (potentially with different aspect ratios, *etc.*), and to find the latent positions of additional displaceable object subparts. We will discuss each of these two approaches in turn below.

Furthermore, there are so many potential negative windows in a typical set of negative training images (*i.e.* ones containing no positive instance), and competitive feature sets currently have such high dimension, that it is impossible to include all of the possible negatives as examples in current (live memory, batch) SVM training methods. Moreover, even it was, this would often lead to poor learning owing to the overwhelming predominance of negatives. For this reason it is usual to start with a small initial set of negatives and to search for and include additional "hard negatives" over several rounds of classifier retraining. This is also discussed below.

3.2 Non-latent Training

The direct (non-latent) training method is initialized as follows, prior to the search for hard negatives described below. First the dimensions of the detection window are estimated from the statistics of the training annotations: the positive annotation boxes are sorted by aspect ratio and the detection window is chosen to have their mean aspect ratio, an area covering 80% of them and a resolution covering 80% of them without up-scaling. Next, the positive training set is obtained by extracting, rescaling and cropping all of the annotated object instances to fit the chosen detection window. If preselected negative training windows are not supplied, a small set of initial negative examples is also obtained by randomly sampling windows from the negative image pyramid[1]. The negative set is typically only a few times the size of the positive one to avoid overly biasing the initial detector towards negative rejection at the expense of positive acceptance. The initial version of the window-level classifier is trained using these examples to get a window-scoring function S_ρ incorporating the classifier weights $\boldsymbol{\rho}$. For example for a linear SVM with window-level feature vector $\mathbf{r}$, $S_\rho(\mathbf{r}) = \boldsymbol{\rho} \cdot \mathbf{r}$.

[1] For cell based feature sets like HOG, LTP, *etc.*, we use precomputed feature pyramids for speed. Each level of the image pyramid is tiled regularly with cells and a feature vector is computed for each cell. Provided that the detection windows always align with cell boundaries – which we assume for speed and simplicity given that it gives good results in practice [Dalal 2006; Felzenszwalb et al. 2009] – this allows the feature vector for any given window to be directly assembled from the feature vectors of its cells. To allow partially visible objects at the image border to be detected, we return zero filled histograms for cells that would be beyond the image boundaries. For mirrored pair roots and parts (see below), a binary beyond-boundary indicator is also added to each cell [Felzenszwalb et al. 2010b] – this helps to calibrate the detector responses for cells lying on or beyond the boundary.

3.2.1 Hard negatives

In typical object detection scenarios there are usually only a few class instances per image on average, so negative windows outnumber positive ones by four or more orders of magnitude. It is thus essential for the classifier to be extremely good at suppressing false positives, which typically requires large negative training sets – again four or more orders of magnitude larger than the positive ones. However given the size of current positive training sets and feature vectors, it is impossible to fit so many negatives into RAM at once, so classical batch SVM training on the complete negative set is impossible.

On-line training methods could be used, but empirically we find that even the best and most recent of these are slower and less reliable than batch methods: they have rapid initial convergence but much slower and more erratic end-game convergence and unfortunately, when comparing many competing methods or variants, searching for hard negatives, *etc.*, training to near convergence is essential for reliable results.

Instead we use batch SVM training over several rounds, in each round using the current classifier to search the negative images to find a new set of currently "hard" negative examples, then retraining the classifier on these together with the initial negatives and positives [Sung and Poggio 1998; Osuna et al. 1997; Dalal and Triggs 2005]. For maximum margin classifiers such as SVM, only the negative examples that fall above or near to the negative margin $S_\rho(\mathbf{r}) \geq -1$ have an influence on the final classifier, and if possible all of these should be included as hard negatives (even though only those with $S_\rho \geq 0$ are actually misclassified). It can be shown that this bootstrapping procedure will converge in a finite number of iterations [Felzenszwalb et al. 2009], where the number of iterations required depends on the discriminative power of the features, the size of the learning cache and the size of the dataset. In practice, the search is repeated at most a limited number of times owing to its computational cost. The negative images are scanned exhaustively in a given sequence, and all of the hard negatives that are found are added to the learning cache until the cache is filled. The classifier is then retrained using the cached examples. Afterwards, the examples that have become easy (that fall below the negative margin of new classifier) are deleted from the cache to make room for new hard negatives. This search and retraining process continues until no more hard negatives can be found (*i.e.* all of the negative training images have been scanned) or until it reaches the maximum number of iterations.

3.2.2 SVM Algorithm

By default we trained our non-latent detectors using Dalal's "densified" (converted to a dense vector representation) variant of SVMLight[2] [Joachims 1999; Dalal 2006].

[2] svmlight.joachims.org/

We tested several alternative SVM algorithms including LibLinear [Fan et al. 2008][3], LIBOCAS [Franc and Sonnenburg 2009][4] and the online LaSVM[5] solver of [Bordes et al. 2005]. LibLinear turns out to be about 1.6 times faster than SVMLight for equivalent accuracy, but it uses a sparse representation of feature vectors so it currently (for our dense feature sets) requires twice as much memory as our densified SVMLight, making it impractical for large problems. We only became aware of LIBOCAS after the experiments in this thesis had been run, but it seems to be a useful alternative to SVMLight. It automatically chooses between the sparse and dense representations based on memory usage. It is around 5.7 times faster than SVMLight for the dense representation, and 4 times faster for the sparse representation. This is for double precision feature vectors, whereas SVMLight uses single precision ones – to handle large problems, LIBOCAS would need to be converted to use single precision features, which might make it even faster. The online method LaSVM [Bordes et al. 2005] gave suboptimal results in our preliminary experiments. It proved to be much slower and more erratic than batch methods such as SVMLight, and this was reflected in poor repeatability which made comparative studies difficult. Although SVMLight is an older algorithm, it is exemplary in this respect.

The fact that object detection training sets typically have overwhelmingly more negative examples than positive ones can lead to counterintuitive results, such as the misclassification of all or almost all of the positive training examples at the default SVM threshold b value owing to the loss function being dominated by negatives[6]. This is particularly common during the later stages of training when there are large numbers of hard negatives. This somewhat counterintuitive setting for b is not necessarily a sign that the final detectors will perform poorly at more reasonable threshold settings, but it is a cause for concern because it indicates that the classifiers are typically being trained at points far from those at which they will be used. The problem can be alleviated by increasing the relative penalty (hinge loss slope) for positive-class errors using the model parameter $J = C_+/C_-$. Intuitively, setting J to equal or exceed the ratio of negatives to positives in the core region of the positive class should suffice to put the default training

[3] www.csie.ntu.edu.tw/~cjlin/liblinear/

[4] http://cmp.felk.cvut.cz/~xfrancv/ocas/html

[5] leon.bottou.org/projects/lasvm

[6] As an aside, the whole notion of separation between positive and negative examples is moot given the extent to which hard negatives dominate the overall training process. For example, for a single root HOG detector on the VOC2006 person class with regularization and positive-to-negative weighting parameters chosen to optimize the final detector AP, the ratio of hard negatives to positives in the final training cache is 108 : 1 and with the default threshold b chosen by the SVM software all of the positive training examples are actually misclassified. Even with LBP+LTP+HOG features, the hard negative to positive ratio is still 38 : 1 and 70% of the positive training examples are misclassified with the default threshold. These observations indicate limitations of current representations and feature sets. They are not the result of inappropriate SVM settings. We stress that (even with kernelization *etc.*), current feature sets are not sufficient to separate the positives from the hardest negatives, so irrespective of the SVM settings a large number of errors necessarily arise, even in training.

threshold near the values required for the final detector, but this is not easy to achieve in practice. Practically, setting J to be too large degrades the final detector performance and greatly increases the training time. Moreover, the C and J settings interact so that in principle a grid search is needed to choose the best C, J pair. Luckily, the optimum appears to be rather flat in practice so that over the range of suitable C values the exact J value is not too critical, and for faster training a J value near the lower end of this range is to be preferred.

3.3 Latent Training

As mentioned above, by giving up the idea that training windows must coincide exactly with annotation ones and instead instituting a search for the best possible (according to the current detector) training window near the given annotation, we can build a training method that (i) is better able to deal with annotations of varying aspect ratio and/or imperfect alignment with the underlying object, and (ii) reflects more closely the actual usage of the final detector – notably the fact that it tests only a limited set of window positions and scales that may not coincide exactly with the initial annotations. These factors result in a better detector in return for a training phase that is more computationally intensive owing to the search for the best latent window positions. In this simple case no latent search is needed at run time, so the run time architecture is not changed. However once the latent search framework is in place, it is easily extended to handle a runtime choice between several alternative root detectors (aspects) for each window tested, and a run-time search for the best placement of object sub-parts, thus providing a more flexible detector for complex visual classes in return for latent variable computations at run time.

3.3.1 Latent Support Vector Machines

Latent Support Vector Machines (Latent SVMs) were introduced by Felzenszwalb et al. [2008][7] to train object detectors with displaceable parts. The part positions, and optionally the choice of which representation to use for the window among several possible templates or "aspects", are encoded as latent variables that need to be instantiated at run time. For a given input example $\mathbf{r}$, the method searches over the permissible values of the latent variables to find the ones giving the optimal detector score, *i.e.*

$$S_\rho(\mathbf{r}) = \max_{z \in Z(\mathbf{r})} \rho \,.\, \varpi(\mathbf{r}, z). \tag{3.1}$$

[7] Latent SVM is a form of structured output learning [Tsochantaridis et al. 2006] that is closely related to multiple instance learning [Andrews et al. 2002].

Here: $S_\rho(\mathbf{r})$ is the SVM scoring function; $Z(\mathbf{r})$ is the domain of the latent variables z; $\boldsymbol{\rho}$ is a vector of learned model parameters including root and part filter weights and part displacement cost weights; and $\varpi(\mathbf{r}, z)$ is a feature vector composed of root, part and part-displacement features extracted from the feature-pyramid at their respective (root and part) scales and positions given the latent assignment z.

During training, Latent SVM uses the $S_\rho(\mathbf{r}_i)$ to estimate the best model parameters $\boldsymbol{\rho}^*$ from the labeled training set $D = \{(\mathbf{r}_1, y_1), (\mathbf{r}_2, y_2), \ldots, (\mathbf{r}_n, y_n)\}$, where each $(\mathbf{r}_i, y_i)$ is an example region and its class label. Formally, it minimizes the objective function

$$L_D(\boldsymbol{\rho}) = \|\boldsymbol{\rho}\|^2 + C \sum_{i=1}^{n} \max\left(0,\, 1 - y_i\, S_\rho(\mathbf{r}_i)\right) \tag{3.2}$$

where C is a standard SVM regularization parameter and the max terms are hinge losses. Note that $S_\rho(\mathbf{r})$, which was simply the linear function $\boldsymbol{\rho}.\mathbf{r}$ in the non-latent case, is now convex rather than linear owing to the latent maximization that it contains, so although $L_D(\boldsymbol{\rho})$ is still convex for negative examples ($y_i = -1$), it is now non-convex for positive ones ($y_i = 1$). In practice this causes few problems because the latent variables of positive examples tend to stabilize during the first few latent training iterations and thereafter remain almost fixed, resulting in effective convexity. However to ensure orderly convergence it is wise to limit the movement of the positive latent variables during the first few iterations. Given the latent assignments, $L_D(\boldsymbol{\rho})$ becomes a standard SVM hinge loss, so any SVM solver can be used to minimize it.

3.3.2 Stages of Latent SVM Training

In practice, latent detectors are trained using an EM-like alternation. In each iteration, the current estimate of $\boldsymbol{\rho}$ is used both to search for hard negatives and to find the best latent variable assignment for each included training example, then these assignments are used to relearn $\boldsymbol{\rho}$ using a standard SVM algorithm. The process is repeated to convergence. More generally, the training of a generic Latent SVM detector can be partitioned into four stages, as follows – further details can be found in [Felzenszwalb et al. 2009]:

Initialization: Latent detectors are initialized in a similar way to non-latent ones. For single root detectors, the dimensions of the detection window are estimated from the statistics of the annotation boxes and a non-latent single root detector is initialized from the positive windows and the sampled negative ones. The detection window dimensions are chosen using the same procedure as in the non-latent case. For multi-root detectors, the positive examples are divided into a number of equal groups by their bounding box aspect ratios, and a separate root is initialized on each group. (This is only a

crude heuristic that captures some of the variability of the training set – if detailed supplementary annotations such as left-facing, front-facing, *etc.*, are available they can also be used for initialization, but we will not test this here).

Stage II: The root(s) are then refined by alternation, finding the best root (if there are several) and root position for each training example and updating the corresponding root filter weights by SVM training. For a given example, all root window positions that overlap the annotation box by at least 70% normalized area overlap are tested, and the root and window giving the best SVM score is taken as the example's candidate for SVM training. This procedure is repeated several times using the complete set of positives and the initial set of negatives, until the root latent variable stabilize.

Stage III: If the final detector will include parts, these are initialized at twice the resolution of the root features (i.e one octave down in the feature pyramid) to capture finer details of the object class. Given the desired number of parts and a pool of possible part shapes, the training algorithm greedily selects the part shape and position that contains the most energy (sum of squared SVM weights) in the root template, removes these weights, then proceeds to select the next part. The part filters are initialized by upsampling the corresponding root filter weights. The parts are attached to the root in a star topology via quadratic penalties on their relative displacements (d_x, d_y) from their ideal locations. This is achieved by adding displacement features $\mathbf{d} = (d_x, d_y, d_x^2, d_y^2)$ to the SVM feature vector along with the part appearance features. The full SVM feature vector thus includes all of the root appearance, part appearance and part displacement features of the best latent configuration found. Once the parts have been initialized, the Stage II optimization of latent variables on the initial training set is repeated for the root and parts detector.

Stage IV: Finally, the Stage II optimization is repeated using the complete set of negative images and searching for hard negatives to augment the negative SVM training set.

3.3.3 Discussion

The maximum number of iterations for each stage is set empirically, but at each stage the process is stopped early if the latent labelings of the positive examples do not change and if all of the hard negatives that are found already belong to the learning cache (*i.e.* if there are no hard negatives left in the training images that would impact the detector).

Besides its ability to train multiple roots and part based models, latent training leads to significant improvements in precision even for single root detectors. For instance,

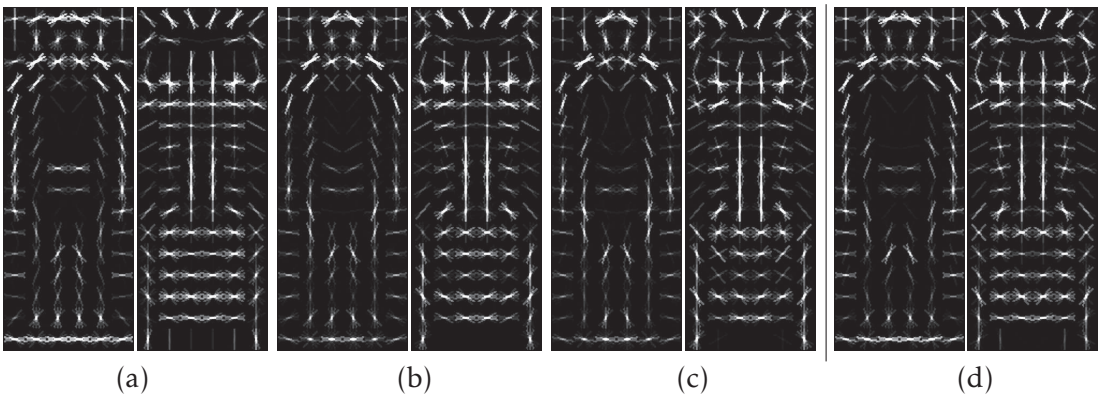


Figure 3.1: *Images of the positive and negative SVM weights at various stages of latent learning for detectors based on HOG features on the INRIA Person dataset: (a) Stage I; (b) Stage II; (c) Stage IV; and (d) the corresponding non-latent detector. The plotting method of [Dalal 2006; Felzenszwalb et al. 2009] is used in this and subsequent HOG-based SVM weight plots.*

for HOG features on the INRIA Person dataset, latent training increases the Average Precision (AP) of the single root detector from 74% to 79%. Figure 3.1 shows an example of models learned during the different stages of latent learning. Note especially the refinement of the head and shoulder region from Stage I to Stage II, which helps to explain why adding latency can improve the performance so much. The final latent template is sharper than the corresponding non-latent one, particularly around the shoulders, head and feet (which are the most critical regions for human/non-human discrimination with HOG). More detailed results are given in Chapter 6 – *c.f.* Table 6.1.

For classes where the annotations vary noticeably in scale relative to the objects themselves, searching over both scale and position turns out to be essential for good results. For example, for single root detectors on VOC2006 with HOG_{31} features (see Chapter 4), restricting the latent variable search to not include scale variations decreases the AP from 23.2% to 20.1% for the person class, but has little impact on the AP for the car class.

Latent learning can be quite sensitive to its initialization and it often gives poor results if the initial root filters are too noisy. For this reason, the initial latent iterations are performed using only a limited search range for the positives and (the relatively small) set of initial negatives. The use of a small negative set helps to preserve the balance between positives and negatives and to focus the initial training on capturing positive characteristics. It also leads to faster training given that a relatively large number of iterations is needed for stabilization in this early phase. In fact, bootstrapping from the positive examples alone may be possible in some cases: for the VOC2006 person class, simply initializing the root filter from the average of the features of the cropped positive examples before using it in the complete training cycle reduces the

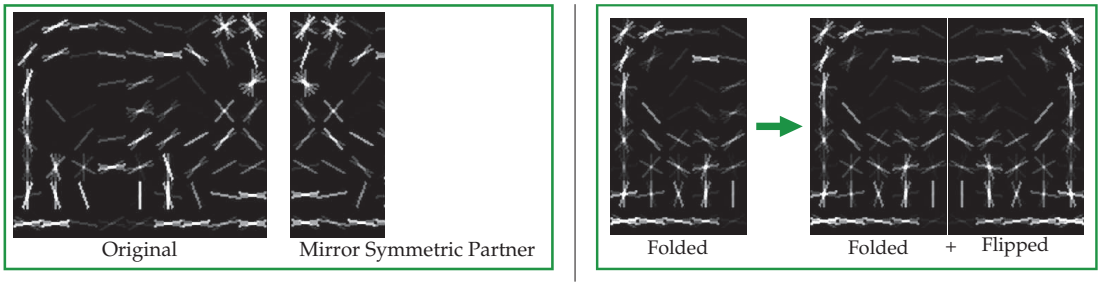


Figure 3.2: *An illustration of our two methods of enforcing bilateral symmetry among components, on the VOC2006 sheep class. The mirrored pair method (left) includes the mirror image of each component as a second competing component, whereas the folding method (right) trains the component using folded features so that the unfolded filter is symmetric.*

final AP by only 1.5%.

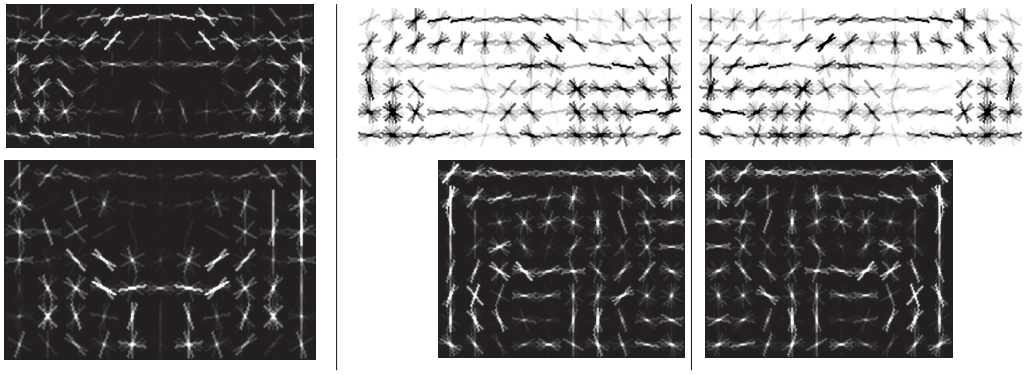
For single root detectors, various optimizations that reduce the overall latent training time are also possible. We pre-calculate and cache the feature vectors for all detector windows having at least 70% overlap with the original annotation boxes in position and scale, as this avoids the recalculation of feature pyramids in each iteration and thus saves a good deal of computation. For single roots there are typically only 8-12 such windows per annotation, whereas for multiple roots or parts there are typically too many to cache. We also tested the possibility of including several positive windows from each annotation box in the positive SVM training set. The hope was that including the best few windows for each annotation would increase the diversity of the limited positive dataset and help to compensate for the imbalance between the numbers of hard negatives and positives. We found that although this method works to some extent, it gives negligible improvement in the overall accuracy.

3.4 Bilateral Symmetry

By default all of our complete latent detectors are forced to be bilaterally symmetric (left-right mirror symmetric). We tested two methods of enforcing symmetry [Felzenszwalb et al. 2008, 2009]. Both use the above training mechanism, but they differ in the way that symmetry is enforced – *c.f.* Figure 3.2. In both cases, training implicitly considers both the original image and its mirror image. Our non-latent detectors are also forced to be symmetric, either by training them on both normal and flipped examples or by using folding (see below).

3.4.1 Folding

The folding method imposes bilateral symmetry on individual components by forcing the root filter to be bilaterally symmetric and the parts (if any) to either appear in mirror



(a) (b)

Figure 3.3: *Examples of VOC2006 car and cow models learned using our two bilateral symmetry methods. (a) Models with single folded roots. (b) Models with a mirrored pair of roots. For both classes, folding causes the loss of some of the characteristic shape information.*

symmetric pairs or to be bilaterally symmetric. For the root filters this is achieved by folding the window and its feature vector in half from left to right and adding the two halves together, so that the learned half-template is bilaterally symmetric when unfolded. This is implemented using a lookup table into the feature vector for each half. The tables work at the cell histogram level. Each entry contains the index of a histogram bin and its mirror symmetric pair, where a gradient orientation (for HOG) or uniform code (for LBP/LTP) is paired with the mirror symmetric orientation or code in the mirror symmetric partner cell. Note that the same effect could be achieved by flipping the window, extracting its feature vector, adding it to the original feature vector and eliminating the redundant (owing to symmetry) entries. Bilaterally symmetric parts are handled in the same manner as the root, while for mirror-symmetric pairs of parts only one member of each pair is learned, by flipping the features of its partner and adding them to its own before training (see below).

Besides enforcing symmetry, folding reduces the feature vector size by a factor of two, making the detector more efficient and allowing more hard negatives to be fitted into memory, thus effectively reducing the number of training iterations.

3.4.2 Mirrored Pairs

Although folding is efficient and it usually gives better results than no enforcement of symmetry, it is also quite restrictive. In particular, for non-mirror-symmetric classes such as sideways facing animals or vehicles, it necessarily confounds left-facing and right-facing examples, thus reducing the discriminative power – *c.f.* Figure 3.3. The second method of enforcing bilateral symmetry simply introduces detector components in pairs, where the second member of each pair is forced to be the mirror image of the first but the components and parts themselves are not required to have any special

symmetry. This doubles the number of components, and feature folding can no longer be used so the feature vectors are longer too. It also requires latent component selection to be run during initialization so that the most appropriate component from the pair can be chosen for the given example. However it is more flexible than the folding method, in particular allowing left-facing and right-facing examples to be separated. In practice, to initialize a mirrored pair of components for a group of examples, positives are partitioned into two subgroups using K-Means under the constraint that an example and its mirror image must belong to different partitions, then a single detector component is initialized on one of the partitions – the filter for the other being implicitly obtained by flipping the initialized one. Once initialized, only one component of each pair is trained: both the original and the mirror-flipped feature vector are supplied for each window and whichever fits the component being trained the best is used.

To allow more uniform comparisons among detectors from the perspective of run times and performance, we will label our detectors with the number of components that they actually use during image scanning, so mirrored pairs are counted as two components. We feel that this designation is fairer because in practice the run time for a mirrored pair is more than twice that for a folded root, and while a mirrored pair can capture left-right asymmetry, two folded roots can capture two different subclasses or aspect ratios – which of the two extensions is more useful depends on the object class. In the tests below we use mirrored pairs by default for multi-component detectors. When folded roots are used, they will be flagged with a subscript "f".

3.5 Post-Processing

3.5.1 Non-Maximum Suppression (NMS)

The Non-Maximum Suppression module post-processes the output of the window-level classifier to generate the final set of detections. Its main role is to fuse repeated detections of the same object that arise because the densely scanned classifier has fired on several adjacent windows that overlap significantly in position and scale, all of which contain the object. The main constraint is that detections caused by the same object should be merged while ones caused by neighbouring objects should not be. We use the iterative greedy NMS heuristic of [Felzenszwalb et al. 2009; Viola and Jones 2004]. The mean-shift based clustering procedure of [Dalal 2006] gives similar results, but it requires several parameters to be set and it is significantly slower. The greedy method works as follows. The bounding boxes of all of the classifier detections in the image are mapped back to ground level image coordinates and sorted by their confidence scores c. The prediction α with the highest confidence score is removed from the list and every prediction β that overlaps it by more than a pre-specified threshold $\frac{area(\alpha \cap \beta)}{area(\alpha)} \geq 50\%$, is

deleted. The process continues recursively until no new predictions can be found, and the list of the surviving predictions α is returned as the final set of candidate detections. The major benefits of this method are that it is relatively simple and fast to evaluate, gives good results and is flexible in the sense that the overlap measure can be tuned to cover different sizes of position and scale neighbourhoods.

3.5.2 Bounding Box Prediction.

For root-only detectors, the object bounding boxes are obtained directly from the detected root locations. However for detectors with parts, we follow [Felzenszwalb et al. 2009] by using linear regression to estimate each of the bounding box coordinates (x_1, y_1, x_2, y_2) from the locations of the detected root and part windows and the width of the detected root window (to provide a scale cue). The regression parameters are learned by running the trained detector on the annotated training data. This method gives a small improvement in accuracy for some of the object classes, with no degradation for the others.

3.6 Evaluation Metrics

For use in later chapters, this section briefly summarizes the metrics that we will use for our detector evaluations. Our detectors use window-level classifiers within a multiscale image sweeping framework, with non-maximum suppression to merge overlapping detections. We can thus evaluate their performance at the level of either the window level classifier or the complete detector. Although the former is fast to run and closely related to classifier training, experience shows that it is not completely reliable as a guide to the performance of the final detector [Dalal 2006; Felzenszwalb et al. 2009], so we use it only as a rough guide to the influence of various feature and classifier parameter settings. Detector-level evaluation is more closely related to the performance achievable in real applications and we use it for all of our full detector evaluations and comparisons.

Currently, the most popular method of summarizing classifier-level performance is Detection Error Tradeoff (DET) curves [Dalal and Triggs 2005], and the most popular means of summarizing detector-level performance are Precision-Recall and False Positive Per Image curves. To facilitate comparison with previous work we use all three of these in our discussions. We describe each of them in detail below. We frequently also use the corresponding Area-Under-Curve figures as simple scalar performance metrics for global comparisons.

To plot any of these curves, the classifier threshold is varied from the lowest possible value to the highest, and the resulting pairs of positive versus negative performance

figures (miss rates, *etc.*) are recorded against each threshold value. In practice this is done by sorting the raw detector scores on the positive and negative test sets into ascending order and counting numbers of scores up to the given threshold. To ensure good coverage without being overwhelmed by negatives, the starting point of our curves is the threshold that detects 95% of the positive training examples.

3.6.1 Detection Error Tradeoff (DET) Plots

DET curves are a variant of Receiver Operating Characteristic (ROC) curves that remain readable at the very low false positive rates that are needed for practical detectors. They plot the Miss Rate (false negatives per positive class member tested) against the False Alarm Rate (false positives per negative class member tested) on a log-log scale. Lower curves (closer to the bottom left) are better.

For classical DET curves, these figures are measured per example (window) tested by the classifier. In a typical object detector tens of thousands of windows are tested per image and very few of these are usually class instances. Conversely, in a complete detector, groups of significantly overlapping positively classified windows are typically merged by a post-processor to produce a single final detection. Given the first point, one typically plots DET using False Positives per Window tested (FPPW) rather than False Alarm Rate (they are almost the same as almost all of the windows tested are negative-class instances), and FPPW's greater than about 10^{-4} are not interesting for practical detectors. The second point makes it problematic to relate counts of classifier hits and misses to counts of detector hits and misses, so detector-level metrics are usually preferable to DET for full detector evaluations. A related problem with FPPW is that test windows typically overlap in both position and scale and each can be sampled more or less finely by different methods, making it difficult to relate per-window performance figures to image level detection rates in the source material.

Another point requiring attention in window-level evaluations is that incorrect processing of pre-cropped examples can easily lead to biases. Both we and Dollár et al. [2009] observed that some of the window-level classifier results on the INRIA Person test set in [Sabzmeydani and Mori 2007] and [Maji et al. 2008] are incorrect, because the classifiers were unintentionally trained and tested on feature sets that contained systematically incorrect feature values at the window boundaries for the positive examples owing to incorrect window cropping. Such effects are less likely to occur in detector-level evaluations as pre-cropped windows are not used.

3.6.2 Precision-Recall Plots

In order to provide performance metrics that relate more closely to their target applications, object detection researchers have adapted Precision-Recall plots from the text

retrieval community. Classically, Precision-Recall is used to measure the effectiveness of document retrieval systems at topic level search tasks. "Precision" is the fraction of the returned documents (detections) that are relevant to the task (correct detections), and "Recall" is the fraction of the relevant documents (images) in the dataset that are returned. The classical definitions can be applied directly to image-level classification and retrieval, but for object-level detection we need to modify them as follows to count object instances not images:

$$\text{Precision} = \frac{\text{Number of objects correctly detected}}{\text{Total number of detections reported}} = \frac{\text{True positives}}{\text{All detections}}$$

$$\text{Recall} = \frac{\text{Number of objects correctly detected}}{\text{Total number of objects in dataset}} = \frac{\text{True positives}}{\text{All true instances}}$$

This brings up the thorny issues of when a detection lies so far from the true position and scale of the object that it should be counted as a false positive with a nearby missed detection rather than as a true detection, and when a real object becomes so small, occluded or truncated that it should be considered to be irrelevant or undetectable. The PASCAL Visual Object Challenge (VOC) guidelines [Everingham et al. 2010b] define standard protocols for object markup and detection overlap when evaluating Precision-Recall curves and Average Precision (AP) metrics and we follow these here. All of the predictions from an image are ranked according to their confidence value, then for each labeled object the prediction with the highest confidence value that overlaps it by at least 50% ($A_o \geq 0.5$) is selected as the true detection and every prediction that is not selected as a true detection is tagged as a false positive. The true and false detections are then sorted according to their confidence scores and the precision and recall metrics for the detections exceeding each given confidence threshold are recorded. The criterion A_o used to measure the overlap of a prediction window B_p with a ground truth window B_{gt} is defined to be the ratio of the area of their intersection to the area of their union:

$$A_o = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}. \tag{3.3}$$

Average-Precision (AP) is an Area-Under-Curve metric used to summarize the overall Precision-Recall performance. It is defined to be the average of the Precision values over the full $[0, 1]$ range of Recall values. In early VOC's it was computed by averaging the majorized precision values $\tilde{P}$ at the eleven uniformly-spaced recall values $R = \{0, 0.1, \ldots, 1\}$:

$$AP = \frac{1}{|R|} \sum_{r \in R} \tilde{P}(r),$$

where $P(r')$ is the observed Precision at Recall r' and $\tilde{P}(r) = \max_{r' \geq r} [P(r')]$ is the maximum precision valued observed over the recall interval $[r, 1]$. Raw empirical Precision-

Recall curves tend be quite jagged so the majorization was introduced to smooth out some of the sampling noise introduced by the finite test sample size. One drawback of this criterion is that it heavily penalizes systems that focus on achieving good precision over only a limited range of Recall values (*e.g.* for applications that prioritize high Precision over full Recall). In 2010, VOC replaced the above 10-sample method with the true area under the curve to achieve more accurate AP estimates, especially for systems with limited Recall ranges. Precision values are sampled densely from the actual system over the complete observed Recall range and the area under the curve is found by numerical integration. To make comparisons with the existing literature simpler and more uniform, we use the new AP computation scheme for VOC2010 and the older one for all of our other results. For multi-class problems, Mean AP – the unweighted mean for the AP values of the different classes – is used to summarize the overall performance.

3.6.3 Recall-False Positive Per Image (Recall-FPPI) Plots

Recall-False Positive Per Image plots are a DET-like variant of Precision-Recall that reports detector-level results over entire images. Specifically they plot the object-level Recall rate against the number of False Positives observed Per Image (FPPI), *i.e.* (number of false positive detections)/(number of images tested). This is only useful for comparing systems head to head on datasets where the average image size and scene type are fixed, as FPPI numbers depend directly on the image size and the density of image content.

Chapter 4: Feature Sets

The choice of feature set is critical for the performance of an object detector: the more representative and discriminative it is, the more reliable classification can be. The feature set must capture the essential similarities between objects of the same class and differences with objects of rival classes despite highly variable object appearance, pose, lighting, clutter, background texture, *etc*. Computational efficiency is also important so simple features are desirable.

Advances in feature sets have been a constant source of progress over the past decade. Most of the early publications used just one feature set [Viola and Jones 2004; Dalal and Triggs 2005; Felzenszwalb et al. 2009], but many researchers now use combinations of features for better results. In such combinations, the more the feature channels complement one another, the better and the more flexible the result will be. Our detectors owe much of their accuracy to the use of a combination of three recent and highly complementary feature sets: Histograms of Oriented Gradients (HOG); Local Binary Patterns (LBP); and Local Ternary Patterns (LTP). Together they provide strong shape and texture cues for object detection.

In this chapter, we give a detailed description of the HOG, LBP and LTP feature sets and their variants and show how they can be combined to produce a basic detector. We finish with a discussion of implementation details and parameter settings.

Before starting, note that the results in this chapter are restricted to single root partless detectors using linear SVM (both non-latent and latent). The restriction to relatively simple classifiers places the focus squarely on the discriminative power of the underlying feature set. To set the parameter values we used a two stage process. First, DET curves (*c.f*. Sec. 3.6.1) on window-level non-latent classifiers were used to quickly select interesting ranges of parameter values for detailed evaluation. Second, the final values were chosen from these sets by running the full detector learning chain (non-latent or latent as the case may be) and comparing Precision-Recall curves (*c.f*. Sec. 3.6.2). We used three datasets: the person class from the INRIA Person dataset [Dalal 2006], and the person and car classes from the PASCAL VOC2006 dataset [Everingham et al. 2006]. Results for the complete detectors on a wider range of datasets will be given in Chapter 6.

4.1 Histograms of Oriented Gradients (HOG)

Histograms of Oriented Gradients (HOG) [Dalal and Triggs 2005] are one of the most successful recent feature sets for visual recognition. Like SIFT [Lowe 2004], HOG is based on the assumption that local image content can be effectively encoded by local distributions of edge directions or intensity gradients, even without recording the precise locations of these. SIFT features were designed for sparse wide-baseline image matching. They combine a multi-scale Difference of Gaussian interest point detector, robust dominant orientation estimation and oriented gradient based local content descriptors. HOG uses a similar descriptor without the multiscale interest point detector and the dominant orientation estimation. Instead, the descriptor is computed on a dense grid of uniformly spaced cells at a single scale, with overlapping local contrast normalization blocks for improved discrimination. HOG has proven to be particularly effective at capturing coarse object shape (contour) information, with strong resistance to illumination variations and some robustness to small spatial variations. Different cell resolutions can be used to capture different levels of information, *e.g.* a large, coarse-resolution cell can be used to capture the overall object shape while smaller and finer-resolution ones capture details of object parts.

Since HOG was introduced, various extensions have been proposed to further enhance its discriminative power. For instance, Zhu et al. [2006] extend it to include variable-sized blocks, Ott and Everingham [2009] use HOG features computed over both edge images and foreground/background segmented detection windows, and Felzenszwalb et al. [2009] include both signed and unsigned gradient information in their HOG cells, using an analytic dimensionality reduction scheme motivated by PCA to reduce the number of features contributed by each cell.

Although HOG is not our main focus here, for completeness we briefly provide some implementation details and test results.

4.1.1 Implementation Details

Our implementation of HOG is similar to that of Felzenszwalb et al. [2009]. The computation involves three main steps: image gradients are computed; the image is divided into a dense grid of rectangular "cells" and a histogram of gradient orientations is computed for each cell; and finally the cells are grouped into small (and typically overlapping) "blocks" and a local contrast normalization is applied to the cell histogram within each block.

Image gradients are computed using $[-1, 0, 1]$ finite difference filters. The gradient orientation at each pixel is quantized (using hard quantization) into n evenly spaced bins in $0 - 360°$ (for signed gradients) or $0 - 180°$ (for unsigned gradients). Each pixel votes into this orientation bin with a vote proportional to its gradient magnitude. For

color images, gradients are computed separately on the R, G and B channels and the channel whose gradient vector has the largest magnitude is quantized. To construct the cell histograms, the image is divided into square cells, and each pixel contributes its vote to the histograms of the 4 nearest cells, using bilinear spatial interpolation to provide soft spatial voting. Cells are then grouped into small blocks of cells and the histogram entries within each block are normalized to L_2 norm 1, after which entries greater than $\gamma = 0.2$ are truncated to γ [Lowe 2004]. To allow faster computation, histograms are not renormalized after truncation and the block-level Gaussian pixel weightings of the original SIFT/HOG are not included [Felzenszwalb et al. 2009].

Our HOG features are typically based on 8×8 pixel cells arranged into 2×2 cell blocks, with unsigned image gradients quantized into 9 orientation bins (evenly spaced over $0 - 180°$) and signed ones into 18 orientations bins (evenly spaced over $0 - 360°$). Every possible 2×2 cell block is taken so each (non-boundary) cell is normalized by four neighbouring blocks leading to four different versions of each cell entry. For unsigned gradients this results in a 9×4 feature matrix or equivalently a 36-D "HOG_{36}" feature vector for each cell.

4.1.1.1 HOG Dimensionality Reduction

If we collect HOG_{36} feature cells from a large corpus of images and use PCA to analyze them [Felzenszwalb et al. 2009], it turns out that almost all of the energy lies in the first 11 PCA components. This suggests that HOG_{36} features could be mapped to an 11-D feature space with little loss of discriminative power. Such a reduction would be slow owing to the 36×11 matrix multiplication required for the projection, but luckily [Felzenszwalb et al. 2009] (*c.f.* Figure 4.1), the PCA components are all approximately constant along either the rows or the columns of the 9×4 HOG feature matrix, which means that they share either the same constant row vector in $\mathbb{R}^4$ or the same constant column vector in $\mathbb{R}^9$. Therefore, we can analytically define a 13-D "HOG_{13}" feature from the 9×4 HOG_{36} matrix by simply summing it along its rows and columns. Projection onto the row space $\mathbb{R}^4$ is achieved by summing the 9 orientation bins for a given normalization block, and projection to the column space $\mathbb{R}^9$ is achieved by summing the 4 differently normalized scores for a given orientation. This reduction is much cheaper than projecting out the top 11 PCA components by explicit matrix multiplication and it gives identical final detector performance.

More generally, this method allows both signed and unsigned gradient information to be captured in a HOG feature cell of manageable size. Dalal [2006] observed that for some object classes signed image gradients give better performance, while for others unsigned ones do. Including signed as well as unsigned gradients would increase the feature dimension by a factor of three. *E.g.* with 9 unsigned orientations, 18 signed ones,

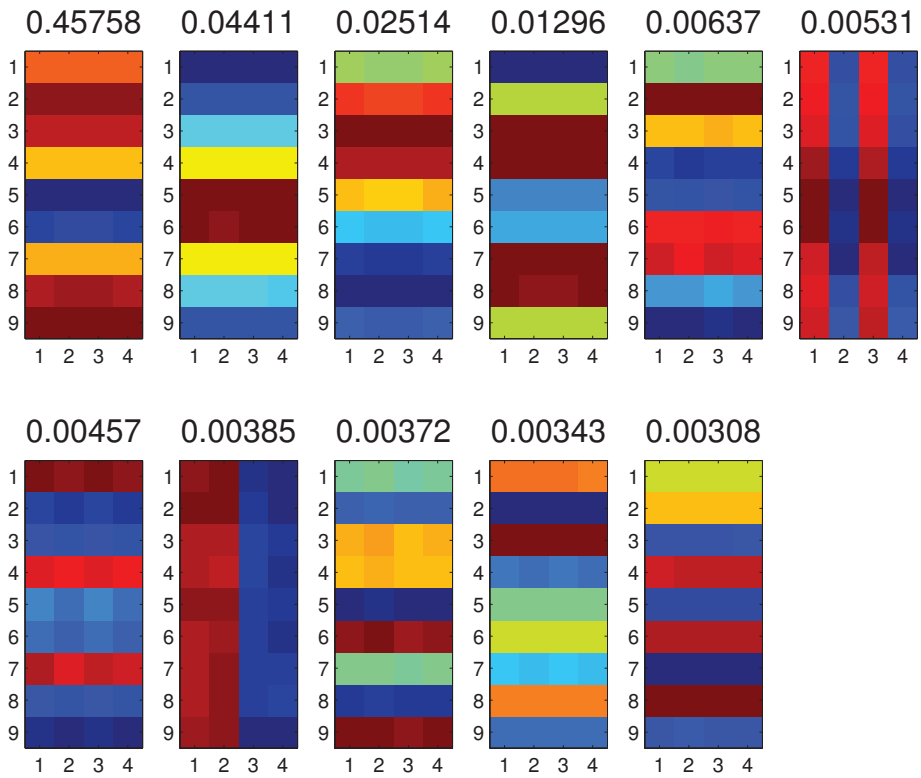


Figure 4.1: *The top 11 Principal Components of HOG_{36} features computed on the VOC2006 training set. The 36-D features are arranged as 9×4 matrices with each row giving the four different block normalizations for one of the 9 orientations. Note the predominantly horizontal and vertical structure of the PCA components. The 12th singular value is five times smaller than the 11th one.*

and four block normalizations, the cell descriptor would be a 27×4 feature matrix with 108 dimensions. This would be prohibitive, but again projecting onto the column and row spaces gives a 27-D orientation vector (summed over all 4 normalizations) and a 4-D normalization vector (summed over all 27 signed and unsigned orientations), and hence a 31-D "HOG_{31}" cell vector. Felzenszwalb et al. [2009] have shown, and we can confirm, that this gives excellent results even though the feature vector is smaller than the original HOG_{36} vector.

4.1.2 Results and Discussion for HOG

We tested the HOG_{36}, HOG_{13} and HOG_{31} variants of HOG, using 8×8 pixel cells with each cell belonging to four overlapping 2×2 cell blocks for L_2 normalization. Table 4.1 shows Average Precisions (APs) for linear Latent SVM detectors trained on these features. For the car class, signed orientation information is known to be useful and HOG_{31} gives the best performance. For the two person classes, unsigned orientation information is the most useful because variations in clothing color often reverse the

HOG Types	INRIA Person	VOC2006 Car	VOC2006 Person
HOG_{36}	79.0	51.7	22.2
HOG_{13}	79.3	50.6	21.0
HOG_{31}	78.7	55.5	23.2

Table 4.1: *Average Precisions for Latent SVM detectors trained on our three variants of HOG features.*

contrasts and all three variants give similar performance. Figure 4.2 shows the learned SVM weights for HOG_{31} features on these classes. These weights give insight into the characteristics needed to accept positives while rejecting negatives. For example for both of the person classes, the positive weights emphasize the head, shoulder, legs and horizontal connections between the feet and the ground while the negative ones suppress both vertical edges that continue into the center of the body (thus helping to eliminate structures such as lamp posts, doors and window borders) and horizontal edges around the torso and leg areas. The learned weights for the VOC2006 person class are significantly "noisier" than those for the INRIA Person class. This is to be expected given that VOC2006 has many annotations of distant people (thus forcing the use of a smaller, lower resolution detection window) and a much wider range of poses including sitting, standing, horse/bicycle riding, *etc*. In the car class, the positive weights capture the overall body contours and the varied gradients of the wheel regions while the negative ones suppress examples with orthogonal edges around the contours and gradients between the wheels. The structure captured by this single-aspect model is somewhat vague because the different object viewpoints must all be represented by a single template. The cell-level organization also makes the shape models rather coarse for all of the classes, but it allows the orientations that best capture the variable layout of the class to be chosen without having to decide whether edges are occluding or internal, what to do with edge junctions, how to threshold, *etc*.

4.2 Local Pattern Features

Texture – characteristics and statistics of local image appearance – is a discriminant property for many object classes, providing complementary information to object shape. There is a large literature on statistical and geometrical texture analysis, including methods ranging from the co-occurrence matrix based method of Haralick [1979] and the Gabor filter based approach of Jain and Farrokhnia [1991] to the Local Binary Patterns of Ojala et al. [1996]. Our texture descriptors are based on the "local pattern"

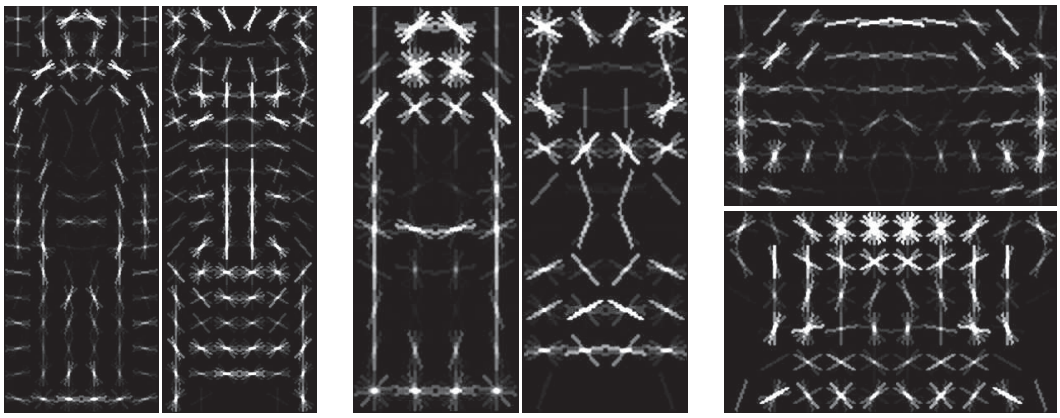


Figure 4.2: *The object shape information encoded by HOG_{31} features. The images display the learned positive and negative weights for single root linear Latent SVM detectors trained respectively on the INRIA Person, and the VOC2006 person and car classes.*

principle of qualitative analysis of micro-local pixel intensity differences. We tested four features of this type: Local Binary Patterns (LBP), Local Ternary Patterns (LTP) and the Center-Symmetric variants of these (CS-LBP/CS-LTP). Moreover, Appendix A describes work in progress on Local Quantized Patterns (LQP), an extension of the local pattern idea based on fast lookup table based vector quantization.

4.2.1 Local Binary Patterns (LBP)

LBP's are computationally simple and highly discriminative descriptors based on the analysis of qualitative micro-local gray level differences. They have proven successful for texture classification [Ojala et al. 1996, 2002] and face recognition [Ahonen et al. 2006] due to their robustness to monotonic grayscale changes, but they were only recently applied to human detection [Wang et al. 2009; Hussain and Triggs 2010]. LBP generates texture-histogram descriptors and, as with HOG, most current methods split the image window into a grid of cells and histogram the descriptors separately within each cell. To compute LBP descriptors, the neighbourhood of each pixel is mapped to a binary code using the LBP feature map, the resulting code values are histogrammed over each cell, and the histograms of the detection window cells are normalized and concatenated into a feature vector that is used as the window level descriptor.

LBP Feature Map. LBP maps each image pixel c to a binary code $\Theta_r^k(c)$ as follows: the gray-levels at k regularly spaced points on a circle of radius r around the central pixel c are sampled, and these samples are thresholded at the gray-level value of the central pixel to generate a k-bit binary string that is used as the code word for the pixel – *c.f.*

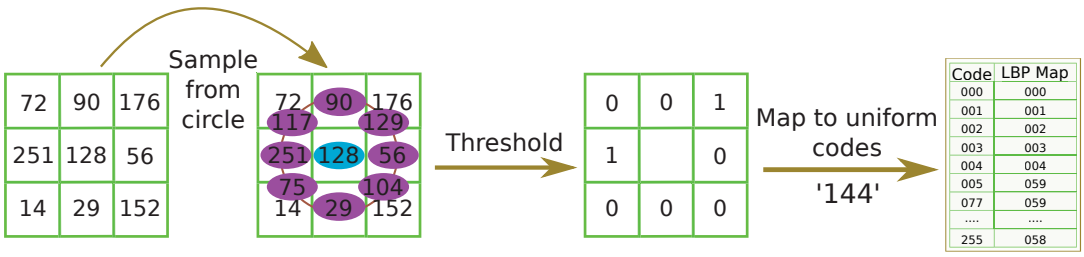


Figure 4.3: *The stages of LBP feature computation. A circle of sample points surrounding the central pixel is evaluated and thresholded* w.r.t. *to the value of the central pixel to produce a binary code, which is then mapped to a uniform code by table lookup.*

Figure 4.3. Formally,

$$\Theta_r^k(c) = \sum_{i=0}^{k-1} 2^i \, \delta(v_i - v_c) \tag{4.1}$$

where i runs over the k sample points around the central pixel c, v_i and v_c are the gray level values at i and c, and $\delta(x) = 1$ if $x > 0$ and 0 otherwise.

A length k binary string gives 2^k possible code values. Originally, Ojala et al. [1996] used the complete set of 256 values obtained by thresholding the 8 neighbours of the central pixel at its gray level value. In our application this would lead to high dimensional feature sets that would make training difficult. However, Ojala et al. [2002] found that most of discriminative information in the histogram is carried by the 58 bins that correspond to so-called *uniform patterns*, while the remaining bins mainly contain "noise". An LBP code obtained from a ring of pixels is called 'uniform' if it contains at most one contiguous group of 1's within the ring. For a length k binary string there are $k(k-1) + 2$ uniform patterns – 58 for an 8 bit string. By taking the bins corresponding to uniform patterns and including one additional bin for all of the non-uniform ones, the number of histogram bins is reduced from 2^k to $k(k-1) + 3$. The uniform patterns turn out to be sufficient to capture important image structures such as corners, edges, *etc.*, and in (low noise) natural images the vast majority of pixels generate uniform patterns. One can also view the uniform pattern coding as a bank of filters that records the co-occurrences of blocks of contiguous orientations.

We will use uniform pattern based codings. However while most other work [Ojala et al. 2002; Ahonen et al. 2006; Tan and Triggs 2010; Wang et al. 2009] uses grayscale based LBP codes, we find that including color information improves the overall accuracy, so we use color image based LBP codes. We tested several different color-spaces and coding methods – see Sec. 4.2.4.2.

Cell Histograms. The pixel-level LBP codes are accumulated over rectangular cells to produce histograms in the same way as for HOG, using bilinear soft spatial voting

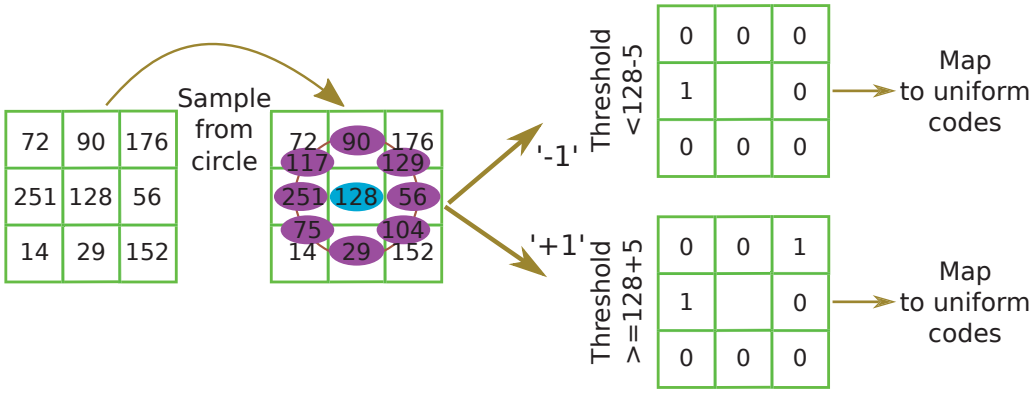


Figure 4.4: *The stages of split LTP feature mapping. Separate "$\geq +\tau$" and "$\leq -\tau$" binary codes are produced* w.r.t. *to the value of central pixel and each is mapped to a uniform code.*

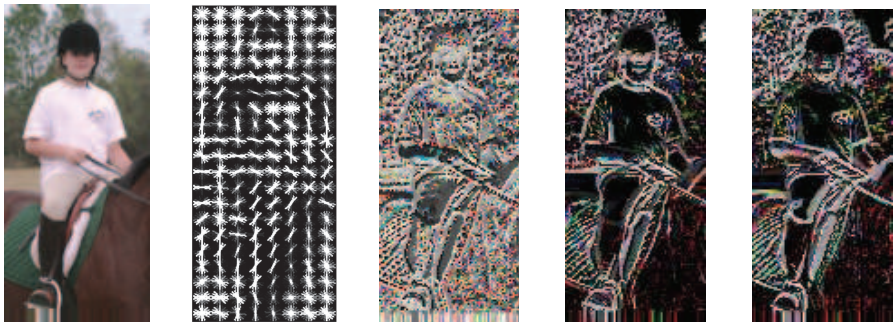
to make the descriptor more robust to small shifts and spatial sampling effects. The contribution of each pixel is thus distributed among the histograms of the four closest cells. Our histograms are normalized to sum one for convenience, but this is cosmetic for local pattern features because (unlike HOG) every histogram has exactly the same total number of counts. However subsequent nonlinear compression of the normalized counts does improve the results – see Sec. 4.2.4.4. Finally, the histograms of the cells in the window are concatenated to form the window-level descriptor.

4.2.2 Local Ternary Patterns (LTP)

LBP is invariant to monotonic gray level changes produced by varying lighting conditions, but it is known [Ahonen and Pietikäinen 2007; Heikkilä et al. 2009] to suffer from noisiness in near-constant image regions owing to the hard thresholding of the local neighbourhood at the central pixel value. Local Ternary Patterns (LTP) is a simple generalization of LBP introduced by Tan and Triggs [2010] that is both considerably more discriminative than LBP, and more robust to noise in uniform regions. Moreover, it can be tuned to extract information that is complementary to LBP. We were the first to test LTP for object detection [Hussain and Triggs 2010].

LTP uses the same sampling structure as LBP, but it replaces the binary codes of LBP with ternary (3-valued) ones by introducing a dead-zone parameter τ and coding the surrounding pixels according to whether their values fall above, within $\pm\tau$, or below the gray-level value of the central pixel – *c.f.* Figure 4.4. Formally, the thresholded value of pixel i is

$$\delta(v_c, v_i, \tau) = \begin{cases} +1, & v_i \geq v_c + \tau \\ 0, & |v_i - v_c| < \tau \\ -1, & v_i \leq v_c - \tau \end{cases} \tag{4.2}$$



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

Figure 4.5: *An illustration of our different feature channels on an input image. (a) An input image. (b) Its HOG descriptor. (c) Its color LBP feature map. (d) Its positive LTP feature map. (e) Its negative LTP feature map. The images suggest that HOG captures mainly coarse object shape and LBP extracts microscopic image texture while LTP captures coarser texture and object contour information.*

The best τ value is selected by testing on validation data.

The total number of codes for an LTP that incorporates k pixel-value comparisons is 3^k. Even with the use of uniform pattern style coding, this number is too high to allow the direct use of LTP codes in cell-histogram feature sets. To work around this, Tan and Triggs [2010] proposed a splitting approach, which we follow. In this scheme the length k ternary code is split into separate "$\geq \tau$" and "$\leq -\tau$" binary length k feature maps as illustrated in Figure 4.4. These are treated as two separate channels of uniform LBP features in the usual way and the resulting histograms are concatenated to form the complete LTP descriptor, which thus has twice the dimension of the corresponding LBP one. Results given in Appendix A suggest that such splitting causes little loss of discriminative power, at least for the datasets studied here.

The introduction of the threshold τ breaks the monotonic illumination invariance of LBP, but it helps to suppress the noise that tends to dominate LBP responses in near-uniform image regions and it provides an additional parameter that can be tuned to extract information complementary to the LBP features. Empirically, across a number of common datasets, a threshold of around $\tau = 5$ gray-levels (out of 255) gives the best performance. Thresholds of between 3 and 10 give very similar results, while larger ones tend to discard too much texture information and smaller ones give descriptors that are too strongly correlated with LBP for complementary.

As Figure 4.5 and the experimental results in Sec. 4.3 suggest, LBP and $\tau = 5$ LTP responses have rather different characters, with LBP capturing mainly dense local texture and LTP putting more emphasis on stronger textures and object contours. Thus even though it is still a local texture descriptor, LTP usefully complements LBP. In fact it would probably be possible to extract multiple levels of texture detail by including

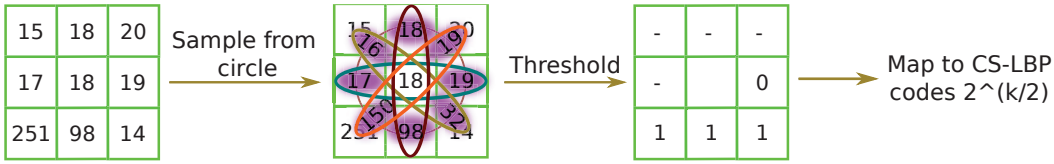


Figure 4.6: *The steps of the CS-LBP feature mapping.*

several different LTP thresholds. For instance a small value such as $\tau = 3$ gray-levels could be used to capture weaker local textures while a higher value such as $\tau = 10$ captures only the stronger ones associated with contours. However each new τ value adds 116 new features to the cell descriptor, which would soon become problematic in applications such as ours. Similarly, attempting to improve the discrimination of LBP/LTP by sampling more pixels (*e.g.* using a larger radius) rapidly increases the number of features per cell, quickly making training infeasible.

4.2.3 Center-Symmetric Local Patterns (CS-LBP/LTP)

Center-symmetric Local Binary Patterns (CS-LBP) were introduced by Heikkilä et al. [2009] to meet a need for more compact region-of-interest descriptors in applications such as wide-baseline matching, image retrieval, *etc*. LBP histograms are comparatively large, so they are costly to use in applications where a great many histogram comparisons are required. CS-LBP is a dimensionality-reduced form of LBP designed to address this. The sampling of neighbouring pixels is identical to LBP, but instead of comparing each pixel in the circle to the value of the central one, diametrically opposite pairs of pixels are compared – see Figure 4.6. This reduces the number of histogram bins from 2^k to $2^{k/2}$. As this is already quite small, no reduction to uniform codes is needed. Formally, the feature map computation is

$$\Theta_r^k(c,\tau) = \sum_{i=0}^{k/2-1} 2^i \, \delta(v_i - v_{i+(k/2)}, \tau) \tag{4.3}$$

where v_i and $v_{i+(k/2)}$ denote a center-symmetric pair of pixels and $\delta(x,\tau) = 1$ if $|x| \geq \tau$ and 0 otherwise[1]. As in LTP, τ is introduced to suppress quantization noise in near uniform image regions. A CS-LBP feature map can be viewed as a special kind of gradient magnitude map as it compares opposite pixels just like a finite difference operator. Here we test CS-LBP for object detection and also extend it to CS-LTP in the obvious way, by replacing the binary code and unsigned δ of Eq. (4.3) with the ternary code and signed δ from Eq. (4.2).

[1] Our definition of CS-LBP differs from that of Heikkilä et al. [2009] in that we use absolute differences between pixels in Eq. (4.3), not simple (signed) ones – otherwise CS-LBP gives much worse results in our experiments.

Note that our work was carried out independently of the use of a CS-LBP variant by Zheng et al. [2010] and it has many differences: instead of using simple CS-LBP codes, Zheng et al. pool the gradient magnitudes of the center-symmetric pairs within the cell, using neighbouring blocks for contrast normalization with Gaussian down-weighting of the peripheral pixels of the block. This HOG-like organization increases the feature dimension by a factor of 4, giving a descriptor with higher dimensionality than standard LBP. To counter this, Zheng et al. [2010] introduce a dimensionality reduction scheme based on the distribution of CS-LBP codes in the dataset. In comparison, our CS-LBP and CS-LTP features are directly analogous to our LBP and LTP ones. We simply record the counts of local pattern codes in the histograms using bilinear spatial interpolation, as with LBP. Again we use color images by default, which increases the performance of the features.

4.2.4 Parameter Settings for Local Pattern Features

For the parameter settings of our local pattern features we present detailed results only for LBP. Similar conclusions hold for LTP, CS-LBP and CS-LTP as they use the same basic structure. Our default settings are as follows. Around each pixel we sample 8 points on circle of 1 pixel radius using bilinear interpolation from the unpreprocessed input image. 59 bin LBP codings are used (58 uniform patterns and one bin for all of the non-uniform ones). The threshold τ is set to 5 gray-levels, and split uniform coding is used for LTP. Histograms are built over 8×8 pixel cells using bilinear spatial interpolation for smoothing. For color images, codes are computed separately on the R, G and B color channels and then pooled into the same histogram. Finally, L_1-Sqrt normalization is applied to each cell histogram *i.e.* the histogram is normalized to sum 1, then square rooted to compress the dynamic range of its bins. We now briefly validate each of these choices.

4.2.4.1 Image Preprocessing

The datasets that we tested have only a limited range of illumination variations and we did not find a preprocessor that was helpful. For example, preprocessing using the method of Tan and Triggs [2010] reduces the object detection performance even though it significantly improves face recognition results for a wide range of methods including ones based on LBP and LTP. More precisely, including gamma compression does not have any significant impact on the performance, while preprocessing the input with a Difference of Gaussian (DoG) filter reduces it significantly. For example, using a DoG with an inner kernel of standard deviation 1 pixel and an outer kernel of standard deviation 3 pixels increases the LBP miss rate on the INRIA Person dataset from 11.2% to 24.8% at 10^{-5} FPPW. It seems likely that the low frequency content and smooth

shading information that the center-surround DoG operation discards is useful for object category recognition, but damaging for the detailed local feature comparisons within the class that are needed for face recognition. Without DoG, the final contrast normalization step of Tan and Triggs [2010] makes no difference to the results as LBP is invariant to this.

4.2.4.2 Color Space

We tested various color models for the local pattern features. Firstly, moving from grayscale to color information by evaluating local pattern codes separately on the R, G and B channels and pooling the results into the same histogram leads to significantly improved performance. For example, using RGB based LBP features on the INRIA Person dataset instead of grayscale ones reduces the miss rate of the window-level classifier from 20.6% to 11.2% at 10^{-5} FPPW. More generally, we tested opponent color spaces as well as RGB ones, in each case either pooling the features for the three color channels into a single histogram ('pooled') or accumulating them into separate histograms to make a feature vector three times as large ('concat'), and we also tested the individual R, G and B channels in isolation.

Motivated by the fact that the human visual system uses red-green and blue-yellow opponency to code color [Wandell 1995], we tested the Opponent Color Space (OCS) models of Wandell [1995] and Sande et al. [2010]. The latter, which converts RGB pixels to OCS ones by a simple linear transformation

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.4082 & 0.4082 & -0.8165 \\ 0.5774 & 0.5774 & 0.5774 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{4.4}$$

performed better in our experiments. The results for LBP features are shown in Table 4.2. On the classes tested, concatenated color predictably outperformed pooled color, which in turn outperformed grayscale. There was no clear winner between the OCS and RGB color spaces, with each performing better on one of the two person classes and mixed results on the car class. The results for the individual RGB color channels are also mixed, with each individual color channel outperforming grayscale on one of the three classes. Indeed, the green channel alone outperforms every other color representation tested on the INRIA Person set. However over a broader set of classes (not shown), we found the results from the individual color channels to be somewhat erratic, so by default we prefer to use representation that incorporates all three color channels. Similarly, we feel that the advantages of OCS are neither large enough nor consistent enough to justify the additional cost of the transformation to the OCS color space as standard practice, and that the dimension of the concatenated representations is too large for their use

Color Space	INRIA Person	VOC2006 Car	VOC2006 Person
Gray	75.3	53.8	18.7
Red	73.4	52.7	18.8
Green	84.3	52.9	17.7
Blue	72.1	54.0	16.4
RGB-Pooled	73.9	54.4	21.6
RGB-Concat	78.0	54.1	22.8
OCS-Pooled	67.4	52.2	24.0
OCS-Concat	72.5	55.4	28.2

Table 4.2: *Average Precisions for LBP based single root Latent SVM detectors trained on grayscale, and RGB and OCS color features.*

to be recommended as general practice given that their overall performance is only slightly better than the pooled ones. We thus use the RGB-pooled representation as the default color coding for all of our local pattern features in the experiments below, while noting that in individual cases better results may be obtainable by using a concatenated representation and/or OCS.

4.2.4.3 Feature map structure

The LBP feature map Θ_r^k could use any geometric arrangement of k sample points within the local neighbourhood of radius r, but a circle of 8 points with a radius of 1 pixel appears to be close to optimal in our application. For example, for 59 bin uniform LBP codes on the INRIA Person set at 10^{-5} FPPW, using a square (the 8-neighbours of the pixel) instead of a circle for sampling increases the miss rate from 11.2% to 14.9% – *c.f.* Figure 4.7(a). Decreasing the number of sampling points k to 6 (33 bin uniform code) increases the miss rate by 4%, while increasing k to 12 (135 bin uniform code) does not change it – *c.f.* Figure 4.7(b). Increasing the radius from $r = 1$ to $r = 2$ pixels increases the miss rate by 9.3% for Θ_2^8 and by 5.2% for Θ_2^{12}. Using complete codes (256 bins) instead of uniform ones (59 bins) does not change the performance.

At the cell level, bilinear interpolation during spatial pooling definitely improves the performance[2] – see Figure 4.7(c). Without it, the Average Precision of LBP on the INRIA Person test set falls from 74% to 69%, although the resulting detector is also about 1.6 times faster. These performance differences remain but become smaller when the local pattern features are combined with HOG, so if speed is critical the local pattern interpolation could perhaps be dropped at the cost of slightly lower accuracy.

[2]Note that the local pattern features in our BMVC paper [Hussain and Triggs 2010] were computed with discrete binning for spatial pooling not bilinear interpolation.

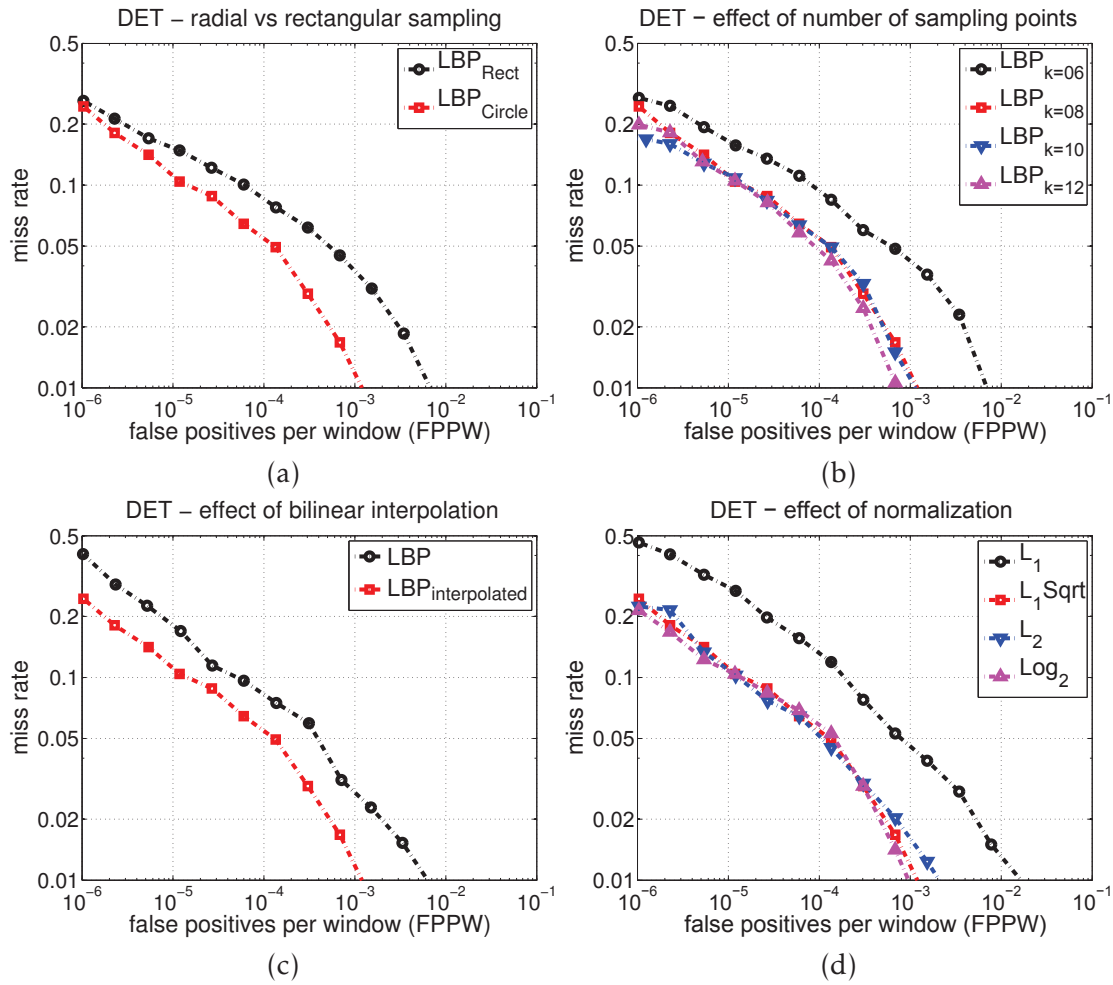


Figure 4.7: *DET curves for several LBP configurations on the INRIA Person dataset. (a) Sampling points on a circle or a square during LBP map computation. (b) Sampling different numbers of points on a circle of radius 1 pixel. (c) Using bilinear interpolation rather than abrupt spatial binning during histogram construction. (d) The effect of different histogram normalization methods.*

A cell size of 8×8 pixels seems to be close to optimal for our applications. Increasing it to 16×16 reduces the dimension of the feature vector by a factor of 4, but it increases the miss rate by 6%.

4.2.4.4 Cell Normalization

Each local pattern cell histogram contains the same total number of counts so L_1 normalization simply rescales the feature vector. However, L_2 normalization or subsequent nonlinear transformations of the bins can change the performance so we tested several different histogram normalization methods. If ϑ_k is the count in the kth bin of the n-bin histogram.

Features	L_1-Sqrt	L_2
LBP	73.9	73.4
LTP	78.9	71.1
HOG	79.0	79.0
LBP+LTP	80.4	72.7
LBP+HOG	80.6	80.6
LTP+HOG	81.3	79.9
LBP+LTP+HOG	82.8	81.6

Table 4.3: *Average Precisions of Latent SVM detectors on the INRIA Person test set using various combinations of local pattern and HOG_{36} features under the L_1-Sqrt and L_2 normalization schemes for LBP/LTP. The LTP features use $\tau = 5$.*

- L_1: $\vartheta_k = \vartheta_k / (\sum_{i=1}^{n} \vartheta_i)$
- L_1-Sqrt: $\vartheta_k = \sqrt{\vartheta_k / (\sum_{i=1}^{n} \vartheta_i)}$ – *i.e.* the histograms are normalized to sum 1 then square-rooted.
- L_2: $\vartheta_k = \vartheta_k / \sqrt{(\sum_{i=1}^{n} \vartheta_i^2)}$ – this tends to emphasize cells whose counts are distributed into many bins.
- Log_2: $\vartheta_k = \log_2(\vartheta_k + 1)$ – *i.e.* a simple logarithmic compression of the histogram counts.

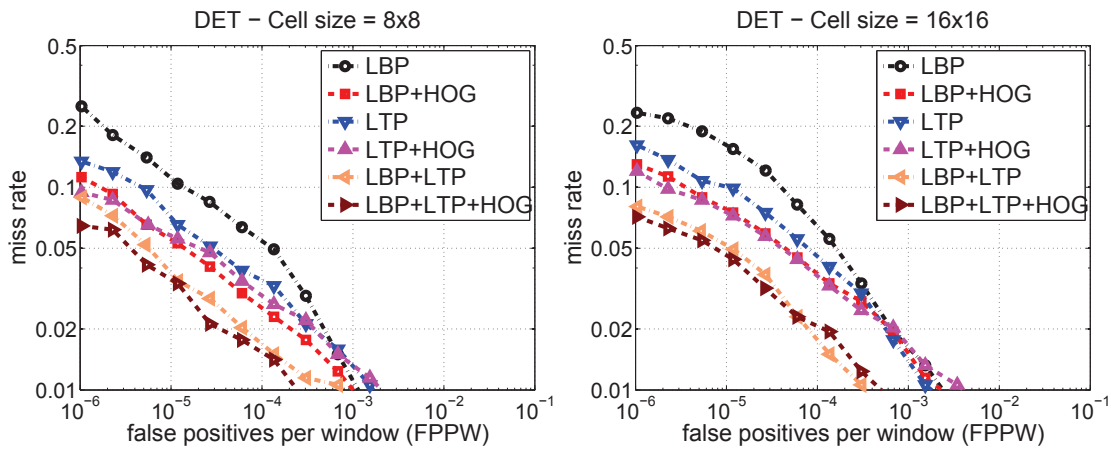
Note that all of these methods work at the cell level. Adding HOG-style block-wise normalization does not improve the performance of any of the local pattern descriptors or cell normalization methods tested.

For LBP based window-level classifiers on the INRIA Person set at 10^{-5} FPPW, all of these normalization schemes give miss rates of around 11.2% except L_1 which has a miss rate of 27.4% – *c.f.* Figure 4.7(d). This suggests that some form of compression is needed to downweight bins that have very concentrated counts, but that the exact form of compression used is not critical. However Table 4.3 shows that there are substantial differences between between the L_1-Sqrt and L_2 LBP/LTP normalization methods when used in combinations. We see that L_1-Sqrt generally gives the best results, especially for LTP, although the differences become smaller when HOG features are also present. L_1-Sqrt may predominate because it downweights bins with large numbers of counts more effectively than L_2. In particular, the "all zeros" bins have large numbers of counts for LTP in uniform regions, and the presence of noise or clutter often leads to the non-uniform bins of the histograms having quite large numbers of counts.

Based on these observations, we use bilinear vote interpolation and L_1-Sqrt normalization for our local pattern histograms.

τ	0	1	3	5	10	15	25
L_2	74.9	76.2	75.2	71.1	63.5	55.0	42.9
L_1-Sqrt	74.3	76.6	78.1	78.9	76.8	73.1	63.7

Table 4.4: *Average Precisions of LTP based Latent SVM detectors on the INRIA Person dataset, for different LTP threshold values τ under the L_2 and L_1-Sqrt normalization schemes.*



(a) (b)

Figure 4.8: *DET curves of non-latent detectors for individual and combined feature sets on the INRIA Person test set, for (a) 8×8 cells, (b) 16×16 cells. In each case LBP+LTP+HOG is the best performer followed by LBP+LTP. The 8×8 cells clearly outperform the 16×16 ones but have four times higher feature dimension.*

4.2.4.5 LTP Threshold

Regarding the LTP threshold τ, we find that $\tau = 5$ gray-levels gives the best overall performance. Table 4.4 shows Average Precisions for detectors with L_1-Sqrt and L_2 normalized LTP features and different values of τ on the INRIA Person dataset. L_1-Sqrt gives consistently better accuracy than L_2 over the range of values tested, and moreover it provides more tolerance to suboptimal τ values.

4.3 Combinations of Features

This section compares the accuracies of linear SVM detectors using various combinations of local pattern and HOG_{36} features – similar conclusions hold for combinations using HOG_{13} and HOG_{31} features. Feature channels can be combined in several ways including simple linear concatenation, tensoring, kernel averaging and multiple kernel learning [Vedaldi et al. 2009]. Here the dimension is too high for tensoring to be prac-

Features	INRIA Person	VOC2006 Car	VOC2006 Person
HOG	79.0	51.7	22.2
LBP	73.9	54.4	21.6
LTP	78.9	55.8	28.9
LBP+LTP	80.4	56.7	31.5
LBP+HOG	80.6	55.9	32.1
LTP+HOG	81.3	56.9	33.8
LBP+LTP+HOG	82.8	56.3	34.4
CS-LBP	67.9	45.6	13.4
CS-LTP	72.2	51.4	17.7
CS-LBPLTP	74.2	51.1	21.6
CS-LBP+HOG	78.1	54.5	28.0
CS-LTP+HOG	78.2	54.9	29.5
CS-LBPLTP+HOG	78.4	54.9	30.1
CS-LBP*	73.5	47.1	16.9
CS-LTP*	72.8	53.2	21.9
CS-LBPLTP*	75.2	53.8	24.3
CS-LBPLTP*+HOG	78.9	56.1	28.4

Table 4.5: *Average Precisions of Latent SVM detectors trained on various combinations of HOG_{36}, LBP, LTP, CS-LBP and CS-LTP on three classes from two datasets. In the last four rows, instead of using unweighted CS-LBP codes, the pixel's gradient magnitude is used to weight its histogram vote then each cell is block-normalized* w.r.t. *the four neighbouring blocks as in HOG, in the manner of Zheng et al. [2010].*

ticable and for computational efficiency we preferred to avoid kernelization despite its discriminative power, so we followed a simple channel concatenation approach. This puts the emphasis on the expressive power of the underlying feature sets and it allows us to use comparatively simple learning machinery. Nonetheless, in this approach it is important to choose the relative weightings of the different feature channels appropriately. Experiments on a validation set showed that, with our well-normalized cell-level histograms, assigning the same weight to each feature channel was sufficient to give good performance and that the results were not too sensitive to the exact relative values of the weightings used.

Figure 4.8 shows DET curves for various feature combinations in non-latent window-level classifiers on the INRIA Person test set, and Table 4.5 presents Average Precisions for latent detectors on three different object classes. Clearly, even though the HOG and local pattern features are already quite discriminant on their own, significant performance improvements can be achieved by combining them. Among the individual

feature sets, LTP gives the best results overall, followed by HOG. In combination, LBP+LTP+HOG gives the best results followed by LTP+HOG and LBP+LTP. The fact that the combination LBP+LTP significantly outperforms both LBP and LTP alone confirms that LBP and LTP extract complementary forms of local texture information. In fact, as Figure 4.5 suggests, LBP focuses mainly on micro-local texture details while LTP extracts stronger shape and texture cues.

The center-symmetric variants of LBP and LTP have much lower performance than the standard ones, both individually and in combinations, so they cannot be recommended for object detection despite their significantly lower dimensionality. The last four rows of Table 4.5 show that using a HOG-like CS scheme motivated by [Zheng et al. 2010], where each CS pair's gradient magnitude is used to vote into the cell histograms followed by HOG-style block-normalization, does increase the accuracy relative to standard CS-LBP/CS-LTP, but at the cost of 4 times higher dimensionality and even then it gives lower performance than standard LBP/LTP. These findings confirm those of Heikkilä et al. [2009] that although CS-LBP performs well for wide-base line matching, it is outperformed by LBP for object category recognition.

To the best of our knowledge, for both DET curves of window-level classifiers and AP's of non-latent and latent single root linear SVM detectors, the results for our LBP+LTP+HOG feature set are the best ever reported on these classes. This illustrates the benefits of using a rich set of complementary features for object detection.

4.4 Speed

Our complete algorithm for local pattern feature computation is summarized in Figure 4.9. For each feature type and each image pyramid scale, we compute the pixel-level feature maps, accumulate them into a grid of $P \times P$ pixel cells using bilinear interpolation, then apply cell or block level normalization as appropriate. This approach is also used for HOG [Felzenszwalb et al. 2009]. It has the advantages of speed and simplicity but it is limited to window step sizes that are multiples of the cell size. Alternatives include: (i) computing all features separately for each window as in Dalal and Triggs [2005] – this is very flexible in that it allows many different options for cell size and shape, step size, Gaussian weighting, *etc*. to be evaluated, but it is also very slow (although Dalal used caching schemes to palliate this); and (ii) using integral histograms [Porikli 2005] or the indirect feature selection approach of [Zhu et al. 2006], neither of which is as fast as using a fixed grid across the whole image. In our experience the cell-sized window stepping used in our scheme suffices for high accuracy while allowing rapid computation – an observation already made in [Dalal 2006].

For comparison, on a 2.4 GHz workstation, a HOG detector based on this image-level implementation takes about 0.7 seconds to process an INRIA Person test image, whereas

Input: A scaled image I and a set of feature map parameters Θ_r^k
Output: A vector of computed features

Initialization
(a) Compute interpolation coefficients and lookup table for uniform features.

Feature Map Computation Θ_r^k
(a) For LBP, compute the feature map Θ_r^k Eq. (4.1) for each of the RGB color channels, then map the LBP codes to uniform ones using table lookup.
(b) For LTP, compute the $\geq \tau$ & $\leq -\tau$ feature maps Eq. (4.2) for tolerance τ, then map the codes to uniform ones.
(c) For CS-LBP and CS-LTP, compute the binary codes without mapping them to uniform patterns.

Cell Histograms
(a) Divide the image into a dense grid of $P \times P$ pixel cells.
(b) Construct the code histogram for each cell using bilinear interpolation to weight each pixel's vote.
(c) Accumulate the separate R, G, and B votes into a single pooled histogram.

Image Descriptors
(a) Apply L_1-Sqrt normalization to each cell histogram.
(b) Collect the descriptors of the complete image into an array organized by cell indices.
(c) The descriptor of any rectangular region consists of its collected cell histograms.

Figure 4.9: *A summary of local pattern feature computation.*

a window level implementation takes around 8 seconds. Similarly our (non-optimized) implementation of the linear LBP+LTP+HOG detector takes about 3.4 seconds, or 2.2 if bilinear spatial interpolation is turned off in the local pattern features.

4.5 Summary

This chapter has detailed the configurations that we will use for our feature sets in all of the experiments below. Based on the results in this chapter, individually the LTP and HOG features perform well and the LBP ones perform quite well, but combinations are even better, with LBP+LTP+HOG performing best and LTP+HOG performing second best. These findings are confirmed by the detailed experiments in Chapter 6. For the HOG features, we confirmed that the [Felzenszwalb et al. 2009] HOG_{13} and HOG_{31} dimensionality reduction methods give good results. For the local pattern features, the inclusion of (pooled) color information, the use of bilinear interpolation for histogram voting, and L_1-Sqrt normalization all improve the results, and uniform codes help to re-

duce the dimensionality with little or no loss of discrimination. LTP clearly outperforms LBP, presumably owing to its ability to suppress noise and focus on contour information, while the center-surround variants of LBP and LTP are not competitive with the regular ones owing to their greatly reduced discriminative power.

Chapter 5: Dimensionality Reduction and Classifiers

In this chapter we discuss both dimensionality reduction methods, including Partial Least Squares and sparse feature selection, and various forms of fast nonlinear classifiers. Although we give some illustrative experimental results, full-scale testing is postponed until Chapter 6.

5.1 Dimensionality Reduction

Our full feature set LBP+LTP+HOG_{31} has 19968 dimensions for 48×128 detection windows. Although feature sets of such high dimension do improve discrimination, they lead to: i) long training times, as the features are bulky to store and slow to process; ii) an increased risk of overfitting, despite the use of well-regularized classifiers; iii) slow final detectors – more than half of the total time is often spent computing dot products between feature vectors and learned classifier weights. We now explore two strategies for tackling the dimensionality problem by finding reduced-dimensional feature sets that have similar discriminative power for the given problem: *discriminant subspace projection* and *relevant feature selection*. Among the available discriminant subspace projection methods we tested Partial Least Squares (PLS), Principal Component Analysis (PCA) and discriminative topic models. Among the wide range of available feature selection methods, we tested SVM weight truncation, L_1 regularization and boosting over the feature pool. The following subsections provide details.

5.1.1 Discriminant Subspace Projection

Discriminant subspace methods aim to find a low dimensional linear projection of the feature vector that preserves much of the discriminative power of the input. We will use them as preprocessors to reduce the feature dimension before standard SVM learning. Note that with any such method, if a linear SVM is learned, its weight vector can be pushed back through the projection to provide an equivalent linear SVM on the original feature space so expensive multidimensional projection is not needed at run time.

We tested three methods for learning suitable projection matrices, Partial Least Squares [Wold et al. 2001; de Jong 1993], Principal Component Analysis (PCA), and max-margin discriminative topic models [Zhu et al. 2009]. PLS is our method of choice because it is very efficient and it gives essentially identical performance to the the raw input features. PCA gives very poor results. For example, for linear SVM over HOG features on the INRIA Person dataset, training on the leading 30 principal components reduces the AP by 21% relative to training on the complete feature set. Our preliminary tests with max-margin topic models also gave less good results than PLS for much longer run times[1], so we will not discuss this further here.

5.1.1.1 Partial Least Squares (PLS)

"Partial Least Squares" or "Projection to Latent Structures" (PLS) is a regularized linear least squares regression method that was initially developed in chemometrics to solve ill-conditioned multiple output regression problems in cases with few training examples and many highly-correlated variables and outputs [Wold et al. 2001]. Here we discuss the different PLS algorithms only briefly. Further details, a comparison with other methods and details of nonlinear extensions can be found in [Rosipal and Kramer 2006; Wold et al. 2001; de Jong 1993].

PLS is a linear L_2 regression method, but instead of including a conventional regularization term it controls the solution by solving the regression problem exactly in a lower-dimensional space given by a truncated power method basis expansion of the input space. Consider a conventional damped least squares regression problem $\mathbf{XB} \approx \mathbf{Y}$ where $\mathbf{X}, \mathbf{Y}$ are respectively the input and output matrices (stored by rows) and $\mathbf{B}$ is the weight matrix to be learned. Regularized least squares solves this as $\mathbf{B} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y}$, which for large λ can be written as $\left(\sum_{k=0}^{\infty} \frac{(-\mathbf{X}'\mathbf{X})^k}{\lambda^{k+1}}\right)\mathbf{X}'\mathbf{Y}$. This is an expansion of the solution in terms of the power method basis matrices $\{\mathbf{X}'\mathbf{Y}, (\mathbf{X}'\mathbf{X})\mathbf{X}'\mathbf{Y}, (\mathbf{X}'\mathbf{X})^2\mathbf{X}'\mathbf{Y}, ...\}$. The expansion is intimately linked to regularization in the sense that it quickly extinguishes directions corresponding to small eigenvalues of $\mathbf{X}'\mathbf{X}$ (small singular values of $\mathbf{X}$), which are exactly the ones that diverge when $\mathbf{X}$ is close to singular and the undamped solution $\mathbf{B} = \mathbf{X}^+\mathbf{Y} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ is used.

Now consider PLS in the case of scalar outputs ($\mathbf{Y}$ is a vector). PLS regularizes the solution not by fixing a λ and summing to convergence, but by truncating the basis expansion and then solving the undamped problem exactly in the resulting subspace. In practice it also orthogonalizes each new vector of the series $(\mathbf{X}'\mathbf{X})^k\mathbf{X}'\mathbf{Y}$ against the previous vectors as it is generated – a standard linear algebra tool known as Lanczos recursion. Besides greatly improving the numerical stability, this orthogonalization

[1] This may be an algorithmic issue so further testing of topic model based approaches is probably warranted.

Input: Feature matrix $\mathbf{X}$, class label matrix $\mathbf{Y}$ and number of latent variables k
Output: Projection matrix $\mathbf{W}$, orthonormal loading matrix $\mathbf{V}$, latent factor matrix $\mathbf{T}$ and response regressor matrix $\mathbf{B}$

Initialization
(a) Center and scale $\mathbf{X}$ and $\mathbf{Y}$ to zero mean and unit variance.
(b) Compute $\mathbf{S} = \mathbf{X}'\mathbf{Y}$.

For $i = 1 \ldots k$

(a) Find the sorted SVD decomposition $\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}'$
(b) Extract the i_{th} weight, latent and loading vectors
  (1) $\mathbf{w}_i = \mathbf{U}_1$
  (2) $\mathbf{t}_i = \mathbf{X}\mathbf{w}_i$
  (3) $\mathbf{p}_i = \mathbf{X}'\mathbf{t}_i$
(c) Scale $\mathbf{t}_i, \mathbf{w}_i, \mathbf{q}_i$ by $1/\|\mathbf{t}_i\|$ so that $\mathbf{t}_i'\mathbf{t}_i = 1$.
(d) Orthogonalize $\mathbf{p}_i$ *w.r.t.* $\mathbf{V}$ to make $\mathbf{v}_i = \mathbf{p}_i - \mathbf{V}(\mathbf{V}'\mathbf{p}_i)$, then normalize $\mathbf{v}_i$ to unit length.
(e) Deflate $\mathbf{S}$ by projecting out the regressed component, $\mathbf{S} = \mathbf{S} - \mathbf{v}_i(\mathbf{v}_i'\mathbf{S})$
(f) Append the $\mathbf{w}, \mathbf{t}, \mathbf{p}, \mathbf{q}, \mathbf{v}$ vectors to their respective matrices $\mathbf{W}, \mathbf{T}, \mathbf{P}, \mathbf{Q}, \mathbf{V}$.

Compute the final regressor matrix $\mathbf{B} = \mathbf{W}\mathbf{Q}'$.

Figure 5.1: *The SIMPLS algorithm for the computation of k PLS components.*

allows the linear regression to be solved as a series of 1D subproblems with closed form solutions, at each step deflating $\mathbf{Y}$ (removing the part of it that has just been explained) in preparation for the next step. Note that the PLS solution lies in the subspace spanned by the directions $\{\mathbf{X}'\mathbf{Y}, \ldots, (\mathbf{X}'\mathbf{X})^k\mathbf{X}'\mathbf{Y}\}$ – increasing k reduces the effective regularization, but the result does not usually coincide with the damped least squares solution $(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y}$ for any zero or finite λ until $k = \mathrm{Rank}(\mathbf{X}'\mathbf{X})$, when the undamped solution is obtained.

For q-dimensional outputs $\mathbf{Y}$, rather than introducing a q-dimensional block of new vectors $(\mathbf{X}'\mathbf{X})^k\mathbf{X}'\mathbf{Y}$ at each step, standard PLS selects the most useful single $\mathbf{X}$-vector in the subspace spanned by the block by finding the leading singular vector of the residual $\mathbf{X}'\mathbf{Y}$ matrix and sending this through the orthogonalization process. At the cost of a (possibly approximate) leading eigenvector extraction at each step, this keeps the regularization effective even when q is large, while assuring rapid decrease of the residual $\mathbf{Y}$ and retaining 1D regression subproblems.

SIMPLS. The historically most common algorithm for PLS is NIPALS [Wold et al. 2001], but here we use SIMPLS [de Jong 1993] which has the advantage of generating an explicit orthogonal projection matrix for the reduced subspace. Let $\mathbf{X}_{m\times n}$ represent the matrix of the n predicate variables for each of the m examples and $\mathbf{Y}_{m\times q}$ represent the

corresponding matrix of response variables. We assume that both are centered to mean zero. Then SIMPLS estimates the regressor $\mathbf{Y} \approx \mathbf{X}\,\mathbf{B}$ column by column in the form

$$\mathbf{Y} \approx \mathbf{T}\mathbf{Q}' \text{ where } \mathbf{T} = \mathbf{X}\mathbf{W} \tag{5.1}$$

so that $\mathbf{B} = \mathbf{W}\mathbf{Q}'$. However to find the next columns of $\mathbf{W}$ and $\mathbf{Q}$ it proceeds indirectly, simultaneously constructing a back-regressor $\mathbf{X} \approx \mathbf{T}\mathbf{P}'$ and (most importantly) $\mathbf{V}$, an orthonormalized version of $\mathbf{P}$. First, using the SVD of $\mathbf{S} = \mathbf{X}'\mathbf{Y}$, it finds the unit vectors $\mathbf{w}$, $\mathbf{q}$ that maximize $(\mathbf{X}\mathbf{w})'(\mathbf{Y}\mathbf{q})$. It then finds $\mathbf{t} = \mathbf{X}\mathbf{w}$ and $\mathbf{p} = \mathbf{X}'\mathbf{t}$ and orthonormalizes $\mathbf{p}$ against the current (initially empty) $\mathbf{V}$ matrix to give a new column $\mathbf{v}$ of $\mathbf{V}$. Finally $\mathbf{S}$ is orthogonalized against $\mathbf{v}$ to provide the $\mathbf{S}$ matrix for the next round of the process. The complete algorithm is given in Figure 5.1.

PLS for Classifiers. Although PLS was designed as a stand-alone regression tool, it can also be used as a dimensionality reduction method for a subsequent classifier [Schwartz et al. 2009; Kembhavi et al. 2010]. This is particularly effective in cases where the input variables are strongly correlated, which happens almost inevitably in object detection with rich feature sets containing many related features from overlapping regions. Here we use PLS as a subroutine to project out a small number of highly discriminative directions for subsequent SVM learning. During each training iteration, after Latent SVM alignment, we use SIMPLS to find and project out a feature subspace for classifier training. Below we will report results for both linear and nonlinear classifiers trained in these reduced spaces.

Note that although $\mathbf{B} = \mathbf{W}\mathbf{Q}'$ is the PLS regressor matrix so that $\mathbf{W}$ might seem to be a good candidate for the reduced subspace projection, $\mathbf{W}$ is not an orthogonal projector (its columns are not orthonormal). On the other hand, $\mathbf{V}$ is orthonormal and being an orthonormalized version of $\mathbf{P}$ it provides a suitable projector as follows: $\mathbf{X} = \mathbf{T}\mathbf{P}' = \mathbf{T}\mathbf{R}'\mathbf{V}'$ where $\mathbf{R}$ is upper triangular and $\mathbf{P} = \mathbf{V}\mathbf{R}$ is the QR decomposition (orthonormalization) of $\mathbf{P}$. By the orthonormality of $\mathbf{V}$, $\mathbf{X}\mathbf{V} \approx \mathbf{T}\mathbf{R}'\mathbf{V}'\mathbf{V} = \mathbf{T}\mathbf{R}'$ is a reduced subspace equivalent to an orthonormalized version of $\mathbf{T}$. Besides being orthonormal, this is an equally good basis for regressing $\mathbf{Y}$ since $\mathbf{Y} \approx \mathbf{T}\mathbf{Q}' = (\mathbf{T}\mathbf{R}')(\mathbf{R}'^{-1}\mathbf{Q}')$. We therefore use the $\mathbf{V}$ matrices to project out our reduced subspaces during SVM learning. This is confirmed experimentally: projecting the features using $\mathbf{V}$ instead of $\mathbf{W}$ gives better precision and computationally more stable solutions. For example, for latent detectors on 30 PLS dimensions, using $\mathbf{W}$ instead of $\mathbf{V}$ reduces the AP by 1.3% for the INRIA Person class (from 79% AP to 77.6% AP) and by 1% for the VOC2006 person class. Figure 5.2 shows positive and negative SVM weight images for the first five PLS components of $\mathbf{W}$ and $\mathbf{V}$ based projection, for HOG features on the INRIA Person class. For $\mathbf{W}$, the components look superficially meaningful, but this is because they are all similar and highly aligned

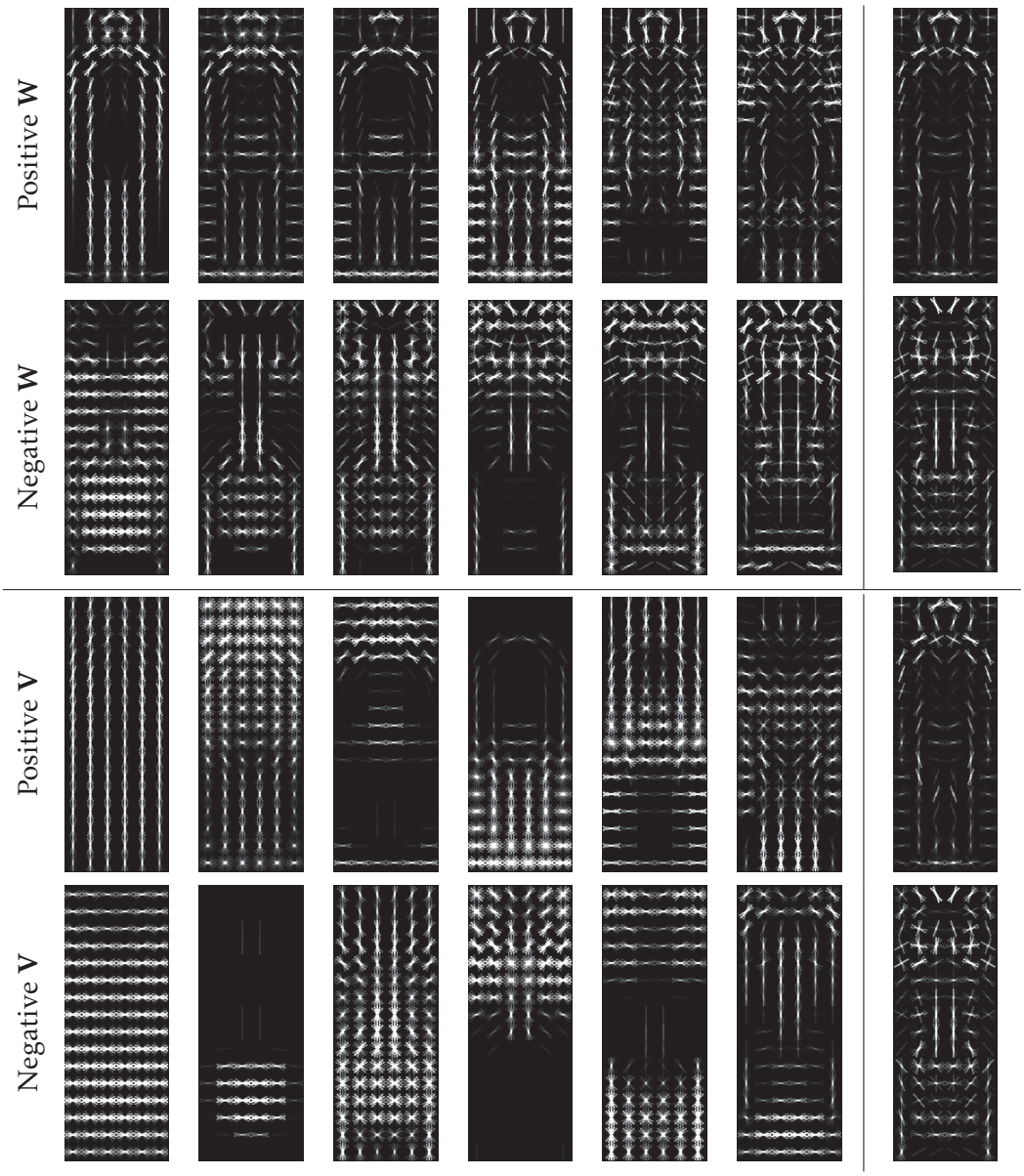


Figure 5.2: *Positive and negative SVM weight images of the first six PLS components under the projections* **W** *and* **V** *for HOG features on the INRIA Person dataset. The last column shows the final SVM learned from the PLS features.*

with the final SVM. This redundancy makes SVM learning more difficult. Conversely, although the **V** components are less "readable", they encode obviously different shape cues like horizontal edges, head, leg and torso cues, *etc.*, and in combination these lead to a better final detector.

For classifier learning, there is a single response variable (the class label y) and the data matrix **S** becomes a vector, eliminating the need for an SVD and giving very fast PLS computation times. When dealing with combined feature sets, we simply learn the PLS reduction on the joint (concatenated) feature vector. The number of PLS dimensions k for a given problem can be chosen using a validation set, but by default we use $k = 30$

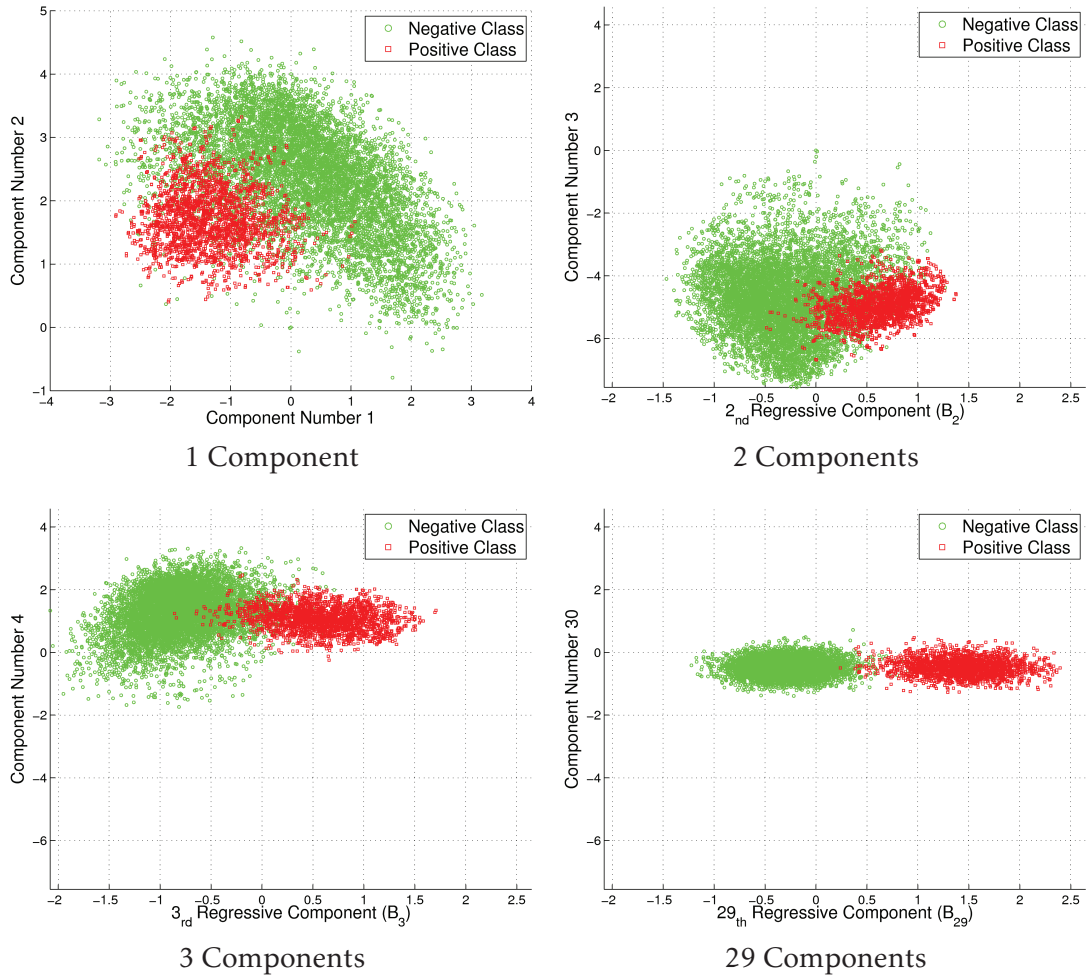


Figure 5.3: *An illustration of the class separations produced by different numbers of PLS components for HOG features on the initial stage INRIA Person training set. The scatter is plotted for the newly added* $(j+1)$*st component against the best Least Squares regressor on all preceding components.*

for all of our linear classifiers as we find that any value about 25 and 60 typically gives similar results. For all of our individual and combined feature sets, training against 30 PLS components does not change the AP significantly relative to training against the corresponding raw input features. Clearly, unlike PCA, the class awareness of PLS allows it to project out highly discriminant feature combinations even for modest number of output dimensions.

Although the first PLS dimension already gives a moderately good linear separator, including additional dimensions does increase the accuracy. The influence of additional dimensions on the separation between the two classes is illustrated in Figure 5.3. Here the class scatter for a newly added PLS dimension is plotted against the scatter for the best least squares class label regressor $\mathbf{B}$ on all of the preceding components. We see

Feature Type	Learning Space	Time spent (in seconds)				% Speed Up
		Features Computation	Latent Search	Classifier Learning	Total	
HOG_{36}	Raw	1668.0	404.0	1204.0	3276.0	37
	PLS	1681.0	476.0	**233.0**	2390.0	
LBP	Raw	2948.0	445.0	1986.0	5379.0	50
	PLS	2969.0	507.0	**107.0**	3583.0	
LTP	Raw	4797.0	688.0	2053.0	7538.0	35
	PLS	4755.0	774.0	**62.0**	5591.0	
LBP+LTP+HOG	Raw	6449.0	1051.0	2336.0	9836.0	29
	PLS	6367.0	1184.0	**49.0**	7600.0	

Table 5.1: *A breakdown of the time spent (in seconds) in various stages of single root detector learning on the VOC2006 person class. PLS significantly speeds up the classifier learning (which includes the cost of PLS itself), resulting in a reduction in the overall detector training time of between 25% and 50%.*

that the separation gradually increases as dimensions are added [Kembhavi et al. 2010].

Table 5.1 shows that PLS significantly speeds up batch SVM training, often by more than an order of magnitude, and hence provides a useful reduction in the overall detector training time. For instance for single root detectors, PLS speeds up detector training by 25% to 50% depending on the feature set used, and similar increases in speed are seen in the 2 root case where PLS speeds up the training of LBP+LTP+HOG detectors by about 45%.

Overall, the advantages of using PLS as a dimensionality reduction tool for SVM training are as follows:

- The resulting classifiers are much faster to train. Even including the cost of PLS learning and reduction, PLS often speeds up batch SVM training by an order of magnitude, leading to faster overall detector training. For example, for 2 Root Latent SVMs on VOC2006 with LBP+LTP+HOG features and 28 latent training iterations, training an SVM with PLS reduced features (including the cost of the PLS) takes around 10 minutes per class in total, whereas training with unreduced features takes around 125.

- The resulting detectors typically have similar or even slightly better[2] accuracies than the equivalent unreduced ones.

[2]SVM detectors are based on limited numbers of imperfect training examples, each with background clutter. PLS dimensionality reduction implicitly averages over many such examples. Presumably this sometimes manages to project away part of the noise and hence reduce overfitting.

- The resulting (linear) detectors are just as fast as standard ones because the learned classifier weights $\hat{\rho}$ can be pulled back through the PLS projection to give an equivalent classifier $\rho = \mathbf{V}\,\hat{\rho}$ on the original feature space.
- Optionally, the low-dimensional representation allows many kinds of nonlinear classifier to be trained in the reduced space, which might not be feasible in the unreduced one owing to its high dimensionality – *c.f.* Sec. 5.2.

5.1.2 Feature Selection and Sparsity

Discriminant subspace projection methods are simple in the sense that they are based on familiar tools from linear algebra, but the resulting projection matrices are typically dense. This potentially makes them slow as all of the features in the set need to be evaluated for use in the projection, and although they downweight irrelevant features that contribute little or nothing to the final decision, they do not categorically identify and suppress them. Such features are expected to be common in visual recognition as many visual classes are characterized by sparse cues such as object outlines or particular kinds of texture. Discarding the unnecessary features is expected to both speed up the detector and improve its generalization because it helps the classifier to focus on structures that are useful for recognition while ignoring noise. Moreover, very sparse "filter" classifiers could potentially be used in the context of coarse-to-fine or cascade approaches to further increase the scanning speed.

There are a large number of feature selection methods designed to choose small sets of discriminative features from large pools of candidate features. We tested three main approaches: greedy selection based on boosting, sparsity-inducing regularizers such as the L_1 regularized variants of SVM and logistic regression, and importance criteria based on trained classifier weights.

5.1.2.1 Boosting For Feature Selection

Besides being a classifier training procedure, boosting can also be viewed as a stepwise greedy method for selecting a set of discriminant features (the chosen weak classifiers) from a large feature pool [Viola and Jones 2004]. We investigated the used of several variants of boosting methods, including asymmetric AdaBoost with decision stumps [Viola and Jones 2004] and GentleBoost with regression stumps [Torralba et al. 2007] as feature selectors for SVM classifiers[3] following latent root and part position estimation.

[3]It is well known, and our experience confirms, that over any give set of features (weak classifiers), linear SVM generally finds a better classifier than boosting, or at least finds an equally good one more quickly. Hence, even though we select features using boosting, we actually relearn the classifier over them using linear SVM. The main strength of boosting is not the absolute quality of its classifier over the selected features, but its ability to select good features/weak classifiers from a large and potentially unbounded set in an efficient stepwise manner.

However our initial results on this were not encouraging and we did not pursue this further. For instance, for LBP+HOG features on the VOC2006 person class, using AdaBoost to sparsify the feature set to ~34% nonzeros (~30 features per cell) reduced the AP by 9.6% from 32.1% to 22.7%.

5.1.2.2 L_1 Feature Selection

Another approach to feature selection is to modify the learning problem in such a way that its solution will naturally be sparse. Sparsity-inducing regularizers are a common way of achieving this. These can be non-convex, leading to methods such as the Relevance Vector Machine [Tipping 2001], but here we preferred to preserve convexity so we tested only L_1 regularizers [Fan et al. 2008; Schwartz et al. 2009]. An L_1 regularized sparse classifier is characterized by an objective function of the form

$$L_D(\boldsymbol{\rho}) = \|\boldsymbol{\rho}\|_1 + C \sum_{i=1}^{m} l\,(S_{\boldsymbol{\rho}}(\mathbf{r}_i), y_i). \tag{5.2}$$

Here, $\|\boldsymbol{\rho}\|_1$ denotes the L_1 norm of the classifier weight vector $\boldsymbol{\rho}$. As before, we use the latent formulation where each of the m training examples consists of an image region $\mathbf{r}_i$ and its label y_i, and $l\,(S_{\boldsymbol{\rho}}(\mathbf{r}_i), y_i)$ is a non-negative loss function defined over a (non-latent or latent) region scoring function $S_{\boldsymbol{\rho}}(\mathbf{r})$. For example, $S_{\boldsymbol{\rho}}(\mathbf{r}_i) = \boldsymbol{\rho}.\boldsymbol{\varpi}_i$ in the simplest non-latent case, where $\boldsymbol{\varpi}_i$ are the region features. The regularization parameter C balances the contributions of the regularizer and the loss function. We will suppose that $l\,(S_{\boldsymbol{\rho}}(\mathbf{r}_i), y_i)$ is a convex function of $\boldsymbol{\rho}$, so the overall objective function remains convex (modulo, in the latent case, the search for the latent positions of the positive examples). The overall objective function is non-differentiable due to the L_1 norm, so mathematical programming based algorithms are needed for optimization. This is an area of current interest in the machine learning community and new algorithms are emerging all the time, especially for large scale problems with many features [Langford et al. 2009; Schwartz et al. 2009].

L_1 Regularized Logistic Regression. For logistic regression (LR), we have

$$l\,(S_{\boldsymbol{\rho}}(\mathbf{r}_i), y_i) = \log\,(1 + \exp^{-y_i S_{\boldsymbol{\rho}}(\mathbf{r}_i)}). \tag{5.3}$$

This loss is twice differentiable, which simplifies the L_1LR optimization problem. We tested the coordinate descent optimization algorithm of [Fan et al. 2008], but the resulting detector do not outperform either standard linear SVM or standard L_2 regularized Logistic Regression (where $\|\boldsymbol{\rho}\|_1$ in Eq. (5.2) is replaced with the squared L_2 norm $\|\boldsymbol{\rho}\|^2$). For instance, for LBP+LTP+HOG features on the VOC2006 person class with L_1 Logistic

Regression set to give 16% nonzero weights, the AP is 5.9% lower than that of dense linear SVM. Even dense L_2 Logistic Regression is not competitive with linear SVM, having a 3.6% lower AP on this class, and a 2.3% lower one on the INRIA Person dataset. Moreover, with the current algorithm training L_1 Logistic Regression is very slow – in fact even for L_2 Logistic Regression a new dual formulation [Yu et al. 2011] was needed because primal based training was extremely slow.

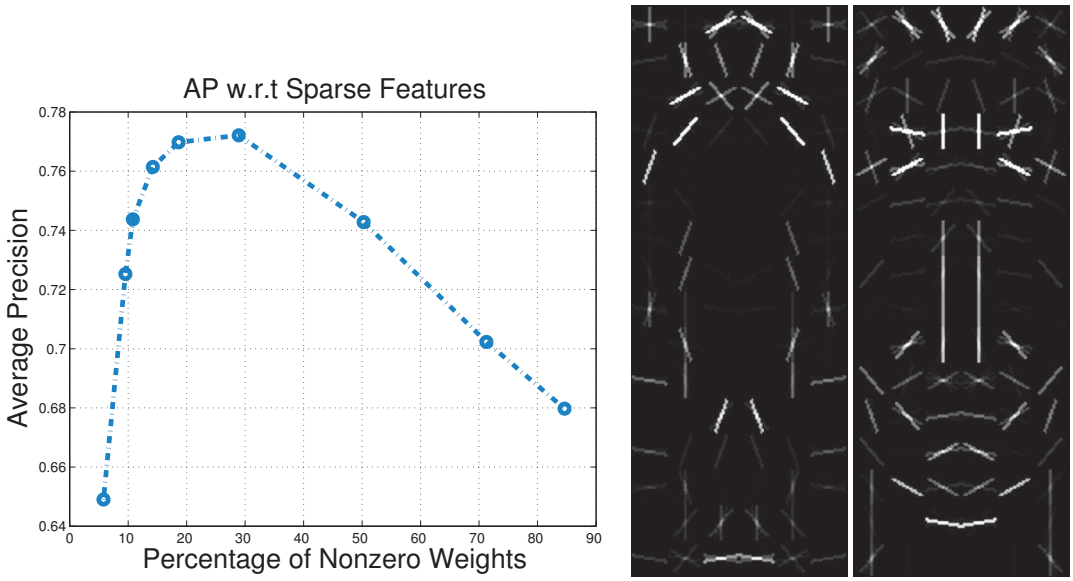
These results confirm our experience with other Logistic Regression classifiers on these problems. Despite the substantial overlaps between the positive and negative training sets, which might suggest that more statistically-based classifiers like Logistic Regression would have an edge over SVM, SVM consistently gives both better results and much faster training. Given that the regularizers are the same and the loss functions are asymptotically the same, this suggests that attempting to enforce a suitable margin on these datasets is more important than detailed modeling of the overlap statistics.

L_1 Regularized SVMs. Although the idea of combining L_1 regularization with SVM is old, the joint non-differentiability of the hinge loss and the L_1 regularizer makes the overall optimization problem somewhat delicate (at least for large scale problems) and it is only surprisingly recently that this issue has received much attention from the machine learning community. As a result, the existing algorithms leave something to be desired[4].

Fung and Mangasarian [2002] gave an early second order method based on the dual of the hinge loss, but this was intended for applications where the number of examples m is small. Their algorithm scales as $O(m^3)$, making it infeasible for object detection problems where there are tens of thousands of examples in the later stages of training. Similarly, a feature-oriented reformulation would scale as $O(n^3)$ where n is the number of feature dimensions, which is also prohibitive.

Shalev-Shwartz and Tewari [2009] describe a stochastic mirror descent algorithm (SMIDAS), making it sparse using a simple trick that works independently of the number of examples to give run times that scale as $O(n)$ with the feature dimension. We slightly modified their code to allow different costs (hinge loss slopes) for the positive and negative examples. Although the resulting method gives comparable results to a dense SVM classifier in the initial stages of latent learning, as soon as hard examples are introduced the differences in performance grow substantially and the final detector gives much worse results. We do not have a good explanation for this and suspect that the problem is algorithmic.

[4]We would expect this situation to change over the next year or two, and we are currently on working this ourselves.



(a) (b) (c)

Figure 5.4: *The effect of L1R-L2SVM classifier sparsity on Average Precision for* HOG_{36} *features on the INRIA Person dataset. (a) AP versus the sparsity level. Images of the (b) positive and (c) negative HOG weights for the classifier with 19% nonzero weights (AP 77%). The most important information is contained in the shoulder, head and leg signals.*

L_1 Regularized L_2 SVMs. Another related approach is the (misleadingly named) "L_2" SVM variant, where the hinge loss is replaced by its square, making the loss function differentiable in return for a method that is (in principle at least) less resistant to outliers. The differentiable loss function allows L_1 regularization to be introduced with fewer algorithmic difficulties and LibLinear [Fan et al. 2008] provides an implementation. Despite the use of the squared hinge loss, the results given by this *L1R-L2SVM* method are encouraging, being much better than both SMIDAS and L_1 Logistic Regression. For example, for HOG_{36} features on the INRIA Person dataset with 19% nonzeros (642 of the 3456 weights), *L1R-L2SVM* achieves 77% AP, as compared to 79% for conventional SVM with the complete feature set. Figure 5.4(a) shows the impact of sparsity on the AP for this dataset, obtained by varying the regularization parameter C. Good results are maintained down to about 14% sparsity. Figure 5.4(b, c) show the learned positive and negative weights for one of the resulting sparse person classifiers. As expected, most of the characteristic information is carried by the shoulder, head and leg outlines. The results for *L1R-L2SVM* are still 2% below those of a conventional dense SVM, but they strengthen our intuition that much of the information carried by the features is redundant, so that sparse classifiers should be able to do almost equally well as dense ones. The method that we turn to next demonstrates this conclusively.

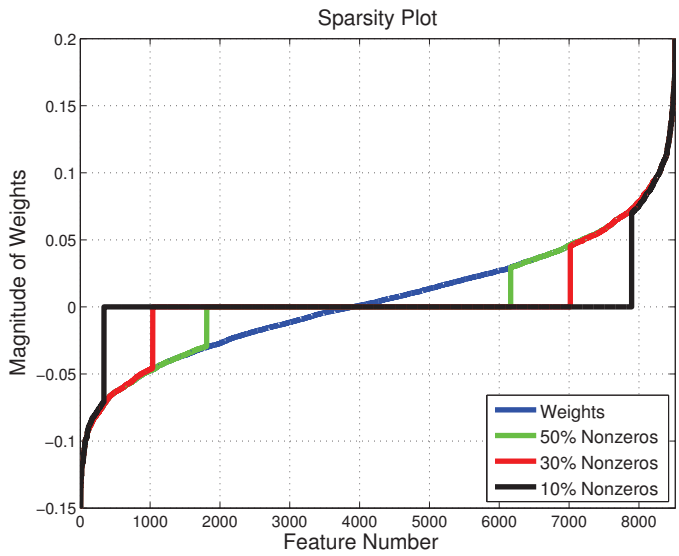


Figure 5.5: *Linear SVM weights for the INRIA Person class sorted into increasing order, with several of the resulting sparsity intervals. Weights that lie within the given sparsity interval, and that thus make only a small contribution to the overall template energy, are truncated to zero.*

5.1.2.3 Weight Truncation Based Sparse Classifiers

Motivated by the above results, we tested a simple heuristic for converting a traditional (L_2 regularized hinge loss) SVM classifier into a sparse one. If we examine the learned weights ρ of a linear classifier, many of them typically have quite small magnitudes – *c.f.* Figure 5.5. Empirically, the bottom 50% of the weights by magnitude carry only about 20% of the total SVM weight energy, whereas the top 10% carry about 30% of it. (Here, weight energy is defined as the sum of the magnitudes of the weights.) This is to be expected because most of the common visual object classes are characterized by relatively sparse cues such as object outlines or particular types of texture. To the extent that the small components represent noise inherited from the training set rather than features essential for recognition, both classifier accuracy and speed may be benefited by suppressing them.

Theoretically, any method based on the SVM weight values and the characteristics of the corresponding features could be used to select the most important weights for retention, but here, for simplicity, we just sort the weights by magnitude and retain the largest ones, setting the rest to zero. The detector is then retrained using only the selected features. Retraining adjusts the weights to compensate for the deleted features (which are often correlated with retained ones), giving a modest but systematic improvement in performance.

There are actually several variants of this idea to test. Firstly, we can truncate either to a given percentage of nonzeros or to a given percentage of the overall weight-energy.

However the difference is merely one of labeling and for simplicity we will always use the percentage of nonzeros here owing to its closer connection with sparsity levels and hence detector speed. Secondly, for multichannel feature sets, we can either force the sparsity to be distributed uniformly across the different channels, or work jointly and let it fall where it will. In practice we do the latter by default, remembering that the different channels are normalized to have similar ranges of variation and hence similar overall weights. We will see that each channel does indeed make a significant contribution to the discriminative power of the feature set, which is an indication that this choice is reasonable. Thirdly, for multiple roots and roots plus parts the given level of sparsity can either be enforced as a whole over the entire ensemble, or individually on each component and part separately. We do the later. Fourthly, we could apply the sparsification once and for all, or in gradually increasing increments over several stages of classifier retraining, for example decreasing the number of nonzeros by a factor of two each time. We do the former as we find that in practice it gives almost identical results with significantly lower training times. Finally, the sparsification can be applied at any stage of the latent learning cycle. We tested the following three approaches:

Final Weight Truncation (FWT). In its simplest form, weight truncation can be applied to an existing detector after the final stage of training, *i.e.* a dense detector is trained through all stages, then the result is sparsified and retrained once using the same hard negatives. (Actually, we skip the retraining if there are more than 50% nonzeros as in this case it makes only a negligible difference). Figure 5.6 shows Average Precision plots for FWT sparse detectors with different levels of sparsity and retained SVM template energy on the 10 classes of the VOC2006 dataset. It is clear that substantial reductions in the feature set dimension are possible with little or no loss in accuracy. However although FWT is simple to train and it already gives good results, it is not optimal because it is trained using latent labels and hard negatives found by the previous dense detector.

Iterative Weight Truncation (IWT). In this strategy, sparsification is re-run at each stage of training after Stage II of the latent learning cycle (*c.f.* Sec. 3.3.2), using hard negatives and Latent SVM positions obtained from the previous-stage sparse detector. Training the sparse detectors using their own latent labels and hard negatives improves their APs to such an extent that they give almost identical results to dense detectors even for very sparse feature sets, and often even better results for ones of intermediate sparsity. For example, for LBP+LTP+HOG features on VOC2006 at 10% nonzeros and at 15% nonzeros, IWT gives respectively 39.3% and 39.6% Mean AP as compared to 39.6% for dense detectors. Although IWT gives good results and it is fast to test, it is rather slow to train because in each iteration training needs to be done on the complete dense

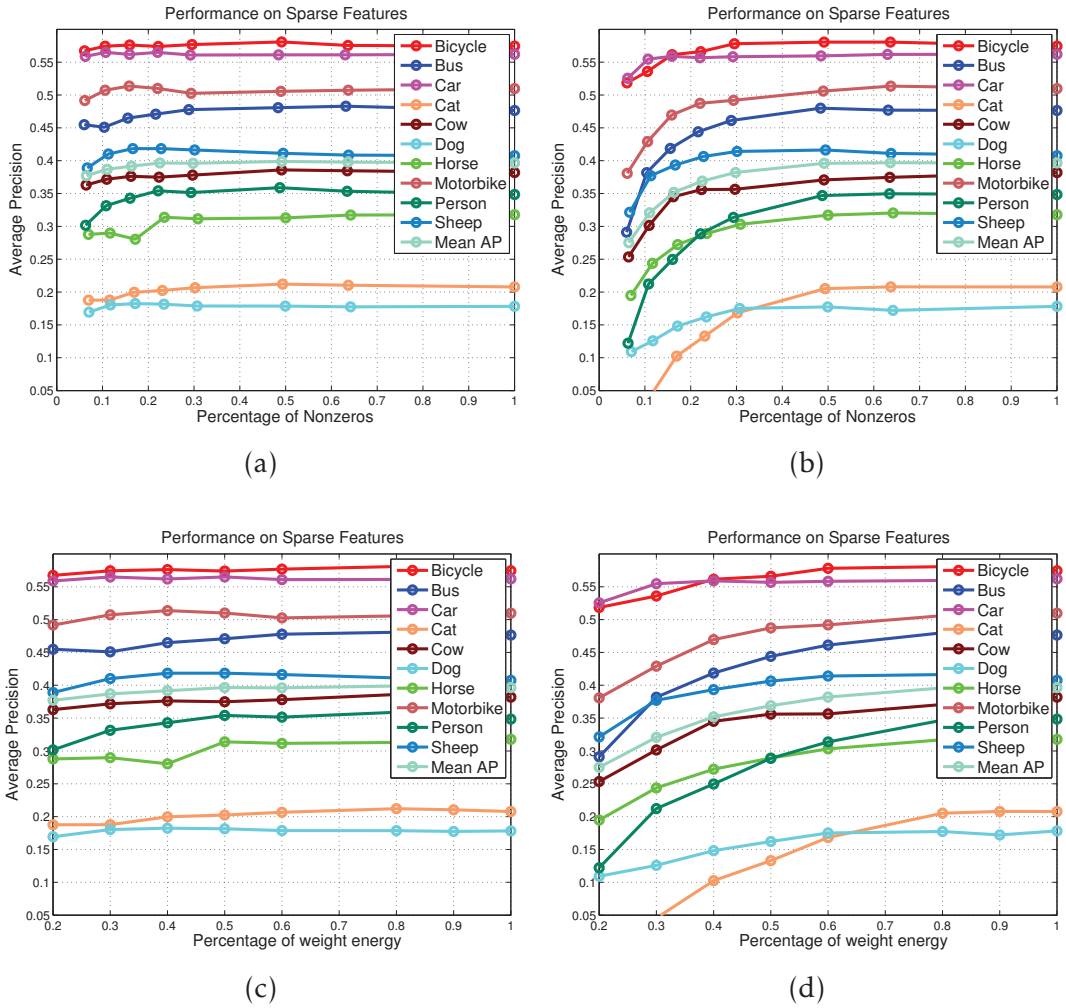


Figure 5.6: *Average Precisions of detectors trained on the VOC2006 datasets and sparsified using the Final Weight Truncation (FWT) method. Plots (a) and (b) show detectors that were (a) and were not (b) retrained after enforcing sparsity. Plots (c) and (d) show the APs of retrained and non-retrained FWT detectors* w.r.t. *the percentage of the initial SVM weight energy retained in the sparsified detectors. Note the consistent improvements produced by retraining and the fact that although about 20% of the total weight energy is carried by the top 6% of the weights, this figure is somewhat variable making it hard to use percentages of energy as a means to set numbers of nonzeros or vice versa.*

feature set in order to find the weights needed for feature selection, and then again on the selected features.

Weight Truncation (WT). In this strategy, the detector is sparsified once and for all after Stage II of the latent learning cycle, and the features selected there are retained at all subsequent stages. WT gives only slightly less accurate results than IWT (*c.f.* Chapter 6) and it is faster to train. However note that it is also somewhat sensitive to its initialization. In particular, if we initialize both IWT and WT directly after the

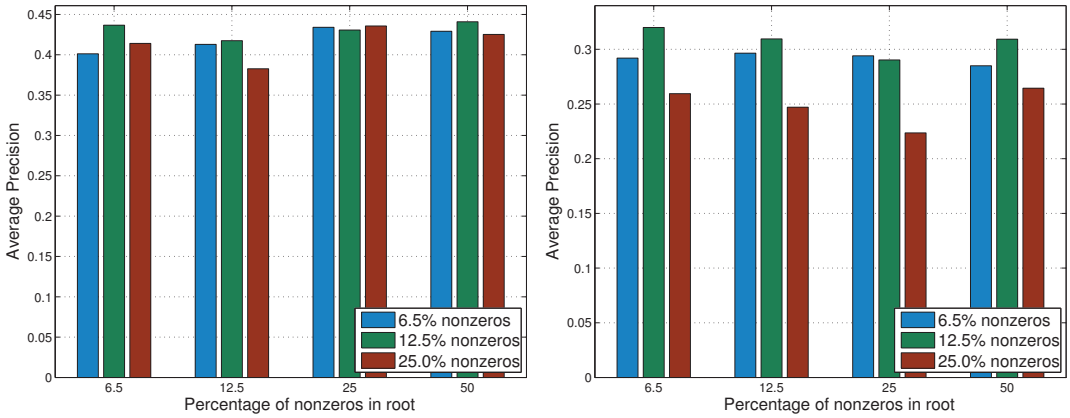


Figure 5.7: *Average Precisions for different percentages of nonzero weights assigned to the root and parts of single root and eight parts WT detectors of VOC2006 person (a) and cat (b) classes. Each group of bars shows Average Precisions for detectors having various percentages of nonzeros for their parts and the given percentage of nonzeros in their root.*

initialization stage of latent learning rather than at Stage II, WT gives results that are much worse than IWT.

Compound Detectors. We can use all three of the above strategies to train multi-root multi-part detectors. Training multi-root detectors using any of these methods and training multi-root multi-parts ones using FWT or IWT does not present any special challenges, except that after enforcing sparsity the positives must be rescanned to find the best latent variables. However multi-root multi-part detectors trained using WT can give poor performance if the parts are not well initialized. We tested various techniques for part initialization and found that the best approach was to first use the sparse root filter to find the best positions and dimensions for the parts, then do a single round of part training using the complete part feature set as in the original method, then finally sparsify each part. This allows the algorithm to determine the best gradient orientations and texture bins to use for the part representation. Once initialized, the usual WT training method is followed.

Figure 5.7 shows the Average Precisions for different percentages of nonzero weights assigned to the root and part filters of single root / eight part WT detectors for the VOC2006 person and cat classes. In both cases, irrespective of the root sparsity, quite sparse (12.5%) part detectors give the best performance, and adopting this, the sparsest root setting (6.5%) gives the best performance. (The corresponding APs for dense roots are respectively 38.0% and 21.2%). Thus, in the experiments below our default method assigns a somewhat higher percentage of nonzeros to the parts than to the root – typically 1.5–2 times higher[5]. As already observed, this gives good results with fast training times

[5]Given that the part filters also run at twice the resolution of their root and thus have many more

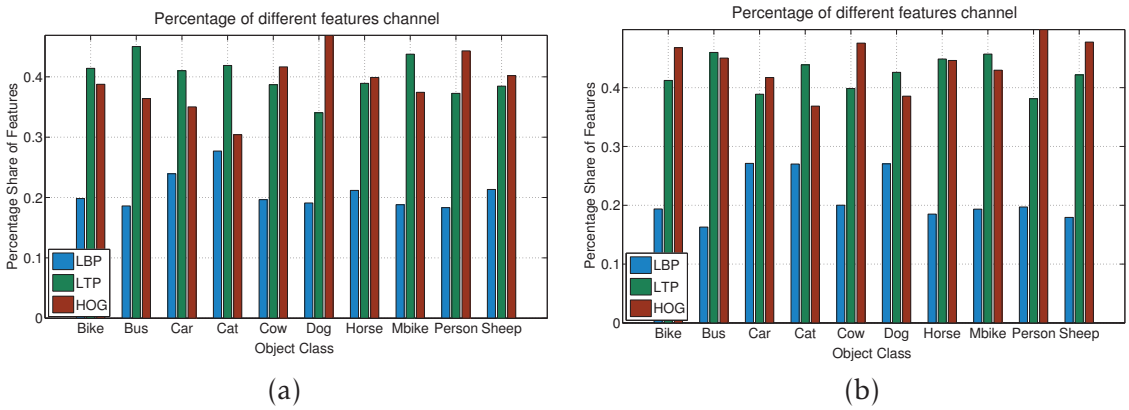


Figure 5.8: *Numbers of features contributed by the different feature channels to 10% nonzero sparse classifiers for the different classes of the VOC2006 dataset. (a) for IWT classifiers. (b) for WT classifiers.*

– *e.g.*, six root eight part detectors with the roots and parts having respectively 10% and 20% nonzeros give 53.4% Mean AP on VOC2006 compared to 54% for dense detectors, while being 1.5–2.5 times faster.

Discussion

Although SVM weight truncation may seem rather ad hoc, it is computationally efficient and it gives much better results in our experiments than every other sparsification method that we tested. It allows our detectors to be sparsified down to ∼10% nonzeros with little or no loss, and in some cases even a small gain in accuracy. The speed/performance trade-off can easily be adjusted by varying the truncation threshold.

The sparsified detectors also give insight into the properties that are important for characterizing the various object classes. For instance for the VOC2006 cat class, most of the features selected come from the LTP channel so texture is probably important for this class, whereas for the car class most of the selected features are from the HOG channel so shape information is probably important – *c.f.* Figure 5.8(b). Figure 5.9 compares sparse FWT models learned for several object classes with dense ones[6]. Compared to the dense models, the sparse ones are relatively clean with little clutter. In this sense they contain the minimum information needed to represent the object category. Additional results for FWT, IWT and WT on multi-root multi-part detectors are given in Chapter 6.

features and finer stepping than it, this means that most of the computation time is spent evaluating part scores.

[6]The HOG displays use the visualization method from [Dalal and Triggs 2005; Felzenszwalb et al. 2009]. For the LBP/LTP ones, in each cell we display a circle of 8 sectors corresponding to the 8 bits of the local pattern, where the intensity of a sector is the weighted sum of the SVM weights of the uniform patterns in the cell that have a '1' bit in that sector. As with HOG, we display positive and negative SVM weights separately, and for LTP we display only the positive half of the split uniform coding (the images for the negative halves are similar). These displays are only a heuristic aid to visualizing the image gradients and patterns that are most important for the discrimination of positives from negatives.

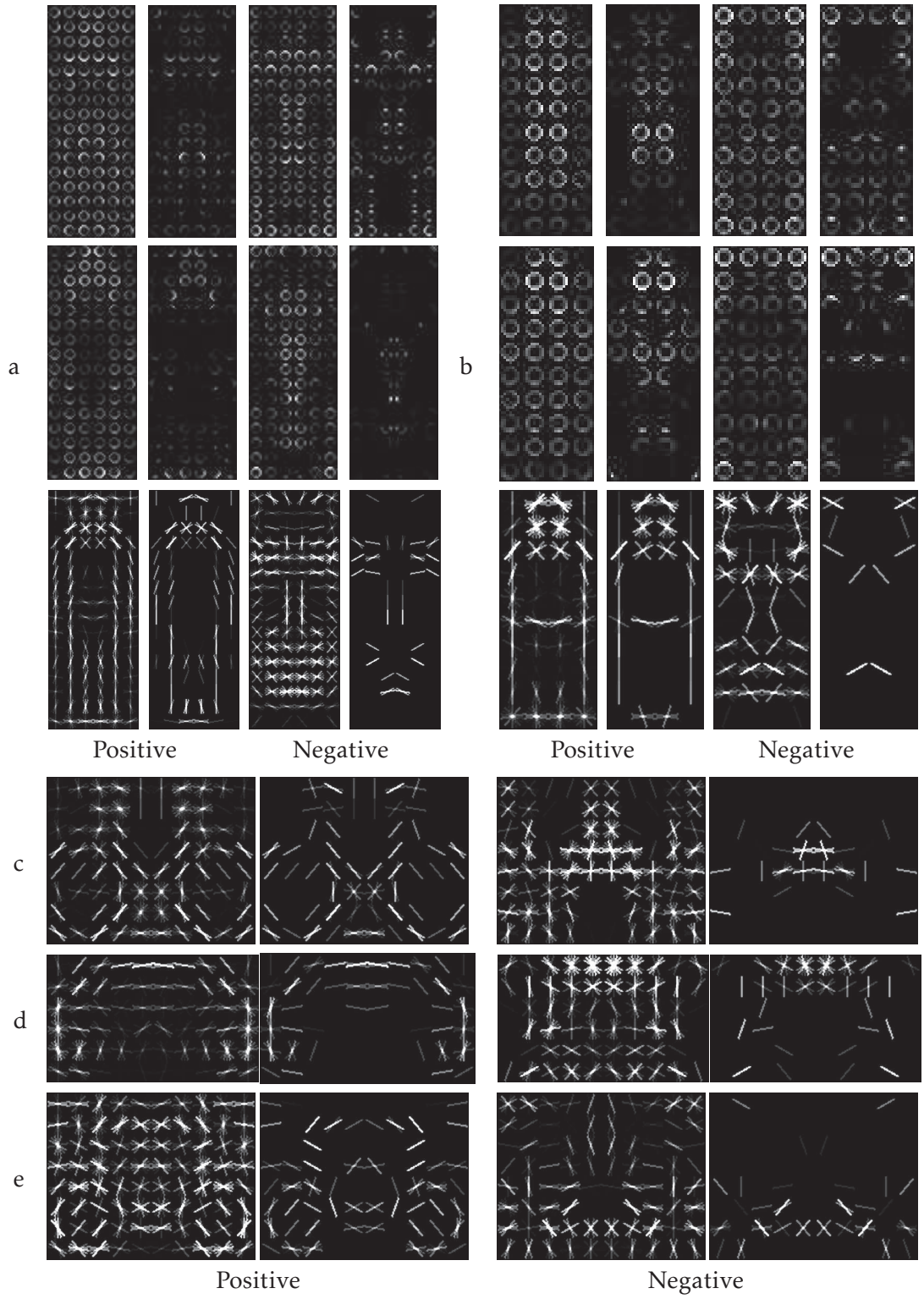


Figure 5.9: *Images of the weights of LBP+LTP+HOG models learned using the dense and 10% nonzero sparse FWT algorithms. Each panel shows the dense and sparse results side by side, with positive weights shown separately from negative ones: images of weights of LBP, LTP and HOG channels for INRIA Person (a), and VOC2006 person (b), and images of weights of only HOG channel for bicycle (c), car (d), and motorbike (e) sets. Note the extent to which sparsification simplifies the positive weights and suppresses background clutter from the positive and especially the negative ones.*

5.2 Nonlinear Classifiers

Although the linear classifiers described above do remarkably well, it is reasonable to expect nonlinear ones to do even better, particularly in the later stages of training where it becomes increasingly difficult to separate the positives from the hard negatives with a linear decision boundary. Currently the most commonly used nonlinear classifiers are kernel SVMs. Most of the common kernels (*e.g.* Gaussians) essentially have the effect of localizing comparisons in feature space, allowing nonlinear decision boundaries to be built in this space. Unfortunately, kernelized classifiers tend to be slow as each example needs to be compared with a large number of support examples, and additional layers such as kernel choice or hyperparameter selection often make them prohibitively slow to train for object detection. For example, even in a PLS reduced feature space, a typical Gaussian kernel SVM detector takes around 30 seconds to scan an image from the VOC2006 test set. In the unreduced 20488-D feature space, the analogous classifier would be hundreds or thousands of times slower again, and even a "fast" nonlinear method like FastIKSVM [Maji et al. 2008] takes around 80 seconds per image[7]. Many researchers have proposed clever methods for speeding up kernel classifier training and evaluation, but training with a complete set of parameters can still involve weeks of computation on a cluster of machines [Vedaldi et al. 2009].

Moreover, although the gains that are achievable by kernelizing of the kinds of feature sets used here are well documented [Dalal 2006], they are relatively modest given the increase in complexity and run time, and they also appear to be fairly predictable in the sense that major surprises have been rare. For these reasons we will not study kernel classifiers further here, instead focusing on more explicit nonlinear mappings that can hopefully provide some of the advantages of kernelization without its prohibitive costs. In fact, most of the methods that we study make a separate nonlinear mapping of each feature dimension, followed by linear classifier training on top of these.

5.2.1 Nonlinear Feature Extension

In this section we cover methods that try to enhance a feature set by including additional features that are simple nonlinear functions of the existing individual input features, while still keeping the overall linear classification framework and hence the possibilities of PLS reduction, sparsification, *etc*. As a simple example, for each input feature w_i we could include both w_i and w_i^2 in the SVM feature vector. This would allow the method to learn diagonal quadratic decision boundaries (coordinate aligned ellipsoids) as well as linear ones. In fact, spherical boundaries (as in one-class SVM, *etc*.) are obtainable from a single additional feature $\sum_i w_i^2$.

[7] This is on the original feature space. Standard IKSVM can not be used in PLS reduced feature spaces because PLS features are not guaranteed to be positive.

We will return to the important quadratic case below, but first we present a general framework for such componentwise feature space extensions. The input features $\{w_i\}$ are replaced by a larger set of scalar functions $\{b_{ij}(w_i)\}$, so the final classifier has the form $f(w) = \sum_i \sum_j \alpha_{ij} b_{ij}(w_i)$, *i.e.* each feature w_i is essentially replaced by a learnable nonlinear function $f_i(w_i) = \sum_j \alpha_{ij} b_{ij}(w_i)$ defined by the basis functions $b_{ij}(w_i)$. Typically, the identity function w_i itself will be representable within this basis but the nonlinear extension will be useful only if the functions learned are actually nonlinear.

This kind of approach can be related to other work in the learning and recognition literature. In particular [Maji et al. 2008; Maji and Berg 2009; Vedaldi and Zisserman 2010] give methods for expanding additive and homogeneous kernels (including Intersection, Hellinger, and Chi-squared kernels) in terms of nonlinear functions of their feature dimensions. These can be applied to any classifier with a homogeneous kernel, and by approximating the basis functions one can build fast approximations of the kernel function. For example, Vedaldi and Zisserman [2010] approximate additive kernels by dot products of vectors of basis functions, which are in turn approximated to give closed form rules allowing rapid kernel evaluation. In such methods, the final classifier has a form analogous to the one above in the sense that it can be expressed as a linear SVM of basis functions, each of which is a simple function of one or a few of the input features, possibly with an additional normalization term.

5.2.1.1 Piecewise Feature Extension

Piecewise SVM. As an example of componentwise extension, consider the approach of Maji et al. [2008]. Although this was originally presented as a method for accelerating Intersection Kernel classifiers, it can also be viewed as a method of learning general componentwise nonlinear mappings $f_i(w_i)$ such that the final classifier $\sum_i f_i(w_i)$ has good performance[8]. Given that viewed independently, the componentwise mappings learned by [Maji et al. 2008] are extremely nonlinear, we wanted to see whether such nonlinearities arise in other componentwise approaches.

This can be achieved by discretizing each axis (feature dimension) into a number of bins and taking the $b_{ij}(w_i)$ to be bin indicator functions. However even with bilinear interpolation between bins, this method gives poor results because it introduces a very large number of bins, each with an independent SVM weight. A smoother but denser representation can be obtained by using a "bar-chart" encoding, where each feature w_i is encoded by a binary sequence, $\{1, \ldots, 1, x, 0, \ldots, 0\}$ where the number of 1's (and the fractional residual x) encodes the number of bin boundaries crossed. This forces the SVM weights to be effectively deltas from one bin to the next, thus encouraging

[8] However it should be pointed out that the SVM regularization implied by Maji et al.'s kernelized formulation is non-obvious from a componentwise point of view.

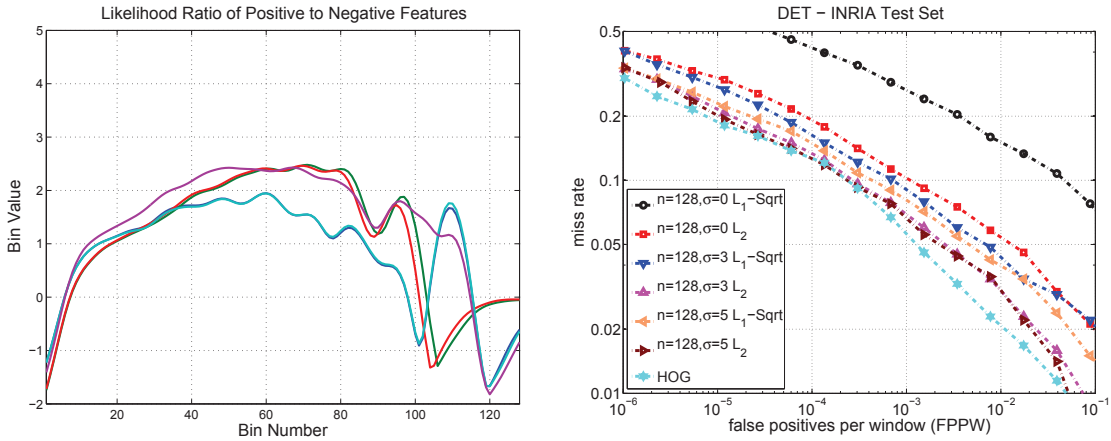
the learned componentwise functions to be smooth. However we found that even this representation leads to significant overfitting in our experiments, with the learned nonlinear functions behaving rather erratically. For instance for HOG_{36} features on the INRIA Person dataset at 10^{-4} FPPW, using a 10 bin indicator function representation for each of feature dimension instead of the usual linear one increases the miss rate from 20% to 27%, while a 10 bin bar-chart representation increases it by 3%. Given that such high dimensional encodings are slow to learn and use and that they do not appear to outperform linear ones, we can not recommend this approach for problems like ours.

Likelihood Ratio Features. We also tested a related approach based on componentwise likelihood ratios. Here, each feature is replaced by the log likelihood ratio of its value arising on the positive versus the negative class, $\log\left(\frac{\Pr(Positive|feature)}{\Pr(Negative|feature)}\right)$, then a linear SVM is learned over these features. The intuition is that these particular nonlinear functions are likely to be especially useful given that a Naive Bayes classifier would simply sum them.

Specifically, for each feature a b bin histogram of the feature values over the positive and negative training examples is built. These histograms are then normalized either using L_1-Sqrt or L_2 normalization (*c.f.* Sec. 4.2.4.4), smoothed by Gaussian filters, and the corresponding log-likelihood ratio lookup table is built. In use, each feature dimension is simply mapped through its lookup table using bilinear interpolation, then a linear classifier is trained and run on the mapped features. Figure 5.10(a) shows the smoothed log-likelihood ratio histograms for the five features that have the largest magnitude SVM weights with HOG features on the INRIA Person dataset. Interestingly, the plots are highly nonlinear and in each case the ratio drops and becomes erratic for high values of the input feature. This is probably due to the comparatively small number of positives and the way that the HOG features are normalized and clipped. The results in Figure 5.10(b) confirm that both L_2 normalization and smoothing are needed to achieve good results, however the quantization can be quite fine. We found that 128 bin histograms together with a smoothing of $\sigma = 3 - 5$ bins gave the best performance. Overall the results given by this method are comparable to, but not significantly better than, those of a linear SVM trained directly on the HOG features. The method may be useful in some problems, but its extra complexity does not seem to be warranted here.

5.2.1.2 Componentwise Quadratic Classifier

The Componentwise Quadratic classifier is a special case of the general componentwise polynomial approach, using a nonlinear extension with just two features per dimension, w_i and w_i^2. In practice we find that it gives almost identical accuracy to IKSVM, and only slightly better accuracy than a simple linear SVM – *e.g.* a 1.5% increase in AP on



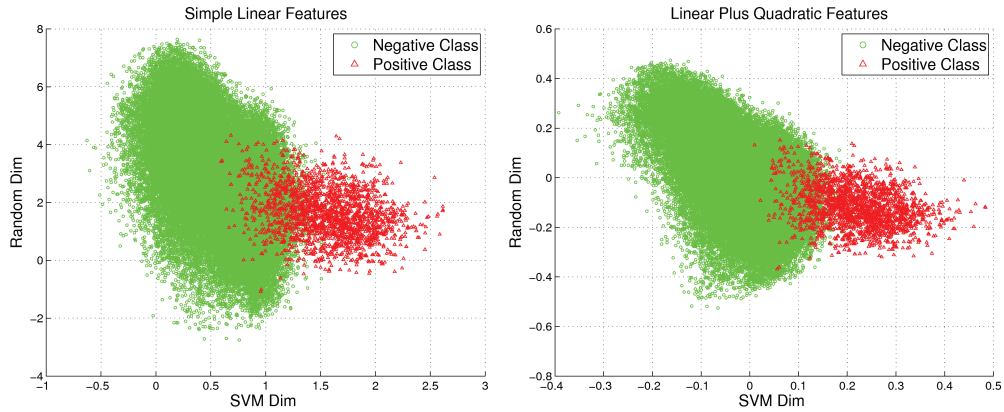
(a) (b)

Figure 5.10: *Results for the log-likelihood ratio features over HOG on the INRIA Person dataset. (a) Log-likelihood ratio histograms of the five HOG feature dimensions with the highest magnitude SVM weights. (b) DET curves for classifiers trained on various versions of the log-likelihood ratio feature set.*

the INRIA Person dataset – in about 4.4 seconds per image, which is much faster than even FastIKSVM [Maji et al. 2008]. Moreover, all of the learned w_i^2 coefficients turn out to be very small, so there is little evidence that the noisy and highly nonlinear 1D functions learned by IKSVM are actually needed for good classification, at least for object detection with the feature sets used here. Experiments with high order componentwise polynomial classifiers (cubic and beyond) confirm these observations.

Componentwise mapping can also be used in the PLS-reduced feature space, and here quite high orders of nonlinearities are feasible. However for our problems and feature sets the benefits of this seem to be quite limited. For LBP+LTP+HOG on the INRIA Person dataset with 20 PLS dimensions, both reduced and unreduced componentwise quadratic classifiers give a 1.7% increase in AP over linear SVM, and adding higher powers of w_i gives little further improvement. On the VOC2006 person class, componentwise quadratics do not give any noticeable improvement in performance.

Overall, the utility of componentwise polynomials and similar nonlinear extentions appears to be rather limited, at least for visual object detection with the currently-popular feature sets. For this reason we remain skeptical that methods that produce highly nonlinear componentwise mappings such as IKSVM will ever be able to significantly outperform linear SVM or simple polynomial feature set expansions of the kind suggested here.



(a) (b)

Figure 5.11: *An illustration of the limited improvement in separations produced in the reduced PLS feature space by the full quadratic classifier learned on the final stage training set (b), relative to the corresponding linear one (a).*

5.2.2 PLS Quadratic Classifier

Another means of introducing a limited amount of nonlinearity while still keeping reasonable control of the overall feature dimensionality and hence the computation time is to build more general nonlinear classifiers on reduced-dimensional feature spaces. Here we consider the case of a full quadratic SVM on the PLS-reduced feature space – *c.f.* [Schwartz et al. 2009]. By full quadratic we mean that as input features, the SVM feature space contains all of the reduced w_i and w_i^2 and also all of the bilinear combinations $w_i w_j$, *i.e.* $\frac{n(n+1)}{2}$ features in total. This allows the linear SVM discriminant to represent an arbitrary quadratic decision surface on the reduced input feature space. This approach is feasible because PLS projections are highly discriminant even with quite small numbers of dimensions. For the datasets tested here, 14 – 20 PLS dimensions (and hence 105–210 SVM dimensions) turn out to give the best results on the validation sets. The relative weighting of the linear and quadratic components affects the performance, but for good performance we found that it sufficed to normalize the range of each to unity.

Adding the quadratic features leads to a slightly better separation between the classes and it also reduces the number of hard negatives found, which in turn reduces the asymmetry between the positive and negative examples in the later stages of training. However empirically, the resulting "*FullQuad*" classifier gives only a modest improvement in performance over the linear one. For instance for HOG features it improves the AP on the INRIA Person dataset by 2.3%, and by 0.8% on the VOC2006 person class. Figure 5.11 illustrates the limited increase in separation achieved by training the PLS full quadratic classifier on the final stage training set obtained by running the corresponding linear classifier through the complete set of latent cycles. Here the vertical axes are just random dimensions chosen to be orthogonal to the SVM projection.

Overall, the PLS full quadratic method has the best accuracy of all of the detectors that we tested on these datasets, giving similar results to IKSVM and Gaussian kernel SVMs. Still, considering the increase in computational cost, the improvements in performance over the basic linear detectors are not very substantial. Moreover, the quadratic PLS classifiers are slower than the corresponding linear ones because back-projection to the original feature space can no longer be used to avoid the need for explicit projection to the reduced feature space at run time and because after projection the features must be then re-expanded to the quadratic feature vector. *E.g*. *FullQuad* requires about 5.7 seconds per image on the VOC2006 dataset, whereas the linear detector requires only 2.8 seconds.

5.2.3 Cascade of Linear and Nonlinear Detectors

To speed up the PLS quadratic approach we can adopt a two-stage cascade strategy. We tested two variants of this. In the first approach reduced quadratic classifiers are learned at each stage of the latent training cycle, but after the final stage an additional linear SVM pruning classifier is trained to remove as many negatives as possible while keeping 95% of the positives. This is then used as a pre-filter before running the quadratic classifier. This '*FullQuadCascade*' approach effectively reduces the testing time to about 4 seconds per image without changing the overall accuracy of the detector. Note that this training strategy actually gives slightly better results than a detector trained using the full Linear+Quadratic cascade at each stage of latent training, presumably because it is not a good idea to prune hard negatives until one knows exactly which ones can best be pruned.

In the second approach, a classical linear Latent SVM is trained (using PLS) through the full cycle, then a single PLS based quadratic classifier is trained as a post-filter on the final-stage hard negatives that pass the resulting linear SVM. This also works, but the results are less accurate than the first approach. For instance, the second approach improves the AP by only 0.3% for the VOC2006 person class, whereas the first one improves it by 0.8%.

5.2.4 Discussion

In general, on these datasets we find that the nonlinear detectors studied are slightly more accurate than the corresponding linear ones but that the differences are small. In particular, the hard negatives found in the later stages of training do not appear to have systematic properties that yield easily to the nonlinearities tested here. Nor is it safe to assume that the hard negatives found by one method (*e.g*. linear) will be similar to those found by another (*e.g*. nonlinear), so the full latent SVM training cycle generally needs to be run for the nonlinear detector. It is also important to note that on the datasets

tested, nonlinear extensions such as quadratic classifiers in the reduced space can easily overfit so validation is needed to find the optimal number of reduced dimensions. Even after validation, although it is generally true that in later stages both the number of hard negatives and the training error become smaller when nonlinear classifiers are used, this does not always lead to corresponding improvements in performance on the test set.

5.3 Summary

In this chapter we described a number of different approaches to training linear and nonlinear classifiers for object detection. The emphasis was on practical approaches that are fast to train and test while at same time giving close to optimal performance.

Although SVM training times scale linearly with the number of examples, for high dimensional feature sets this still leads to slow training, especially in the later stages of the training cycle. We showed that incorporating the discriminative dimensionality reduction method Partial Least Squares provides much faster training for both linear and some nonlinear object detectors without any corresponding reduction in accuracy. We also investigated various feature selection methods for reducing the number of features needed in the detector, concluding that simple truncation of SVM weights considerably outperforms the other methods tested (including some apparently much more sophisticated ones), allowing feature vectors to be reduced to 10-15% of their original sizes with negligible losses in precision.

Linear classifiers are the default choice for our detectors. We investigated several methods for introducing a degree of nonlinearity using explicit feature mapping while remaining much faster than conventional kernel classifiers. The resulting methods provided slight improvements in overall accuracy at a limited cost, without seriously challenging the predominance of linear methods in these applications. It would be interesting to see how these classifiers perform in other settings and to compare them with methods based on closed-form feature mapping such as [Vedaldi and Zisserman 2010].

Chapter 6: Experiments

This chapter provides a detailed experimental evaluation of our approaches. For completeness, we begin by briefly recapitulating the standard configurations and settings that are used for detector training and evaluation. We then provide experimental results and comparisons with state-of-the-art methods on a number of publicly available benchmark datasets including the INRIA Person dataset [Dalal and Triggs 2005], several PASCAL Visual Object Challenge (VOC) object detection datasets [Everingham et al. 2010b], and the ETHZ Pedestrian dataset [Ess et al. 2007, 2008]. The most thorough testing is done on the INRIA Person and VOC2006 datasets. Each section provides a brief introduction to its dataset followed by a performance analysis of various detector configurations on it and a comparison of our results to state-of-the-art methods from other authors. The discussion covers all forms of the detectors that we have developed, and also analyzes the effect of each detector component on accuracy and speed.

6.1 Parameter Settings and Detector Configurations

We test a number of detector configurations, ranging from a simple non-latent single root detectors to multi-component multi-part latent ones. This section summarizes the configurations tested and the parameter settings used. For more details, see chapters 3-5. As discussed in Sec. 3.6, we use mainly Precision-Recall and their Average Precisions (APs) to quantify detector accuracy, occasionally using DET curves or Recall-False Positive Per Image ones when this makes comparison with previous work easier.

Feature Set. By default we use a feature set obtained by simple concatenation of normalized LBP, LTP and HOG_{31} histograms. All features are computed over 8×8 RGB pixel cells with the color channels merged at the histogram level. Bilinear interpolation is used to distribute votes among neighbouring cells. L2-hysteresis normalization is used for HOG and L_1-Sqrt normalization for LBP/LTP. No image preprocessing is done before feature extraction (*c.f.* Chapter 4).

Multiple Components and Parts. We tested several different forms of detector. Our non-latent single root detectors are trained using the direct method of Sec. 3.2, while all forms of latent, multiple component and parts-based detectors are trained using the latent method of Sec. 3.3.2. Our datasets do not have component-level labellings so we initialize the components by using the aspect ratios of their annotation boxes as surrogate component indicators [Felzenszwalb et al. 2009, 2010b]. Parts are also initialized using the procedure of Sec. 3.3.2. All of our detectors enforce bilateral symmetry, using either folding (typically only used in the single root case) or mirrored pairs of components – see Sec. 3.4.

SVM Settings. By default, our single root (latent or non-latent) detectors use linear classifiers trained with SVMLight [Joachims 1999; Dalal and Triggs 2005], while our multiple component and/or parts based ones are trained using the LatentSVM implementation of Felzenszwalb et al. [2009] – the later gives slightly better performance in this case and it is more efficient owing to its specialized cost function and implementation. In either case, PLS dimensionality reduction and/or weight truncation based sparsification can optionally be included.

Empirically, we find that setting the SVM regularization parameter C to the inverse square of the average norm of the training feature vectors, $C{=}(\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}\|)^{-2}$ [Joachims 1999], gives near-optimal results for the root-only detectors trained using SVMLight, and thus eliminates the need for a cross-validation cycle during training. Similarly, we follow Felzenszwalb et al. [2009] in fixing C=0.002 for all of the LatentSVM based multi-component and/or parts based models. Our single root detectors were trained with positive to negative SVM error weighting J=3, while the part based and multi-component ones were trained with J=1 because their better accuracy reduces the need for asymmetric weighting.

Dimensionality Reduction. We tested both Partial Least Squares subspace projection and Weight Truncation based feature selection. In both cases, each root and each part filter is reduced in dimension separately, while the part displacement cost features are left unreduced. In the PLS case, we limit the range of the possible part displacements by setting their quadratic displacement penalties to infinity outside of a given range as this gives slightly better results. For sparsification, we tested initial (WT), iterative (IWT) and final (FWT) Weight Truncation – *c.f.* Sec. 5.1.2.3. Recall that the first runs sparse feature selection only once (at Stage II of latent learning), the second runs it in each round of the training and the third runs it only after full dense detector training. In the experiments given here, the part filters are by default assigned a percentage of nonzeros that is twice as high as that of the roots. When "percentage nonzeros" figures are given, they are totals over the whole part and root feature set.

(a) (b) (c) (d) (e) (f)

Figure 6.1: *Some examples of cropped images from the INRIA Person training set. Here, the internal* 32×96 *pixel box surrounds the original person, the* 16 *pixel margin around it (yellow box) is added to include some contextual information, and the additional* 16 *pixel margin (green box) around that is allowed to ensure undistorted feature calculation and interpolation. (a - d) Two examples of cropped images and their boxes. (e, f) The mean of the images from the cropped positive training set.*

6.2 Results on the INRIA Person Dataset

The INRIA Person dataset was introduced by Dalal and Triggs [2005] to study the problem of pedestrian (standing or walking human) detection. Although it is limited to one class and views in which the whole person is visible and comparatively unoccluded, it remains a moderately challenging dataset that has been used by many authors including [Zhu et al. 2006; Tuzel et al. 2008; Lin and Davis 2008; Wu and Nevatia 2008; Schwartz et al. 2009; Wang et al. 2009; Lin et al. 2009; Dollár et al. 2010; Ito and Kubota 2010; Gerónimo et al. 2010; Gualdi et al. 2010; Vedaldi and Zisserman 2010; Bar-Hillel et al. 2010]. The positive training set contains 641 images with 1208 annotated instances of pedestrians while the negative training set contains 1218 images of similar scenes with no pedestrians. Similarly, the positive test set contains 288 images with 589 annotated instances while the negative test set contains 453 images with no instances. As a convenience, the INRIA dataset is shipped with sets of pre-cropped positive windows for the training and testing of plain (non-latent) classifiers. A corresponding set of negative training windows must still be sampled by the user. Each training window has size 96×160 pixels, where the innermost 32×96 pixels contain the person. A surrounding 16 pixel border is included to provide 64×128 windows that allow some of the visual context around the person to be captured if required, and an additional surround of 16 pixels (3 pixels in the case of the test set) is included to allow for feature computation and histogram interpolation without introducing boundary effects – see Figure 6.1. When evaluating overlaps or drawing detection windows, only the central 32×96 pixel region containing the person is used. To increase the number of positive examples, the training and test sets are traditionally augmented with left-to-right flipped versions of

Features	Non-latent	Latent
HOG	73.1	79.0
LBP	67.8	73.9
LTP	75.7	78.9
LBP+LTP	69.7	80.4
LBP+HOG	74.2	80.7
LTP+HOG	79.2	81.4
LBP+LTP+HOG	76.6	82.8

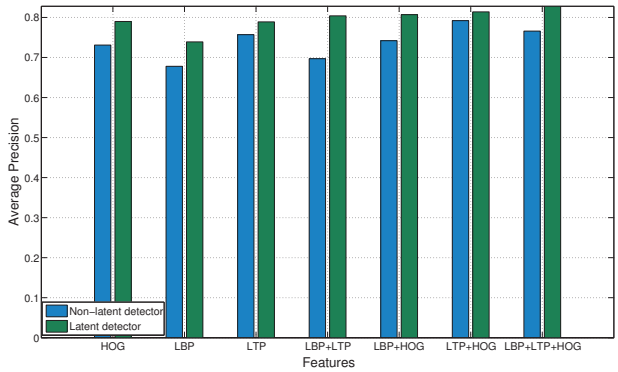


Figure 6.2: *Average Precisions for non-latent and latent detectors trained using various combinations of features on the INRIA Person test set.*

the original examples, but this is redundant if a bilaterally symmetric detector is used.

6.2.1 Non-Latent Detectors

Among the individual feature channels tested on the INRIA Person dataset, LTP gives the best performance for the non-latent detector (75.7% AP), followed by HOG then LBP. Among the feature combinations, LTP+HOG leads the way (79.2% AP) followed by LBP+LTP+HOG and LBP+HOG – *c.f.* Figure 6.2. The LTP+HOG figures are better than the results of [Ott and Everingham 2009; Ramanan 2007], who report respectively 75.7% and 77.4% AP for linear methods that use additional segmentation cues to achieve better performance. Ott and Everingham [2009] combine conventional HOG_{36} features with HOG_{36} features computed on the foreground of foreground/background segmented windows, where the segmentation model is learned offline. They also report 78.3% AP for the equivalent method using a quadratic kernel classifier. Ramanan [2007] uses a segmentation model learned offline from extended regions around detections produced by a traditional HOG_{36} detector as a postfilter to reduce false positives.

Figure 6.3 provides window-level DET curves comparing our classifiers with the competing methods of [Dalal and Triggs 2005; Tuzel et al. 2008; Wu and Nevatia 2008; Dollàr et al. 2008; Schwartz et al. 2009; Wang et al. 2009; Ito and Kubota 2010]. Given that HOG was already a significant improvement over previous feature sets when it was introduced, this illustrates the extent of progress in feature sets for object detection over the past five years. Our window-level classifier achieves 3.5% miss rate at 10^{-5} FPPW. In comparison, [Schwartz et al. 2009; Wang et al. 2009; Ito and Kubota 2010] report respectively 5.8%, 5.6% and 5.6% miss rate[1]. These methods resemble ours in the sense that they use combinations of heterogeneous feature channels combining color,

[1] Unfortunately, Precision-Recall or Recall-False Positive Per Image results are not available for all these detectors so our comparison with them is limited to DET curves.

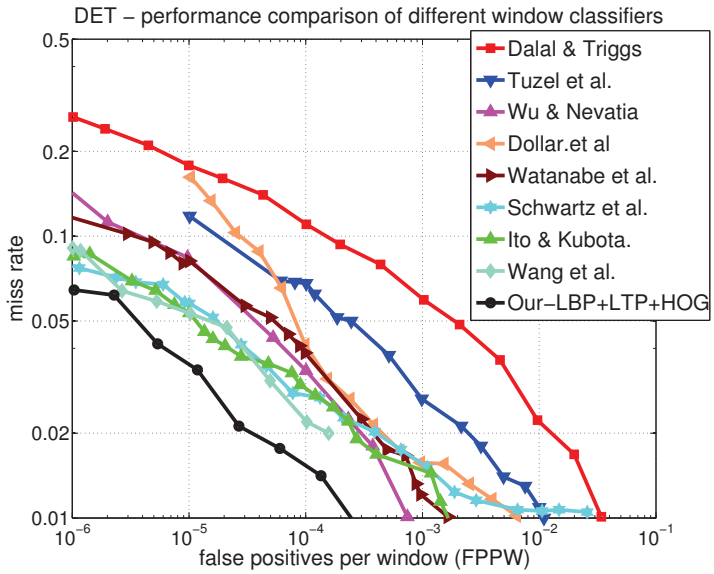


Figure 6.3: *A comparison of the performance of various non-latent window-level classifiers on the INRIA Person test set. Several of the curves were traced from the original publications [Tuzel et al. 2008; Wu and Nevatia 2008; Dollàr et al. 2008; Schwartz et al. 2009; Wang et al. 2009; Ito and Kubota 2010].*

shape and texture cues. Schwartz et al. [2009] use a PLS based Quadratic Discriminant classifier over a very high dimensional feature set (170820 features for 64×128 windows) obtained by concatenating multi-block HOG [Zhu et al. 2006], co-occurrence matrix texture [Haralick 1979] and HSV color histogram features. Wang et al. [2009] use a combination of HOG and LBP features with explicit occlusion modeling in a linear SVM classifier. Ito and Kubota [2010] use a combination of linear and embedded quadratic classifiers on extremely high dimensional feature vectors obtained by combining co-occurrence histograms of oriented color gradients, pairs of edge orientations and color differences with color histograms in Cb-Cr color space. Our feature set is an order of magnitude smaller than [Schwartz et al. 2009; Ito and Kubota 2010], and it is faster to evaluate and gives better results than all three. For instance, it is around 5 times faster than [Schwartz et al. 2009].

However we again emphasize that tests on window-level classifiers are not a fully reliable guide to the performance of complete detectors. For example, note that although our window-level classifier over LBP+LTP+HOG features has a ∼5× lower miss rate at 10^{-5} FPPW than our HOG classifier (*c.f.* Figure 6.3), the corresponding complete detector has an AP only 3.5% higher (76.6% versus 73.1%). Similarly, in DET plots the LBP based window-level classifier has better accuracy than the HOG one, but the situation is reversed for the APs of the complete detectors. For this reason we will concentrate on complete detector metrics such as AP from now on.

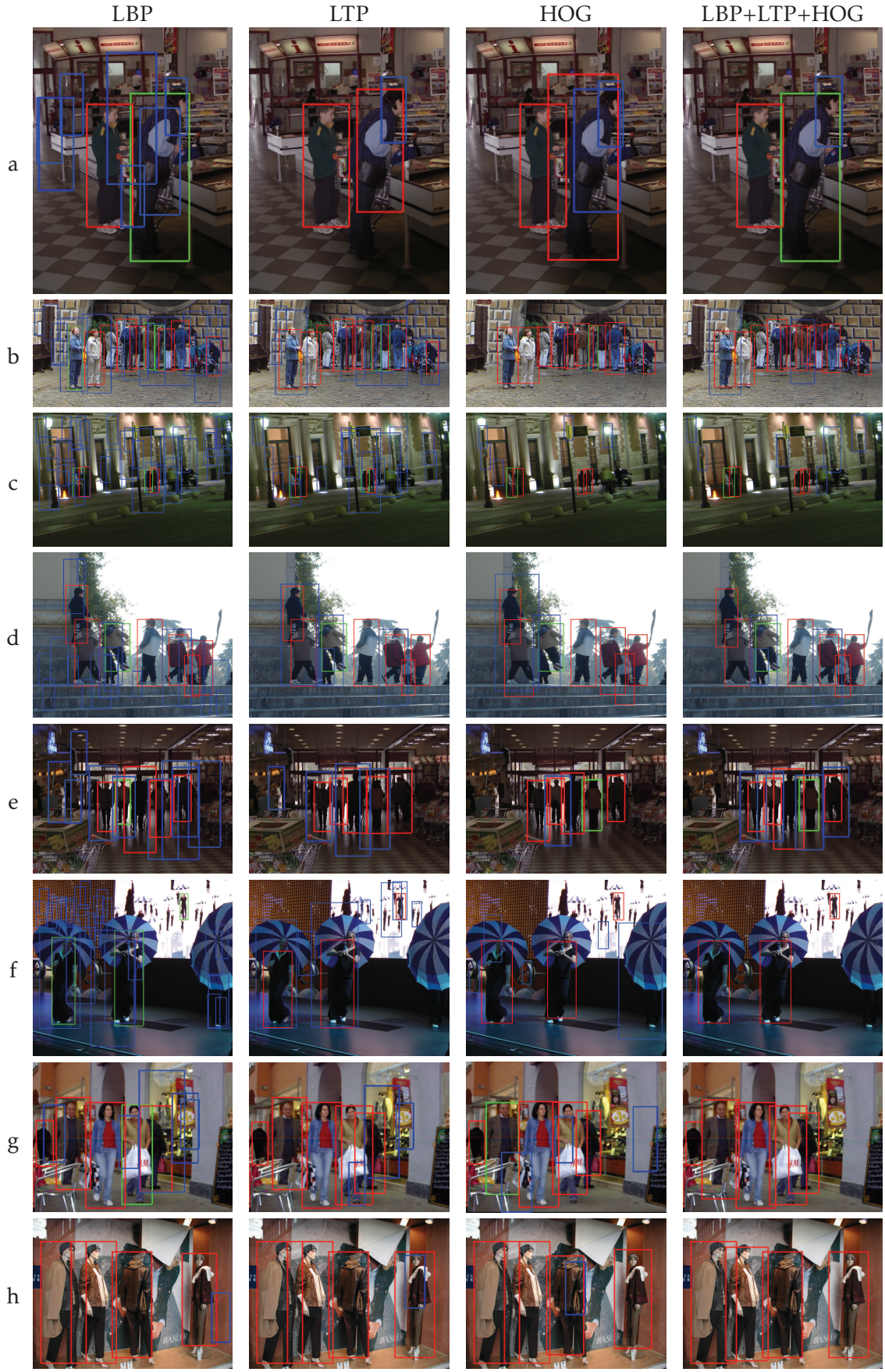
6.2.2 Latent Detectors

Figure 6.2 compares the Average Precisions of latent and non-latent single root detectors for various feature sets and combinations. The only difference between the latent and non-latent detectors is the way that they are trained. Latent training leads to significant improvements in accuracy for all of the configurations tested. Among the individual feature sets, LBP shows the largest improvement with a 6.2% increase in AP, followed by HOG and LTP with respectively 5.9% and 3.2%. Among the feature combinations, LBP+LTP shows the largest improvement with a 10.7% increase, but the improvement decreases as the feature set becomes stronger and for LTP+HOG only a 2.2% increase is observed. This is perhaps due to near-saturation of the method for these features and dataset. The latent search over position and scale also seems to allow the detector to find a better registration with the data during training, leading to cleaner, more-sharply defined root filters – *c.f*. Figure 3.1.

Multiple roots and/or parts provide only modest improvements in AP on the INRIA dataset, perhaps because in contrast to the VOC datasets below where humans appear in a wide range of poses and occlusion states, most of the subjects in the INRIA dataset are in largely unoccluded and upright poses. Replacing the single folded root detector with a two root mirrored pair one increases the AP from 82.8% to 83.1%. A 2Root×8Part detector with default settings (C=0.002) gives 85.9% AP, but this figure is lowered by significant overfitting for this configuration on this dataset. If we use sparsity as a regularizer, sparsifying the 2Root×8Part detector to 10% nonzero weights using iterative sparsification (IWT) and training with C=6.7×10^{-4} improves the AP to 87.3%. This performance is still below the current state-of-the-art of 88.2% [Felzenszwalb et al. 2009], using HOG_{31} features alone in the same configuration. We have confirmed the 88.2% result ourselves for HOG_{31} with the default C of 0.002. However we find that the results are sensitive to the exact C value. In general it seems that for HOG_{31}, LBP+LTP+HOG and similar feature sets, there is a considerable risk of overfitting with multi-part configurations on the INRIA Person dataset. This is seen in the sensitivity to C, and also in the increase and the variability of the number of the training iterations needed for convergence.

6.2.3 Discussion

Figure 6.4 shows some examples of detections on the INRIA test set for latent detectors with various combinations of features. To provide more meaningful comparisons, all of the detectors were tuned to a recall rate of 85%. These examples give insight into the characteristics of the different feature channels. The LBP features have a high false positive rate at this recall, firing on many regions that have either strong texture or partial resemblances to human shapes (rows c, d, e, f). The LTP and HOG features



continued on next page

Figure 6.4: *Some examples of detections from our latent single root detectors on the INRIA Person test set, for various feature sets. The red boxes indicate true detections according to the PASCAL VOC criteria, the blue boxes indicate false positives, and the green boxes indicate the ground truth annotations. Notice the similarity of the responses from HOG and LTP (although LTP is able to detect some instances missed by HOG – rows f, g, k), and the fact that the combined feature set significantly improves the results.*

have similar overall performances, with some cases where LTP corrects false positives from HOG (rows e, g, k). LTP and HOG both appear to be strongly resistant to lighting variations, giving good performance in under- and over-exposed conditions (rows a, c, e, f). As expected, incorporating all three channels to give LBP+LTP+HOG improves the performance under all conditions.

A more through examination of the errors on the test set reveals the main sources of problems. Detections can be missed either because the classifier fails to fire – which happens most often when the subjects appear at very small scales or at the borders of the image, in unusual poses, under challenging lighting conditions or with significant occlusions – or because overlapping detections from nearby subjects are incorrectly merged by the non-maximum suppression method – *i.e.* in crowded scenes. Moreover, even at high recall rates, almost all of the false positives occur on images from the positive test set, not on the negative one. In fact, most of them occur on the bodies of real subjects at significantly finer scales than the true annotations, thus escaping the mode

merging algorithm. Other notable sources of false positives are overlapping subjects or crowds, complex structures with strong vertical edges such as lamp poles, trees, *etc.*, and uncommon textures on human clothing.

This association of false positives with positive images can be explained as follows. The negative training set appears to be reasonably representative of the negative test set so that training on hard negatives from the former makes the detector resistant to false positives on the latter. However, the negative training set does not contain any fragments of bodies or clothing and it may not be fully representative of background structures that are common in the positive images, so the detector has not learned to reject these – *c.f*. Figure 6.5.

Clearly this is essentially a training issue not a dataset one: we have simply failed to exploit the negative information that is present in the positive training images. To work around this, we could retrain the detectors by including false positives from the positive images as new hard negative examples, where a detection is labelled as a false positive if either it does not overlap an annotation or the degree of overlap with the closest annotation is less than a prespecified threshold. Unfortunately, as Figure 6.5 shows, there are many missing annotations on the positive INRIA training images, which would lead to many correct but unannotated detections being incorrectly labelled as false positives. One could perhaps handle this by rewriting the SVM code to give such hard negatives lower weights (SVM hinge loss slopes) during training, but here we test a partial fix that does not require either a code rewrite or reannotation of the dataset. As hard negatives from the positive images, we include only the false positives that overlap an annotation by more than 0% and less than 30%, simply discarding ones that have zero overlap. This has the effect of including most of the false positives that fall on body parts as negatives, without including the many incorrectly-false positives produced by the missing annotations. A final stage of retraining with these additional hard negatives increases the overall accuracy of most of our detectors. For instance, it improves the latent LBP+LTP+HOG detector's precision from 70% to 75% at 85% recall rate, and its overall AP from 82.8% to 83.8%. Likewise, it improves the AP for LBP and LTP by respectively 0.8% (74.7% versus 73.9%) and 1.2% (80.1% versus 78.9%) although interestingly, it provides no improvement for HOG_{36} features alone. Similar improvements are seen on some other classes, *e.g*. for the VOC2006 person class, including the extra hard negatives improves the AP from 34.4% to 35.3% for LBP+LTP+HOG features, and from 28.9% to 29.3% for LTP ones, while it reduces the APs of both HOG_{36} and LBP features by about ~4%. Because of these inconsistencies, we cannot currently recommend including hard negatives from the positive training images as a general practice, but it remains useful in some cases and the development of more advanced techniques to use the negative information that is contained in (potentially incompletely annotated) positive images is an area that warrants further attention.

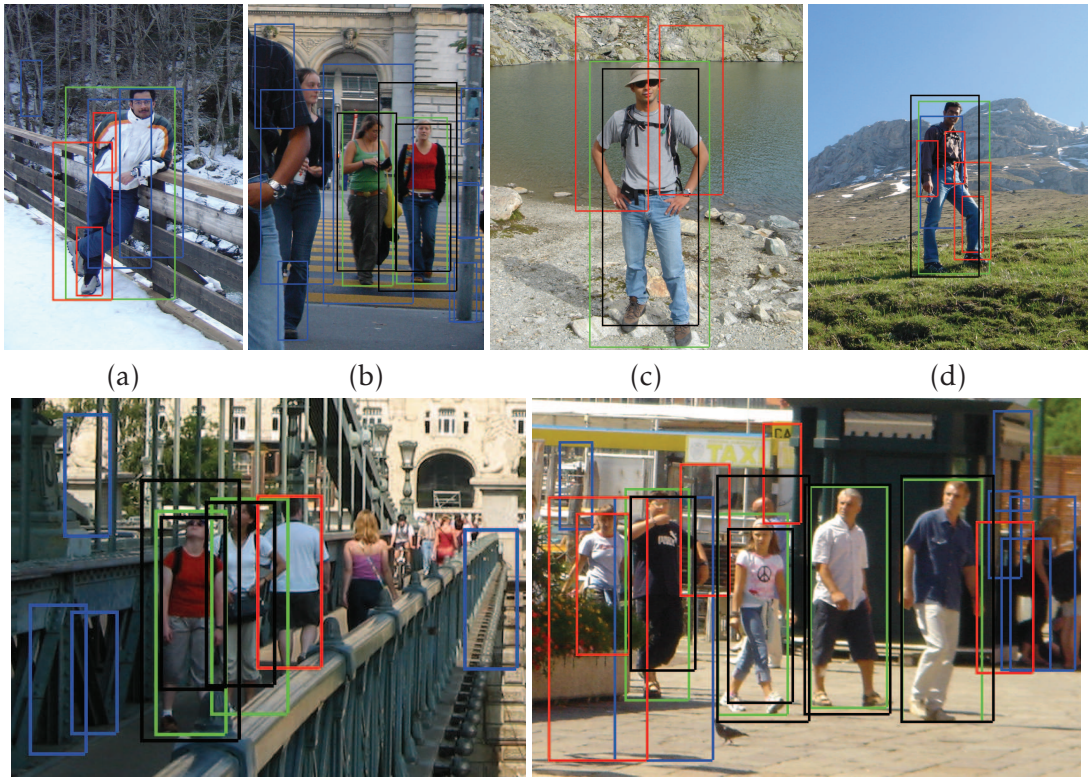


Figure 6.5: *Some examples of responses of our latent single root detector at 95% recall on the positive INRIA Person training set. The green boxes indicate the original annotations, the black boxes indicate detections having at least 50% overlap with a true annotation box, the red boxes indicate detections having overlap* $> 0\%$ *and* $\leq 30\%$ *with a true annotation box, and the blue boxes indicate other false positives.*

Another possible means of reducing the incidence of false positives on positive images is contextual modeling, where scene geometry and/or easy-to-detect examples are used to estimate additional cues such as ground planes [Park et al. 2010], vertical surfaces, *etc.*, which are then used to discard false positives whose scales or positions are implausible. A segmentation based post-processor Ramanan [2007] could also be used to reject low confidence false positives. However even with these methods, improving the use of the negative information from the positive training images seems to be a useful first step.

Our single root LBP+LTP+HOG detector takes around 3.4 seconds to process an INRIA Person test image. In comparison, our HOG, LBP and LTP detectors take respectively around 0.7, 1.7 and 2.6 seconds. The LBP and LTP detectors are slower than HOG because their feature sets are higher dimensional, making dot products slow to evaluate, and because the pixel-level bilinear interpolation that is needed to sample a circle from the neighbourhood of each pixel slows the feature computation.

Figure 6.6: *Pixelwise means of the positive training windows for the 5 viewpoint annotations of the 10 classes of the VOC2006 dataset. The integers denote the number of available examples for the given viewpoint. Note that for the majority of the examples the viewpoint is unspecified. The images here have been resized to* 100×100 *pixels so they are not displayed at their true aspect ratios. (Figure courtesy of Tomasz Malisiewicz:* `http://www.cs.cmu.edu/~tmalisie/pascal/means_trainval.html`*)*

6.3 The PASCAL Visual Object Challenge Datasets

The PASCAL VOC datasets [Everingham et al. 2010b] are important visual recognition benchmarks. The VOC has been run annually since 2005 with the aim of establishing the best performing methods and advancing the state of the art in image classification, object detection and semantic segmentation. We use only the object detection datasets. Each dataset contains a training/validation subset '`trainval`' and a test subset '`test`' (which has not been released publicly since VOC2008 – instead a public evaluation server is provided). There are two sections of the object detection competition: in '`comp3`', only the provided `trainval` dataset can be used for training, while in '`comp4`' any dataset (except the test set) can be used. We follow the `comp3` protocol for all of our experiments.

The VOC datasets contain object classes from four high-level groups, vehicles, animals, household objects and people, with each group being divided into subgroups such as two wheeled versus four wheeled vehicles and then into individual classes such as car, bicycle, *etc*. The image annotations specify the class names and bounding boxes of all objects of the designated classes that are present, together with additional information such as viewpoint labellings and information about whether the object is truncated or difficult to detect without help from the surrounding context. For training and evaluation we include both the complete and truncated examples, but we exclude ones labelled as difficult. When training we find that they tend to disturb the learning process owing to their inherent difficulty and the standard VOC evaluation protocol ignores errors on 'difficult' instances so they are left out of the evaluation as well.

The viewpoint annotations specify only the coarse categories front, rear, left, right, with any other viewpoint being labeled as unspecified. In practice most examples are labeled as unspecified. The weakness of these labellings makes learning a separate detector for each viewpoint difficult because: i) intermediate poses between front, left, are common and tend to be labelled as 'unspecified'; ii) for deformable classes like person, horse, cat it is often difficult to identify a unique viewpoint owing to complex poses where the head is pointing in a different direction from the body; iii) for other classes like table, boat it can even be difficult to define the front, *etc*. Figure 6.6 shows the normalized mean images of the VOC2006 object classes for each of the five labeled viewpoints. Comparing these to Figure 6.1(e), we see that even if the viewpoint annotations were used to train separate detectors, the VOC2006 images are significantly less uniform than the INRIA Person ones and hence less likely to respond to a simple rigid template based detector.

In this and the following sections we introduce each VOC dataset that we used and analyze the results of our detectors on it. We tested on VOC 2006, 2007 and 2010 but we give our most detailed results on 2006. The 2010 dataset includes the 2008 and 2009

#	Methods	**Mean**	Bike	Bus	Car	Cat	Cow	Dog	Horse	Mbike	Person	Sheep
1	1Root_f-HOG	31.6	57.2	40.1	55.0	5.0	31.9	5.3	22.0	41.7	25.1	32.2
2	1Root_f-LBP	33.1	54.0	39.5	53.8	15.9	33.0	9.2	21.1	44.8	21.8	38.4
3	1Root_f-LTP	37.8	56.2	45.2	56.1	17.3	35.8	16.8	29.8	51.4	28.9	40.0
4	1Root_f-LBP+LTP	37.9	58.2	45.7	56.3	18.5	37.3	15.5	29.9	47.1	32.2	38.6
5	1Root_f-LBP+HOG	38.6	56.0	46.8	55.9	20.6	35.7	17.8	30.2	51.9	30.2	41.2
6	1Root_f-LTP+HOG	38.7	58.5	48.5	56.5	14.6	37.9	17.6	30.4	50.6	32.9	39.6
7	1Root_f	39.6	57.4	47.6	55.7	20.8	38.2	17.8	31.8	51.0	34.4	40.8
8	2Root_f	43.1	58.3	57.7	60.6	21.7	40.8	18.5	41.0	54.7	34.6	43.6
9	2Root	43.3	57.5	51.9	58.5	24.0	42.1	20.3	38.3	56.0	39.6	44.3
10	6Root	48.1	64.4	58.8	65.6	21.9	47.5	22.1	47.5	63.5	40.8	48.7
11	1Root_f-11%FWT	38.7	57.4	45.1	56.5	18.8	37.2	18.0	29.0	50.7	33.1	41.0
12	1Root_f-10%WT	38.6	58.1	47.7	56.1	13.8	37.8	19.4	30.4	49.2	32.6	41.3
13	1Root_f-10%IWT	39.3	59.0	47.5	56.4	17.7	38.5	18.4	30.3	50.9	33.6	41.1
14	1Root_f-15%IWT	39.6	58.7	48.4	57.1	18.7	37.9	19.1	30.0	50.8	34.5	41.1
15	6Root-10%IWT	47.9	65.3	60.1	67.0	20.5	46.5	20.7	45.5	64.2	40.5	49.0
16	6Root-10%WT	47.7	65.3	61.2	65.8	21.6	46.2	20.9	44.3	63.3	39.4	48.9
17	$1\text{Root}_f\times$6Part	45.9	64.5	53.9	63.3	27.4	43.3	21.6	40.6	55.7	42.9	46.0
18	6Root×8Part	**54.0**	**69.8**	**68.0**	69.5	**34.8**	**52.5**	**25.4**	**52.8**	**69.9**	47.3	**49.9**
19	6Root×8Part-18%WT	53.4	69.6	66.5	**71.0**	31.4	51.9	23.6	52.2	69.4	**48.9**	49.2
20	1Root_f-PLS	39.9	57.6	49.2	55.4	20.0	38.1	19.0	30.4	52.5	34.6	42.1
21	2Root_f-PLS	42.7	58.9	56.0	61.2	21.7	40.3	17.4	39.8	52.5	34.8	44.4
22	$1\text{Root}_f\times$6Part-PLS	45.1	63.6	51.6	61.7	28.4	41.0	21.4	40.3	58.4	41.2	43.3
23	$1\text{Root}_f\times$6Part[1]	34.3	59.2	40.7	54.5	8.1	34.6	7.0	28.3	48.5	32.2	30.3
24	$2\text{Root}_f\times$6Part[2]	42.5	62.0	49.3	63.5	19.0	41.7	15.3	38.6	57.9	38.0	40.2
25	6Root×8Part[3]	49.5	67.1	65.8	70.7	26.8	47.7	15.8	48.3	66.0	41.0	45.6

Table 6.1: *Average Precisions on VOC2006 for some of our latent detectors and those of Felzenszwalb et al. [2008, 2009, 2010b][1, 2, 3]. The "f" subscript denotes detectors with folded root filters instead of mirrored pair ones. The feature set used is LBP+LTP+HOG_{31} unless otherwise mentioned.*

ones as subsets (and 2007 too) so we did not test on these, while 2006 and 2007 were the last years in which the test set was made publicly available, so many authors have provided detailed results on these.

6.4 PASCAL VOC2006

The VOC2006 dataset consists of 2618 `trainval` images containing 4754 annotated instances and 2686 `test` images containing 4753 annotated instances of the 10 object classes bicycle, bus, car, cat, cow, dog, horse, motorbike, person and sheep. We detail the effects of different feature sets and combinations, single and multiple roots with and without parts, and sparsity on this dataset. Table 6.1 gives an overview of the results for

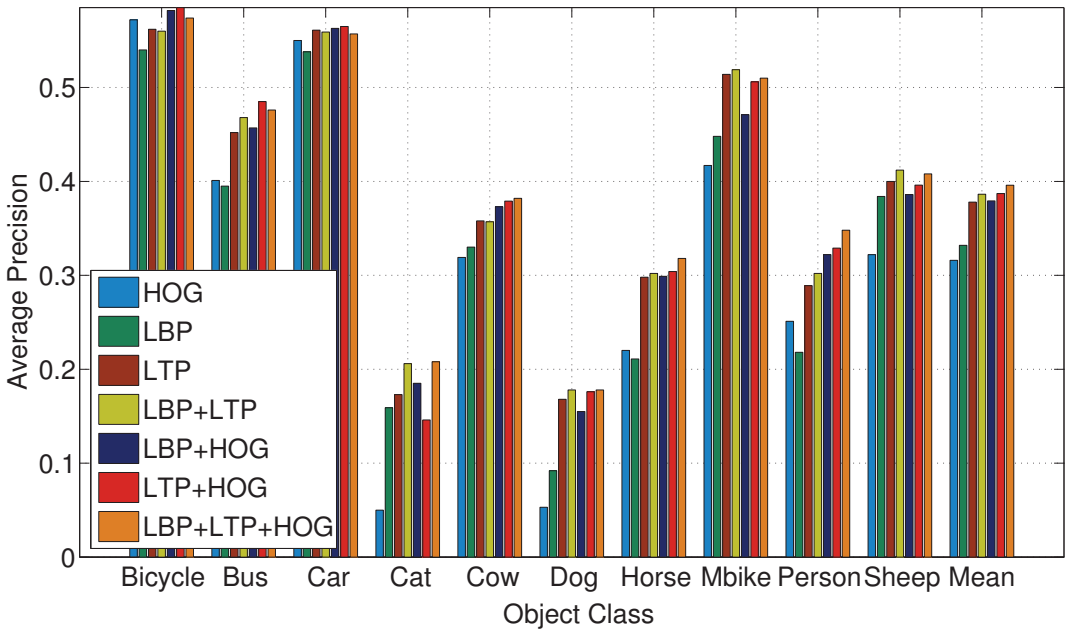


Figure 6.7: *Average Precisions for single root (1Root$_f$) latent detectors on VOC2006 using the given feature sets.*

several of our detectors and those of Felzenszwalb et al. [2008, 2009, 2010b].

6.4.1 Feature Sets

Figure 6.7 summarizes the Average Precisions on VOC2006 of single folded root (1Root$_f$) latent detectors based on various single and combined feature sets. Several observations can be made. Firstly on Mean AP, LBP outperforms HOG on VOC2006, beating HOG on 5 of the 10 classes, notably on animals such as cat, cow, dog, sheep where texture is a highly discriminative cue. Conversely, HOG outperforms LBP on the other 5 classes bicycle, bus, car, horse, person, which are characterized more by their geometry than by their texture.

Secondly, LTP features seem to inherit the good characteristics of both LBP and HOG, giving consistently better performance than either on both structural and textural object classes and improving the Mean AP by respectively 4.7% and 6.2%. Note that in these experiments the threshold of the LTP features was always set to $\tau = 5$ (for 256 level color images). This is the optimal value for both the VOC2006 car and person classes. The optimal values for the other classes may be somewhat different, but using a single value is simpler and in general we find that LTP results are not very sensitive to the exact value chosen.

To the best of our knowledge, the results for LTP are the best ever reported on VOC2006 for single root detectors based on a single feature channel. In fact, these

detectors outperform the HOG single root and parts approach of [Felzenszwalb et al. 2008] (row 23 of Table 6.1) on 8 of the 10 classes, improving the Mean AP from 34.3% to 37.8%. Both methods use the same training scheme, the only differences being the feature set and the fact that the LTP detectors are trained without including parts. The LTP detectors have a lower feature dimension than the HOG parts based ones, so the observed improvement seems to be due solely to LTP's ability to capture rich local texture and shape information.

Thirdly, as Figure 6.7 shows, combinations of feature sets typically outperform any of their individual components. Globally, LBP+LTP+HOG gives the best results with 39.6% Mean AP, followed by LTP+HOG with 38.7%. However compared to LTP alone, LBP+LTP+HOG has 78% more dimensions yet improves the Mean AP only by 1.8%. This is presumably because the single root detector architecture has only a limited capacity to model class variability and its results are already close to saturation for these features.

To sum up, LTP is the most successful individual feature set tested on VOC2006, but the combination LBP+LTP+HOG has even better performance – an 8% increase in Mean AP relative to HOG_{31} features and a 1.8% increase relative to LTP – in return for a significantly larger feature vector. Given that large feature vectors can limit a detector's speed and even – owing to overfitting and the fact that memory usage limits the number of examples that can be stored in memory during training – its accuracy, LTP features appear to be a promising compromise for practical detectors.

6.4.2 Single versus Multiple Roots and Parts

We ensure that all of our complete detectors are bilaterally (left-right) symmetric as we find that this gives better results from the limited training data, without the need to explicitly flip the positive and negative training sets. Our single root (1Root_f) detectors use the folding method of Sec. 3.4.1. For multi-root detectors, the components can either come in mirrored pairs or be individually symmetric using folding – *c.f.* Sec. 3.4. The former is more flexible in the sense that the individual components can be asymmetric, the later in that a larger set of different aspect ratios can be represented. Overall, compared to 1Root_f on VOC2006, both the 2Root (mirrored pair) and the 2Root_f (2 folded root) methods increase the Mean AP by about 3.5%. 2Root is slightly better than 2Root_f overall but although the differences are quite large for some classes they do not appear to be systematic within each broad group of classes (*c.f.* rows 8 and 9 of Table 6.1). Nonetheless, the results for 2Root on classes such as person, cow, cat suggest that these classes are better characterized by allowing bilateral asymmetry (left-facing versus right-facing components) than by enforcing symmetry and allowing two aspect ratios (*e.g.* sitting versus standing or front versus side).

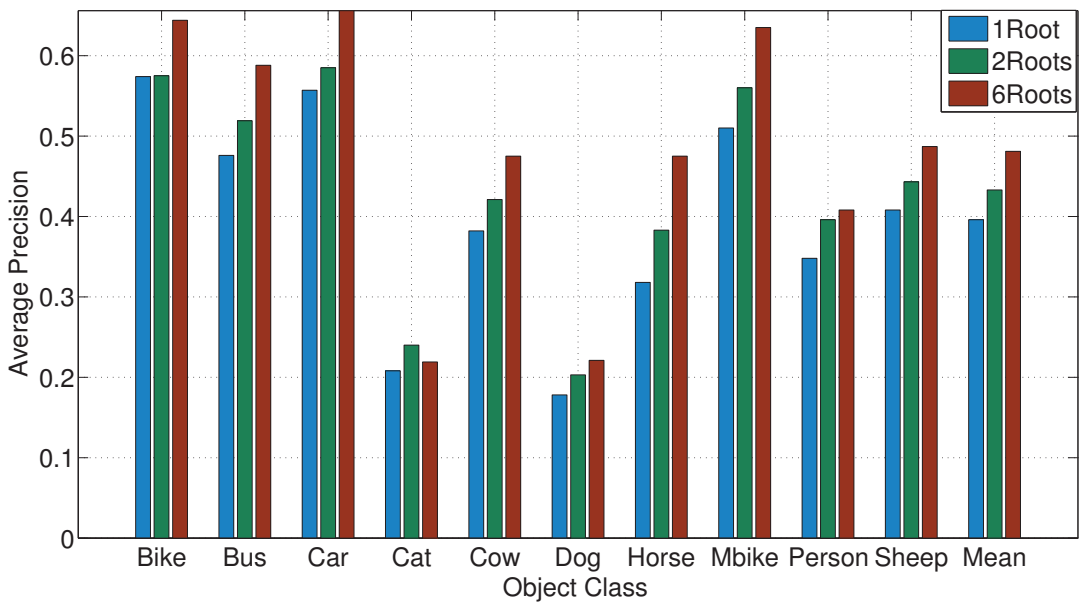


Figure 6.8: *Average Precisions for single root (1Root$_f$), two root (2Root) and six root (6Root) detectors on the 10 classes of the VOC2006 test set.*

Figure 6.8 compares the Average Precisions of single root folded and two and six root mirrored pair detectors on VOC2006. As expected, increasing the number of roots generally improves the results, although exceptions such as cat are reminders that current heuristics for initializing many roots are imperfect and that overfitting can easily occur. Overall, increasing the number of roots from one to six improves the mean AP by 8.5%. Owing to our stronger feature set, the performance of our root-only detectors is roughly on a par with the corresponding HOG$_{31}$ parts based approaches of [Felzenszwalb et al. 2009, 2010b]. For example, our 2Root$_f$ (row 8 of Table 6.1) detectors outperform the 2Root$_f$×6Part method of [Felzenszwalb et al. 2009] (row 24) on 5 of the 10 classes, increasing the Mean AP from 42.5% to 43.1%, while our six root (6Root) detectors (row 10) have performance close to the current best method, the Felzenszwalb et al. [2010b] 6Root×8Part approach (row 25). Admittedly our root only detectors have higher feature dimension than these part based models, but they are still much faster to train and run owing to the absence of the latent part search. For instance, a 6Root detector requires about 6.7 seconds to process an image[2].

Although our root-only detectors already have very good performance, adding parts further enhances their accuracy, particularly for classes with high degrees of pose variability. For example for the 6Root detectors, introducing eight parts (rows 10 and 18

[2]Note that these timings do not use the parallel version of the Felzenszwalb et al. [2010b] code in which multiple images/windows are scanned in parallel during training and testing, as we were not able to run this owing to Matlab license constraints.

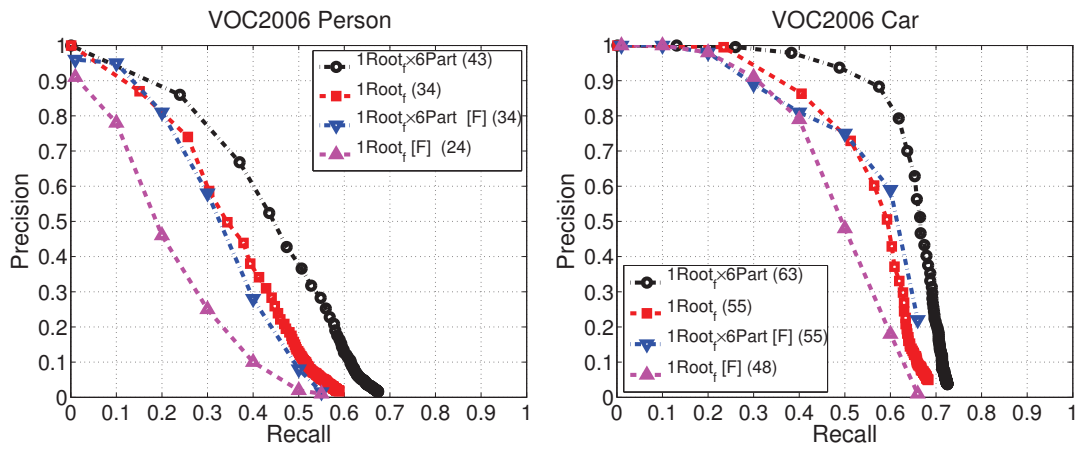


Figure 6.9: *Precision-Recall plots for several root and root plus parts detectors on the (left) person and (right) car classes of VOC2006. "F" denotes methods from [Felzenszwalb et al. 2009]. The figures in parentheses are the corresponding APs.*

of Table 6.1) improves the APs for the person and cat classes respectively by 7.2% and by 12.5%, while the Mean AP increases by 5.9% from 48.1% to 54.0%.

Relative to the current state-of-the-art 6Root×8Part method of [Felzenszwalb et al. 2010b] (row 25), our 6Root×8Part detectors increase the Mean AP by 5.5%, with better accuracy on 9 of the 10 classes. In fact our 1Root_f×6Part detectors (row 17) already improve on the earlier state-of-the-art 2Root_f×6Part ones of Felzenszwalb et al. [2009] (row 24) by 3.4%. However although these detectors have higher accuracies than other existing parts based ones, they are also quite slow to train and test owing to the high dimensionality of the feature set. Also note that parts are quite expensive – computationally, an additional pair of parts costs roughly as much as a newly added root. For instance, our 6Root×8Part detectors require about 23.5 seconds to process an image.

Figure 6.9 illustrates that both the improvements from using our extended feature set and the improvements from adding parts extend across the entire Precision-Recall range, with the extended features contributing slightly more for high precision/low recall settings and the parts for low precision/high recall ones.

6.4.3 Partial Least Squares Dimensionality Reduction

The above detectors used linear SVM trained on the raw input features. This gives good results but it makes training rather slow for high dimensional feature sets. To speed up training, we can use PLS-reduced feature sets. In both the single root (1Root_f) and two root (2Root_f) cases, detectors trained using 30D PLS-reduced features per component have slightly better accuracy than unreduced ones on 5 of the 10 classes (rows 20 – 21 versus 7 – 8 of Table 6.1), with a 0.3% increase in Mean AP for 1Root_f and a 0.4%

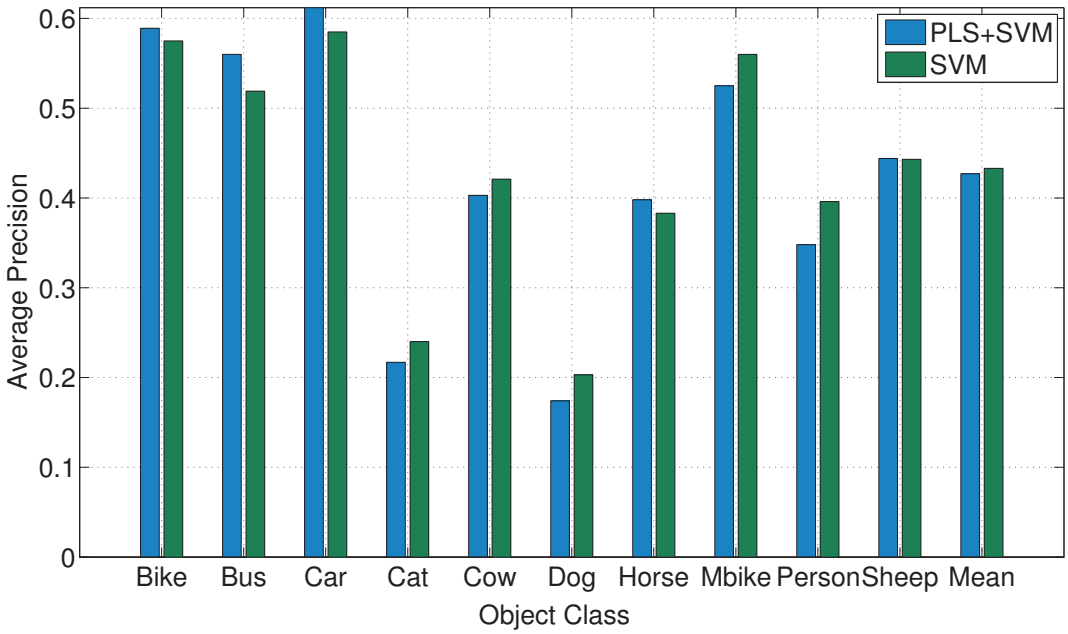


Figure 6.10: *Average Precisions of 2Root$_f$ detectors on the 10 classes of VOC2006, obtained by training a linear SVM on 30D PLS-reduced and unreduced features.*

decrease for 2Root$_f$. Increasing the PLS dimension to 60 instead of 30 reduces the Mean AP of 1Root$_f$ by 0.1%. The PLS based detectors are faster to train than unreduced ones. For example using PLS speeds up the overall training of 2Root$_f$ by about 45%, and decreases the time spend in classifier learning by a factor of 12. The above conclusions regarding precision continue to hold for parts based models (row 22).

6.4.4 Sparsification

We tested three variants of our SVM Weight Truncation feature selection method. FWT sparsifies a given dense final stage detector, IWT recomputes the sparsification during each cycle of latent learning, and WT selects the active features once and for all at Stage II of the latent training process. IWT is thus more expensive computationally than FWT, which in turn is more expensive than WT. Relative to the 39.6% Mean AP of dense 1Root$_f$ detectors on VOC2006, FWT at 50% nonzeros gives a slightly improved Mean AP of 39.9%, and even at high sparsity levels such as 11% and 6% nonzeros it still gives respectively 38.7% and 37.7% Mean AP. Similarly, Figure 6.11 compares 1Root$_f$ detectors trained with several levels of sparsity using IWT. Sparsification now gives ever better results. Even with an order of magnitude reduction in the feature dimension (10% nonzeros), the sparse detectors give better accuracy than dense ones on structural object classes such as bicycle, motorbike, *etc.*, with only minor losses on more textural

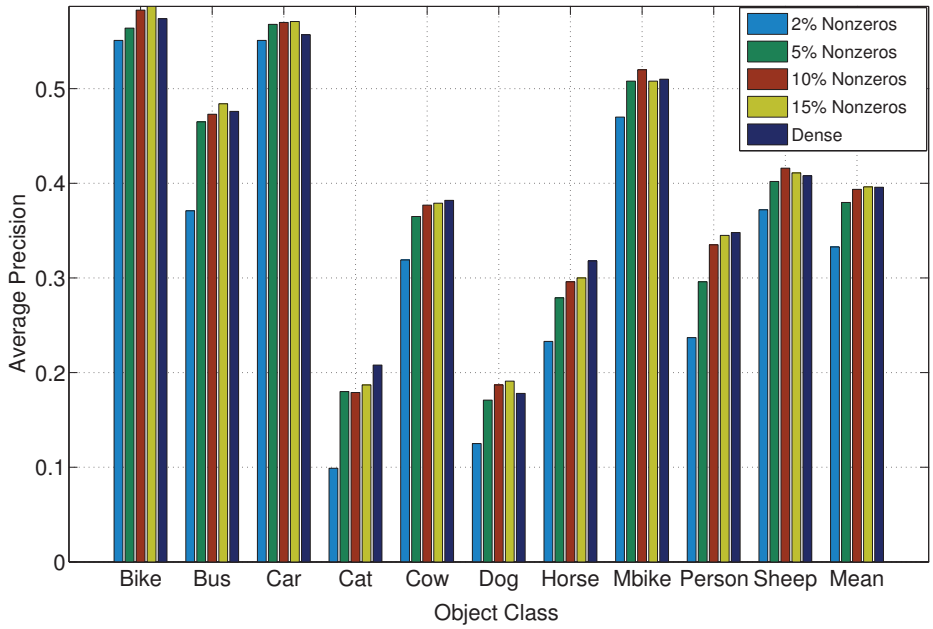


Figure 6.11: *Average Precisions on the VOC2006 test set for single root (1Root$_f$) detectors with various levels of sparsity. The sparse detectors are trained using Iterative Weight Truncation. Note that sparsification improves the accuracy in most cases, especially for classes where geometry (sparse edge layout) is more important than texture. The "nonzeros" figures (2%,* etc.*) are the percentage of the original feature set that is used in the final detector.*

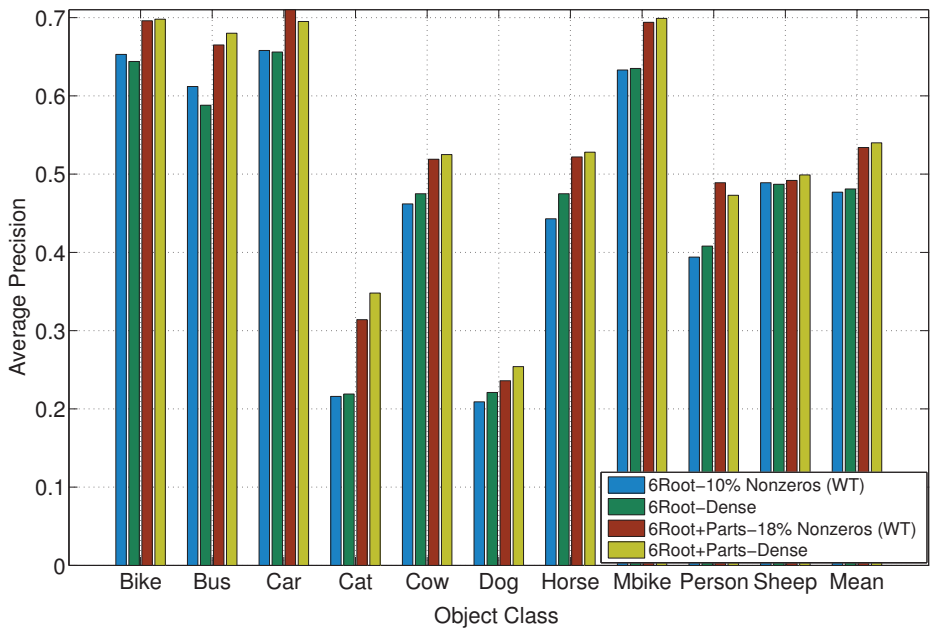


Figure 6.12: *Average Precisions for sparse and dense six root (6Root) and six root plus parts (6Root×8Part) detectors trained using Weight Truncation on the VOC2006 test set.*

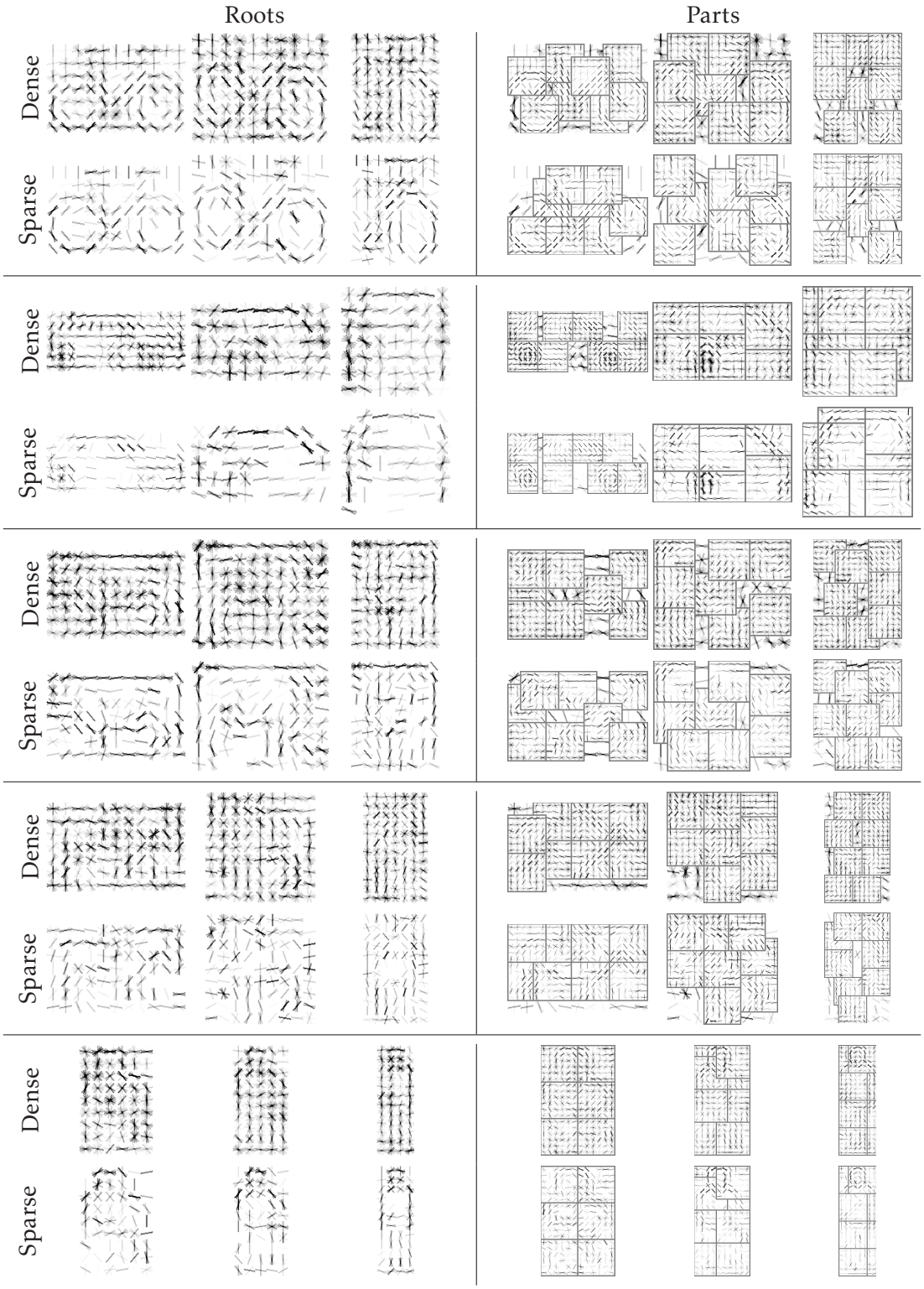


Figure 6.13: *Dense and sparse (10% nonzero WT) 6Root×8Part models learned for the VOC2006 bicycle, car, cow, horse and person classes. Only the positive-SVM weight entries of one filter from each mirrored pair is shown, and the roots and parts are shown separately. Note that the sparse filters, particularly the roots, are significantly sharper and less cluttered than the dense ones.*

classes such as cow and cat. This suggest that relatively small sets of edges and texture bins suffice to characterize these object classes and that enforcing sparsity filters out clutter and allows the detector to learn better weights for the surviving features. Overall, the accuracy of the sparse 1Root_f detectors on VOC2006 is as good as that achieved by dense detectors, with respectively 39.3%, 39.6% and 39.6% Mean AP for the 10% nonzero, 15% nonzero, and dense detectors.

Although IWT predictably gives the best results of our three sparsification strategies, detectors trained using weight truncation (WT) are only slightly less accurate and faster to train. For example, for 1Root_f at 10% nonzeros (rows 12 and 13 of Table 6.1), WT gives 38.6% Mean AP compared to 39.3% for IWT while being 12-15% faster. Moreover, the accuracy differences decrease when multiple roots and parts are included. For instance, 6Root detectors trained with 10% nonzero WT, 10% nonzero IWT, and dense features have Mean APs of respectively 47.7%, 47.9% and 48.1%, while for 6Root×8Part detectors with 18% nonzeros (10% nonzero roots and 20% nonzero parts) WT gives 53.4% Mean AP compared to 54% for dense features (*c.f.* Figure 6.12), yet training WT sparse detectors is about 1.5–2.5 times faster than training dense ones. Note that the Mean AP of these 18% nonzero detectors – which have ∼37 features per cell, as compared to 31 for dense HOG_{31} – is 4.1% better than that for the dense HOG_{31} of Felzenszwalb et al. [2010b]. Figure 6.13 shows some of our dense and sparse 6Root×8Part models. Note that introducing sparsity filters out a good deal of the background clutter and the redundant foreground orientations, leading to cleaner and somewhat sharper final detectors. This effect seems to be more pronounced for the root filters than the part ones. To give an idea of the overall performance attained, Figure 6.14 shows some examples of detections for sparse models on images from the VOC2006 test set.

An examination of the sparse features selected shows that all three feature channels LBP, LTP and HOG make significant contributions to the sparse detectors. Forcing the sparsity to be distributed uniformly among them leads to little loss of accuracy, *e.g.* reducing the Mean AP of 1Root_f IWT from 39.3% to 39.2%. Note that at 15% nonzeros, LBP+LTP+HOG_{31} cells have on average the same number of nonzero elements as HOG_{31} ones, so subsequent processing costs are similar despite significantly better accuracy. In fact, even with only 2% nonzeros (∼4 features per cell on average), the combined feature set still gives better accuracy that either HOG or LBP alone with respectively 31 and 59 features per cell.

Overall, sparsification leads to significant speedups during both training and testing, typically with little loss of accuracy and in some times even a small gain. It can be used with any feature set, single or multiple roots, parts, *etc*. Regarding the choice of sparsification method, FWT has the advantage that it can be used to sparsify already trained detectors, and IWT that it achieves the best available accuracy for a given level of sparsity, but overall we prefer WT owing to its very fast training times and

Figure 6.14: *A few examples of the output of our sparse (10% nonzero WT) 6Root×8Part detectors on the VOC2006 test set. Each image is scanned by all of the participating detectors,* i.e. *bicycle, bus, car, cow, horse, motorbike and person.*

minimal loss in accuracy relative to IWT. As an avenue for future work, the speed of very sparse detectors may make them useful as initial stages of multistage detectors and soft cascades [Bourdev and Brandt 2005]. As Figure 6.15 shows, even at very high sparsity levels, the recall range of these detectors remains identical to that of dense detectors, so it should be possible to find settings that provide useful pruning without losing too many true positives.

6.5 PASCAL VOC2007

The VOC2007 dataset has the same training and testing protocols as VOC2006 but twice as many classes and almost twice as many images. The `trainval` set has 5011 images with 12608 annotated instances while the `test` set has 4952 images with 12032 annotated instances of the 20 classes aeroplane, bicycle, bird, boat, bottle, bus, car, cat,

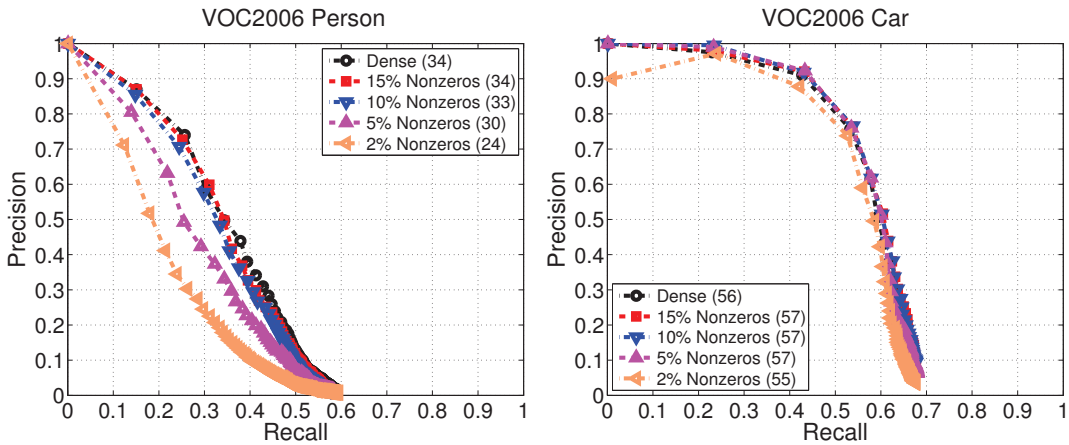


Figure 6.15: *Precision-Recall plots for sparse WT detectors on the VOC2006 (left) person, and (right) car classes. The figures in parentheses are the corresponding APs.*

chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train and tv monitor.

We use exactly the same parameter settings as for VOC2006. No separate tuning was done on the VOC2007 training set. Our findings for the various detector configurations (types of features, numbers of roots, addition of parts, dimensionality reduction, *etc.*) are all consistent with those for VOC2006. Table 6.2 summarizes the results for a few of our detectors and for the best-performing methods from [Vedaldi et al. 2009] and [Felzenszwalb et al. 2008, 2009, 2010b]. Our single root 1Root_f detectors (row 1) already have better accuracy than the $1\text{Root}_f\times 6\text{Part HOG}_{31}$ detectors of Felzenszwalb et al. [2008] (row 6) on 11 of the 20 classes, and adding an additional root (row 2) or parts (row 4) further improves the results in most cases. Our $1\text{Root}_f\times 6\text{Part}$ detectors (row 4) outperform the $2\text{Root}_f\times 6\text{Part}$ ones (row 7) of Felzenszwalb et al. [2009] on 12 of the 20 classes, increasing the Mean AP by 1.5% to 28.3%.

With 32.0% Mean AP, the performance of our six root detectors (row 3) is better than that of the combined localization and classification approach of Harzallah et al. [2009] who report 28.9%, and on par with the multiple kernel learning approach of Vedaldi et al. [2009] and the earlier state-of-the-art 6Root×8Part method of Felzenszwalb et al. [2010b], who report respectively 32.1% and 32.3% Mean AP. Our six root detectors outperform Vedaldi et al. [2009] on 11 of the 20 object classes and they are also much faster to train and evaluate – a 6Root detector requires ∼7 seconds to process an image, as compared to ∼1.2 minutes for the Vedaldi et al. [2009] cascaded multiple kernel learning approach. Compared to the Felzenszwalb et al. [2010b] 6Root×8Part approach, our 6Root method gives better results on 8 of the 20 classes. Adding parts give state-of-the-art results. Our 6Root×8Part method outperforms the [Felzenszwalb et al. 2009] 6Root×8Part HOG_{31} approach on 16 of the 20 classes, and that of [Vedaldi et al. 2009] on

Methods	**Mean**	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	TV
1Root_f	22.8	25.7	39.8	9.2	6.3	23.6	33.2	37.3	12.6	10.6	17.3	25.7	9.6	35.5	35.4	14.2	13.1	16.6	24.6	27.0	38.0
2Root_f	26.1	27.5	47.5	0.04	13.8	26.0	37.3	42.4	14.4	16.1	22.9	17.3	13.6	43.7	38.5	32.8	11.7	21.2	23.4	36.9	35.9
6Root	32.0	32.1	54.0	13.2	17.7	24.3	48.7	50.7	17.6	21.4	26.9	25.5	15.8	56.1	45.6	37.6	15.3	22.3	32.0	43.0	40.2
$1\text{Root}_f\times$6Part	28.3	31.0	45.8	10.3	7.6	29.7	38.6	48.6	22.9	17.3	23.2	29.5	13.1	41.2	37.7	34.1	16.2	22.5	26.7	27.1	43.3
6Root×8Part	**36.0**	**39.2**	57.9	**17.1**	**21.6**	**29.7**	**52.2**	56.7	27.9	**23.0**	27.5	**34.1**	17.6	**60.1**	**51.4**	39.5	**14.9**	**24.3**	**35.6**	44.8	44.8
$1\text{Root}_f\times$6Part[1]	21.3	18.0	41.1	9.2	9.8	24.9	34.9	39.6	11.0	15.5	16.5	11.0	6.2	30.1	33.7	26.7	14.0	14.1	15.6	20.6	33.6
$2\text{Root}_f\times$6Part[2]	26.8	28.7	55.1	0.6	14.5	26.5	39.7	50.2	16.3	16.5	16.6	24.5	5.0	45.2	38.3	36.2	9.0	17.4	22.8	34.1	38.4
6Root×8Part[3]	32.3	28.9	**59.5**	10.0	15.2	25.5	49.6	**57.9**	19.3	22.4	25.2	23.3	11.1	56.8	48.7	**41.9**	12.2	17.8	33.6	45.1	41.6
Vedaldi et al.	32.1	37.6	47.8	15.3	15.3	21.9	50.7	50.6	**30.0**	17.3	**33.0**	22.5	**21.5**	51.2	45.5	23.3	12.4	23.9	28.5	**45.3**	**48.5**
Zhang et al.	34.3	36.7	59.8	11.8	17.5	26.3	49.8	58.2	24.0	22.9	27.0	24.3	15.2	58.2	49.2	44.6	13.5	21.4	34.9	47.5	42.3

Table 6.2: *Average Precisions for some of our LBP+LTP+HOG$_{31}$ detectors, Felzenszwalb et al. [2008, 2009, 2010b][1, 2, 3], Vedaldi et al. [2009] and Zhang et al. [2011] on VOC2007.*

15 of the 20 classes. It also outperforms the recent state-of-the-art 6Root×8Part method of Zhang et al. [2011] on 16 of the 20 classes, improving the Mean AP by 1.7%. The Zhang et al. method is comparable to ours in the sense that it uses multiple features (local structured HOG and LBP) without additional context, however it uses boosting to select the discriminant features.

6.6 PASCAL VOC2010

The VOC2010 dataset has the same 20 classes as VOC2007, but now has 10103 images with 23374 annotated instances in the `trainval` set and 9637 images with 22992 annotated instances in the `test` set. Table 6.3 summarizes the results for some of our detectors, for Wang et al. [2010] and for two state-of-the-art performers[3] [Yu et al. 2010; Felzenszwalb et al. 2010b]. It is not easy to find methods that are comparable with ours on this dataset because relatively few methods have been tested on it, details are available for only a few of these, and most of the good performers are complex methods that include global context, segmentation and similar cues that are beyond our scope here. Wang et al. [2010] is included because it is the best performer in the published VOC2010 rankings that is known to resemble our method, using linear SVM over multiple feature channels including HOG, LBP and dense HOG based bag of words features. Our 6Root detectors give better performance than Wang et al. on 13 of the 20 classes, with a 2.1% increase in Mean AP.

The other two methods are not directly comparable to ours in the sense that they use complementary cues – either features extracted from segmentation maps or classification with global context modeling – in addition to deformable part models. Felzenszwalb

[3] `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/workshop/`

Methods	**Mean**	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	TV
2Root$_f$PLS	17.5	32.7	29.7	0.80	1.1	19.8	39.4	27.5	8.6	4.5	8.1	6.3	11	22.9	34.1	24.6	3.1	24.0	2.00	23.5	27.0
6Root	25.6	40.3	35.8	10.5	11.3	23.9	46.9	36.5	11.8	15.9	22.0	14.1	9.0	38.6	38.8	37.6	6.6	31.5	10.2	40.3	31.2
Wang et al.[1]	23.5	40.4	34.7	2.7	8.4	26.0	43.1	33.8	17.2	11.2	14.3	14.4	14.9	31.8	37.3	30.0	6.4	25.2	11.6	30.0	35.7
6Root×8Part[2]	33.8	52.4	54.3	13	15.6	35.1	54.2	49.1	31.8	15.5	26.2	13.5	21.5	45.4	51.6	47.5	9.1	35.1	19.4	46.6	38.0
6Root×8Part[3]	36.8	53.3	55.3	19.2	21	30	54.4	46.7	41.2	20	31.5	20.7	30.3	48.6	55.3	46.5	10.2	34.4	26.5	50.3	40.3

Table 6.3: *Average Precisions on VOC2010 for some of our detectors, the LBP+HOG+Bag of Words method of Wang et al. [2010][1], the segmentation combined method of Felzenszwalb et al. [2010b][2], and the boosted LBP+HOG+Multi-context detectors of Yu et al. [2010][3] on VOC2010.*

et al. [2010b] use their 6Root×8Part detectors to generate segmentation masks, learn separate segmentation models from these masks, then use both the detection and the segmentation modules for their final detector. Yu et al. [2010] combine HOG and boosted LBP based multi-root multi-part detectors [Felzenszwalb et al. 2010b] with a multi-context model learned using Radial Basis Kernel SVM over features combining local, global and inter-class contextual cues. They report a 3% increase in Mean AP over the HOG based multi-root multi-part detectors of [Felzenszwalb et al. 2010b] on VOC2010 and a 4% increase on VOC2007. However, the improvement comes at the cost of both additional complexity and increased training and testing time because to include the global contextual cues, separate classification models need to be learned for each class using linear SVM and vector quantized SIFT features. Similarly, to get the inter-class contextual features, separate root-only HOG and LBP detectors are learned for each class and all of these need to be evaluated around each detected bounding box.

6.7 ETHZ Dataset

The ETHZ dataset [Ess et al. 2008] consists of three test sequences used to evaluate pedestrian detectors. They were recorded using a mobile stereo platform at different locations in a busy city center under various conditions: Sequence 1 (999 images with 5193 annotations) contains images captured on a cloudy day; Sequence 2 (450 images with 2359 annotations) contains images of moving people recorded in a busy square; and Sequence 3 (354 images with 1828 annotations) contains images captured on a sunny day.

To facilitate comparison with other methods tested on this dataset, we plot Recall vs. False Positives Per Image (FPPI) curves – *c.f.* Sec. 3.6.3. Figure 6.16 presents results for our single root detectors trained on the INRIA and VOC2006 person sets and tested on the ETHZ sequences. For more direct comparison with Schwartz et al. [2009], the INRIA detectors were trained using the non-latent approach. Although Ess et al. [2007, 2008]

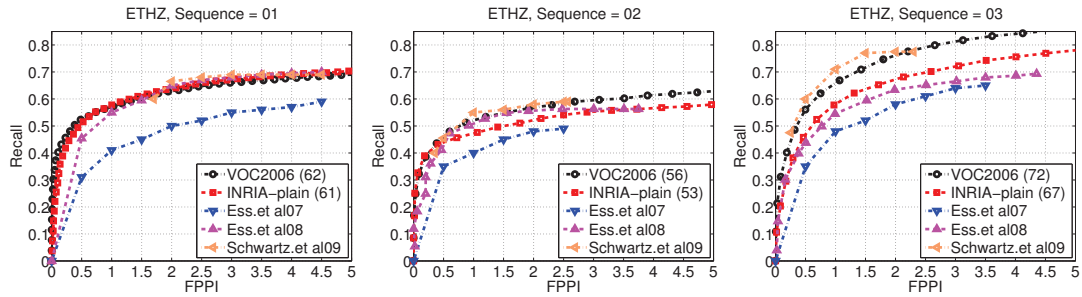


Figure 6.16: *Recall-FPPI plots on the three ETHZ test sets for our linear 1Root$_f$ detectors trained on the VOC2006 and INRIA person sets, versus some competing methods. The figures in parentheses are APs on the given test set, where known.*

use depth as an additional cue in their detectors, our 1Root$_f$ detectors still give better performance whether they are trained on INRIA or VOC2006. Likewise, despite being linear and having a 40× lower feature dimension, they have a higher recall at low FPPI than the Quadratic Discriminant Analysis based method of Schwartz et al. [2009] on Sequence 1, and near-identical performance on the other two sequences. Their linearity allows them to process an image every 3 seconds, whereas Schwartz et al. [2009] take 120 seconds (or 60 for their two stage method). These results show both the accuracy of our detectors and their ability to generalize across datasets.

6.8 Summary

This chapter has provided experimental results to support the choices of features, training mechanisms and classification methods introduced in Chapters 4 and 5. Among the individual feature sets tested, we found that the Local Ternary Patterns (LTP) introduced for face recognition by [Tan and Triggs 2010] were also very promising for object recognition, outperforming the well-established HOG on many classes. As many authors have found [Harzallah et al. 2009; Vedaldi et al. 2009; Wang et al. 2009; Schwartz et al. 2009; Ott and Everingham 2009], well-chosen combinations of feature sets often offer improved performance. Here, our LBP+LTP+HOG$_{31}$ feature set gives better accuracy than the existing combinations that we are aware of despite being simpler, faster to calculate, and having relatively modest dimensionality compared to many of the other combinations that have been proposed.

The good performance of the Latent SVM training and multiple root and part approach of [Felzenszwalb et al. 2010b] was also confirmed. Adding roots generally improves the accuracy, as does adding parts, with the resulting detectors having state-of-the-art accuracies for context-free detectors on the INRIA Person, VOC2006, VOC2007, and ETHZ datasets, but not competing with the best context-incorporating ones on

VOC2010. However, our combined feature set is strong enough to allow our root-only detectors to outperform the corresponding HOG based root plus parts ones of Felzenszwalb et al. [2009] in many cases. This is useful because root-only detectors are much faster than ones including parts.

Finally, reducing the feature set dimensionality using Partial Least Squares was shown to reduce training times with little or no loss of accuracy relative to the corresponding unreduced detectors. Alternatively and perhaps more significantly, we found that enforcing sparsity (feature selection) using a simple SVM weight truncation heuristic both acts as a regularizer and speeds up training and testing. Even very sparse feature subsets with only a few features per cell give surprisingly good results, sometimes even improving the accuracy relative to dense features. The combination of rich feature sets such as LBP+LTP+HOG with sparsification appears to be very promising, allowing the most pertinent features to be selected while keeping the overall dimension as low as, or even lower than, that of individual feature sets such as HOG.

Chapter 7: Conclusions and Perspectives

This thesis has made several contributions to algorithms for the visual object detection. It has presented a family of sliding window Latent SVM object detectors based on a rich visual feature set that combines Histogram of Oriented Gradient, Local Binary Pattern and Local Ternary Pattern features to give state-of-the-art performance on several important datasets including PASCAL VOC2006 and VOC2007, INRIA Person and ETHZ. In order to tackle the detector speed and memory usage problems associated with modern high-dimensional feature sets, it has proposed two effective dimensionality reduction techniques. The first, feature projection using Partial Least Squares allows all kinds of detectors to be trained more rapidly with negligible loss of accuracy and no loss of run time speed for linear detectors. The second, feature selection using SVM weight truncation allows active feature sets to be reduced in size by almost an order of magnitude with little or no loss, and often a small gain, in detector accuracy. We also introduced Local Quantized Patterns, a generalization of local pattern features that provides local pattern style image neighbourhood codings that have the speed of local patterns and some of the flexibility and power of traditional visual word representations. We complemented these contributions with a detailed experimental study of the influence of the various configuration parameters and components of our approaches, and a comparison of them with state of the art methods on a selection of challenging publicly available datasets.

The following sections summarize these contributions and discuss some perspectives for future work.

7.1 Key Contributions

Local Ternary Patterns and Feature Combinations. Local Ternary Patterns (LTP) were introduced for face recognition [Tan and Triggs 2010]. We adapted them to object detection and showed that they give better performance on average than either HOG features or Local Binary Patterns (LBP) alone. HOG alone gives good results for object classes that are characterized mainly by shape cues, while LBP gives good results for classes that are characterized mainly by their local image textures. LTP inherits the good

characteristics of both HOG and LBP, capturing rich local texture and shape information to give equally good performance for both kinds of classes. To the best of our knowledge, LTP gives the state-of-the-art result for a single feature set on all of the datasets tested. The results also indicate that LTP is highly resistant to variations in lighting conditions and to changes in its threshold parameter τ. Moreover, we showed that HOG, LBP and LTP complement one another, so that an extended feature set that incorporates all three of them gives further improvements in performance leading to state of the art results for linear classifiers that use individual or compound feature sets based only on local appearance cues.

Partial Least Squares. We demonstrated that Partial Least Squares (PLS) is a useful discriminative dimensionality reduction method for all of the linear detectors that we tested including ones with multiple roots and/or parts. Using it can speed up classifier training by an order of magnitude and overall detector training by 25%-50% with little or no loss of accuracy. There is no loss of run time speed either because the (linear) classifier learned can be pulled back through the PLS projection to work on the original feature space, so that no feature projection is needed at run time. We also found that simple (*e.g.* quadratic) nonlinear classifiers based on low dimensional PLS-reduced feature sets give slightly better accuracy than linear ones without too much loss of speed, remaining much faster than conventional kernel classifiers.

Sparse Classifiers. As an alternative to feature projection, we introduced feature selection/sparse classifiers based on a simple SVM weight truncation heuristic. This allows classifiers that have an order of magnitude fewer active features to be obtained efficiently, with negligible loss in accuracy and often even a small gain. Both training and testing are faster than for the corresponding dense classifiers. Despite its simplicity, this method is far more effective than the other feature selection methods that we tested including L_1 approaches and boosting.

Local Quantized Patterns. Although both LBP and LTP perform well in object detection problems, feature set size considerations prevent them from coding more than an extremely local neighbourhood around each pixel (typically a circle of 8 immediate neighbours of the pixel). Conversely, although indirect representations such as Bag of Words codebooks can handle significantly larger neighbourhoods, they are too slow to use for pixel-level local pattern coding. We combined the two ideas to produce Local Quantized Patterns (LQP). These are local pattern features over larger neighbourhoods than LBP/LTP, that use lookup table based vector quantization to efficiently reduce a large direct neighbourhood code to a much smaller codebook based output code. In practice LQP can code neighbourhoods of up to about 25 pixels with no increase in

feature vector size and only a negligible increase in run time over LBP/LTP. Codebook learning is very efficient and neighbourhood shapes can easily be adopted to give the best results for the given problem. Our experiments show that the resulting LQP features give state-of-the-art performance, outperforming both LBP/LTP alone, and in many cases even the LBP+LTP+HOG combination. Work on this is still in progress.

7.2 Future Work

We finish with a few suggestions for extending our work, and for future work in object detection in general.

Action Recognition/Localization. Humans and their actions are one of the most important components of audiovisual content. Methods that automatically analyze such content and provide high-level descriptions, highlights or rapid search capabilities are still in their infancy. Many current approaches use Bag of Word strategies to recognize and localize actions, but as in object detection, a multi-faceted approach may be useful to handle actions at a variety of different image and temporal scales, with extreme changes in viewpoints and frequent partial occlusions. In particular, humans are central to most action recognition problems so sliding window action detectors based on spatio-temporal visual features should be able to complement other existing approaches. Recent results such as [Kläser et al. 2010] support this intuition. In this regard, our features could be extended to the spatio-temporal domain and used in sparse root and parts detectors to localize actions.

Local Quantized Patterns. Local Quantized Patterns (LQP) are a generalization of local pattern features that give very good results for the object detection. They need to be tested in other visual recognition tasks such as face and texture recognition, image classification, semantic segmentation, *etc*. Also, only a small subset of the many possible variants of the LQP idea have been tested. For example, it would be interesting to study LQP codes over local image representations such as local Haar pyramids, Discrete Cosine Transforms, *etc*., and also with more sophisticated pattern comparison techniques such as Earth Movers distance metrics and kernel representations.

L_1 **Support Vector Machines.** Although our sparse detectors have shown considerable promise, our current sparsification algorithm is rather heuristic. Ideally we would like to have a method that provides stronger theoretical guarantees. SVM's that combine the traditional hinge loss with L_1 weight regularization seem to be the most promising avenue for this, but current algorithms for learning these are not very satisfactory and

more reliable large scale algorithms (that simultaneously handle both many examples and many features) need to be developed.

Nonlinear Extensions. Kernel classifiers are known to be very powerful for object detection, but they are usually too slow to be practical. The combination of sparse feature sets selected using our linear method and fast additive kernel classifiers [Vedaldi and Zisserman 2010] seems to be a promising way forward.

Contextual Modeling. Although our proposed detectors have good performance, object detection is still far from a solved problem and new ideas are needed. One promising avenue is to include more contextual information into the detectors. In particular, the inclusion of spatial co-occurrence cues based on nearby objects is likely to reduce the number of false positives. For instance, Desai et al. [2009] incorporate spatial co-occurrence statistics of different objects classes into the non-maximum suppression process to support more probable arrangements/occurrences and discourage less probable ones. Similarly, incorporating segmentation and image classification information [Ramanan 2007; Harzallah et al. 2009; Yu et al. 2010] and explicit object scale models [Park et al. 2010] should also improve the performance. As object detection is just one small piece of the overall jigsaw of scene understanding, modeling the physical, functional and causal relationships among the detected objects is not only likely to improve the performance of the detectors, but also to aid in the interpretation of scene content [Gupta et al. 2011].

Hierarchical Modeling. Our current detectors use shallow structure, however like many other researchers [Serre et al. 2007; Ranzato et al. 2007] we feel that "deep hierarchical learning" architectures hold great promise for solving difficult problems like visual recognition. The proposed top-down detectors could be deepened by including parts of parts or training mixtures of latent tree models that represent the object parts as tree nodes [Zhu et al. 2010].

Large Datasets. Relative to the number of object categories that a human can recognize and the vast amount of labeled, semi-labeled and unlabeled data that is available on the internet, current detectors are trained only for small numbers of categories on small datasets. Both lack of algorithmic maturity and scarce computational resources are to blame for this. The dominant mode of computation is rapidly migrating from desktop workstations to clusters of GPUs, cloud computing, *etc.*, but current representations and algorithms need to be adapted to process the huge amounts of data that are available on these resources in the most effective manner. Possible solutions include online domain adaption [Jain and Learned-Miller 2011], where detectors trained offline are adapted

to new test data distributions with the help of supplementary training data, active learning techniques where, *e.g.*, crowd sourcing [Vijayanarasimhan and Grauman 2011] is incorporated into the detector training loop to allow reliable training from huge datasets, and combined training where detectors for a large set of categories are trained jointly by sharing common features among them [Torralba et al. 2007].

3D. Humans use models learned from the 3D world to recognize object categories and interpret scene content in 2D image space. Conversely, computers are usually trained and evaluated using only 2D representations. This loss of the third dimension makes the problem harder. As we are moving to an era where 3D-equipped consumer cameras, televisions and game consoles will be part of our daily lives, the collection of large well-segmented 3D datasets will become much easier. Such data could be used to bootstrap detector training, even if the final detectors are used only for 2D content.

Appendix A: Local Quantized Pattern Feature Sets

This appendix details some unpublished work in progress. We have seen that despite their extreme locality and very coarse quantization, local pattern features give good performance in many object detection tasks. However it is worth asking whether other variants of this idea – perhaps incorporating more pixels or additional levels of quantization or using different supports such as filled rectangles or strips of pixels – would be even more effective. The problem with larger supports and/or deeper quantization is the number of code values generated and hence the size of the cell-level local feature histograms. For practical purposes we would like to limit the latter to at most a few hundred entries per cell. Naively quantizing a group of n pixel gray-level comparisons into k levels gives a code of size k^n, limiting such codes to around 8 binary comparisons or 5 ternary ones per group. For LBP, uniform patterns were introduced to address this problem, but it is unclear how best to generalize them to non-circular group topologies and even for circular ones, the quadratic increase in the number of uniform patterns with circumference limits uniform coding to groups of at most about 20 pixels (and even then, the quality of the coding is likely to suffer as the fraction of the k^n codes that are uniform becomes vanishingly small). For ternary and higher order codes this problem is even more pronounced as the number of codes increases very rapidly with n and it is not at all obvious what the best analogue of uniformity is. In particular, it is not obvious that splitting LTP codes into two binary halves as in Chapter 4 allows LTP to reach its full potential. Nor is it obvious that assigning all of the nonuniform codes to a single histogram bin is the best solution – it might be better to assign each nonuniform code to the "nearest" (according to some metric) uniform one. In general it seems worthwhile to seek methods for learning an efficient and discriminative reduction from a large set of local codes (say k^n) to a much smaller set that allows a compact representation.

On the other hand "visual word" style approaches, where (potentially) continuous-valued feature vectors are quantized to a limited number of code values by vector quantization methods such as K-Means, have become very popular and successful

in recent years [Leung and Malik 1999; Varma and Zisserman 2002; Schmid 1999; Csurka et al. 2004]. These visual words approaches were imported from the text processing community and were initially used mainly for texture classification [Leung and Malik 1999; Schmid 1999; Varma and Zisserman 2002]. However in recent years they have been used to solve many vision problems ranging from image classification [Csurka et al. 2004] to action recognition [Kläser 2010]. Such visual word approaches certainly be applied to local pattern representations, where they would resolve many of the issues mentioned above. The problem is that most such approaches (*e.g.* K-Means) require a candidate to be compared with every dictionary element to discover its quantization class, or at very least they require an expensive data structure traversal for each candidate. This is (just) acceptable for patch-level descriptor quantization in visual word representations, but it is much too slow for a practical detector that needs to quantize a local pattern based at every single image pixel and pyramid level. For an approach of this sort to be useful for object detection, we need to limit the run-time code evaluation step to a constant-time operation per pixel – *e.g.* a closed form formula or a table lookup. We have developed a family of methods based on table lookup that combine the advantages of local patterns and vector quantization in an efficient and practical form. For simplicity we will refer to them as Local Quantized Patterns (LQP). The basic is sketched in the next section, while the subsequent sections give further details and experimental results.

A.1 Local Quantized Patterns

Although vector quantization against a large dictionary is ordinarily quite slow, if the set of possible input values is finite and small enough, the quantization code for each possible input can be precomputed and stored in a lookup table, allowing codes to be assigned at the cost of a single table lookup per example. We will use this to learn efficient reductions from initial, very high dimensional discretized local pattern representations to much lower dimensional vector quantization codings that can be used to build visual feature histograms. The lookup table representation also allows dedicated versions of algorithms such as K-Means to be used, making codebook learning very efficient. As an example of what is feasible, a ternary coding of a local pattern with $n = 16$ pixel comparisons gives a $3^{16} = 43$ million bin lookup table, while a binary coding of a pattern $n = 24$ pixel comparisons gives a $2^{24} = 17$ million bin table. In contrast, a typical vector quantization dictionary might contain 100 output codes.

During codebook learning the (codebook) training dataset is scanned to record the number of occurrences of each input code that occurs, storing these in a hash table or index, then these values and counts are passed to an algorithm (*e.g.* a count-weighted version of K-Means) that learns the actual codebook. In practice this gives very fast

training times because most of the possible input codes do not occur and all of the counts of those that do are processed in a single step. *E.g.* for 24 bit binary vectors over the INRIA Person positive training set, only 671,000 of the 17,000,000 possible input values actually occur and as a result 10 rounds of K-Means clustering for a 100 element dictionary takes only 12 minutes.

The above approach has the advantages that it allows local patterns to have many more pixels and/or quantization levels and a much wider range of geometries than standard hard-coded LBP/LTP, with only a negligible increase in overall run time. It also allows the size of the output code to be customized for the application, and adapts the coding to the dataset for better results. We will demonstrate some of these benefits on object detection problems below, but even in this case many possibilities remain to be tested and we make no attempt to test LQP in other applications of local patterns even though we believe that it would prove equally useful there. Moreover, for now we have only tested vector quantization methods based on simple L_2 patch comparison. It would also be interesting to test more sophisticated metrics such as Earth Mover style distances, given that the LQP architecture allows them to be incorporated at no additional run-time cost.

The LQP approach does have some important limitations. In particular, it requires a large lookup table at run time and even then it only allows the code length to be increased by a factor of about 3 (*e.g.* from 8 to 24 bits for binary codes, or from 5 to 16 for ternary ones) before the lookup table size starts to become prohibitive, while more finely quantized codes (say 5 or more quantization levels for $n \geq 8$) remain beyond reach.

A.1.1 Implementation Details

LQP codes can incorporate many more pixel comparisons, and hence a much wider variety of local pixel-level supports, than LBP/LTP. We tested a number of different geometric layouts including lines, crosses, circles and rectangles. Different numbers of quantization levels can also be tested, including binary, ternary or even quinary, and codes can be generated by comparing pixels with either the central pixel of the layout, a symmetric partner or the next pixel along a chain, or even by coding thresholded local Haar wavelets. Moreover, the vector quantization stage can use various discriminative or unsupervised codebook learning methods over a number of different inter-example distance metrics. Below we discuss each of these choices in turn. However note that for simplicity, at the cell level we will always retain the now-familiar structure for our LQP histograms, using 8×8 pixel cells, bilinear spatial interpolation of votes and L_1-Sqrt normalization, as for LTP.

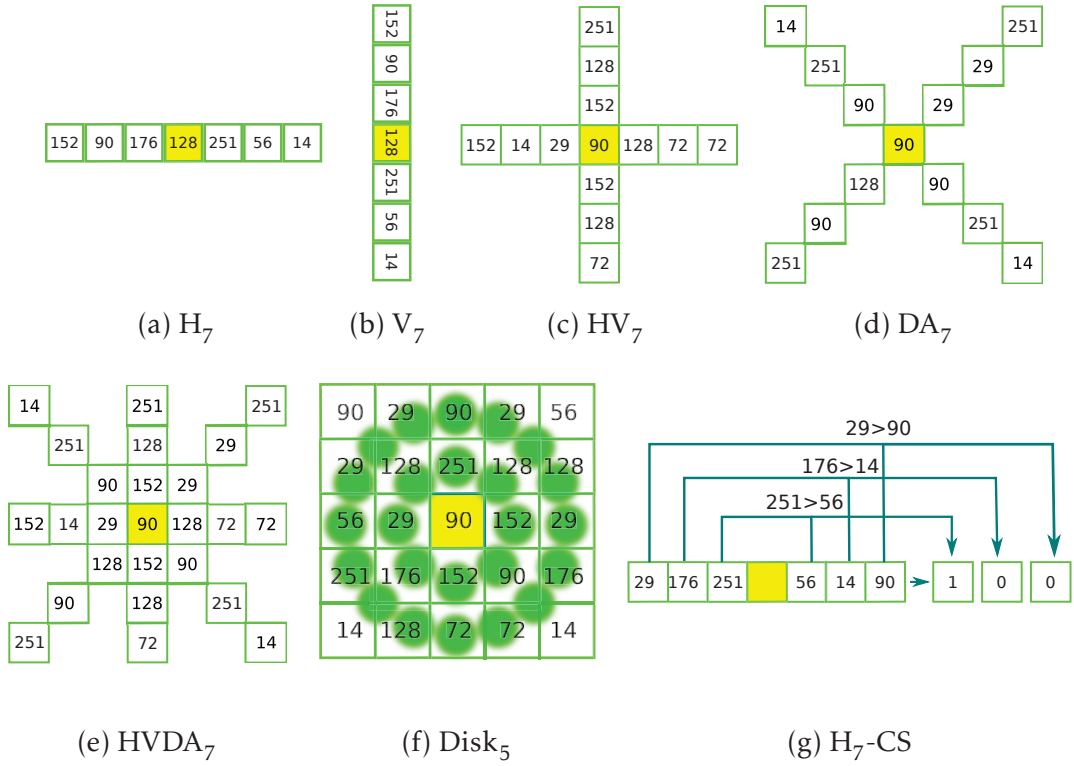


Figure A.1: *Some examples of the local pattern geometries that we have tested. The subscript denotes the diameter of the pixel-level sampling region. In each case, pixels are sampled around a central pixel (shaded yellow) and compared either with the central pixel or with the diametrically opposite pixel (CS case).*

A.1.2 Local Pattern Geometry and Sampling

As already mentioned, for vector quantization tables of practical size, the number of pixel comparisons in a local pattern neighbourhood is limited to at most about 16 for ternary coding, and 24 for binary or split ternary coding[1]. Within these limits many different neighbourhood geometries are possible and one of the main advantages of LQP is the fact that it allows efficient codes to be learned for any neighbourhood geometry, thus encouraging experimentation to find the geometries most suited to the given class. Here we test a few selected geometries corresponding to horizontal (H), vertical (V), diagonal (D), and antidiagonal (A) strips; combinations of these like horizontal-vertical (HV), diagonal-antidiagonal (DA) and horizontal-vertical-diagonal-antidiagonal (HVDA); and traditional circular or rectangular arrangements – *c.f.* Figure A.1. Most of these compare each non-central pixel to the central one, but Center Symmetric ("CS") codes, where each pixel is compared to the diametrically opposite one, are also considered. Below,

[1] Various bit-level implementation strategies could be used to increase these limits somewhat, but we have not yet investigated this in detail.

geometries will be denoted by notation such as HV_7^3, where HV is the neighbourhood shape (here a horizontal-vertical cross), the subscript 7 indicates the neighbourhood width (a layout 7 pixels in diameter) and the superscript 3 denotes the quantization (here native ternary – 3* denotes split ternary and 2 binary coding). Separate vector quantization tables are learned for the "+1" and "-1" halves of split ternary codes. The results below show that it is almost always worth splitting a ternary code to allow a larger neighbourhood to be used during local pattern computation.

As usual, codes from different types of neighbourhoods can be combined by histogram concatenation at the cell level, in the same way that (*e.g.*) LBP+LTP is formed. This allows more pixels to be included in the combined neighbourhood at the cost of ignoring co-occurrences between the patterns in its different sub-neighbourhoods. However it should be noted that very elongated neighbourhoods such as the strips H_{25} and V_{25} are difficult to handle owing to boundary effects because they are so wide that they cross several cells.

A.1.3 Code Book Learning Method

By default we use K-Means [Elkan 2003] over the standard L_2 inter-vector distance to learn visual codebooks. Our K-Means implementation directly uses the bin counts as weightings during its cluster mean computations. As K-Means is sensitive to its initialization and can get stuck in local minima, it is run ten times with different random initializations and the run that gives the smallest overall vector quantization coding error provides the codebook for visual word generation. We tested both soft and hard coding methods. In soft coding, the lookup table cells store the $c \leq 10$ nearest code centers and each local pattern is soft-quantized against all c codes of its cell using uniform weighting. However we do not use soft coding below as we find that it actually reduces the accuracy relative to hard coding.

For comparison, we also tested a fast discriminative neighbourhood coding method that does not require a preliminary quantization of its input, but that does not offer such dense coverage of the set of possible input values. Specifically, we trained random forests of classification trees and used them to construct the tree-leaf-level visual codes in the same way as [Moosmann et al. 2007, 2008]. To better capture the diversity of the dataset, relatively large numbers of relatively shallow trees were trained. The Random Forests can handle continuous input values so we used raw pixel differences for the input vectors.

In our applications, the K-Means codebooks turn out to be both faster to train and more accurate than Random Forest ones, perhaps because despite its quantized input, K-Means uses an underlying distance metric and thus provides a smoother, more distance-sensitive representation of the input space. For instance, using the H_7^3+V_7^3

LQP Type	H^3	V^3	D^3	H^2	V^2	D^2	H^3-CS	V^3-CS	D^3-CS
Strip Size 7	63.3	61.4	60.5	-	-	-	-	-	-
Strip Size 9	64.9	64.8	61.8	46.4	49.0	47.1	-	-	-
Strip Size 15	67.2	68.4	62.3	54.7	53.8	51.4	59.3	59.1	52.5

Table A.1: *Average Precisions for 1Root$_f$ Latent SVM detectors trained using different types of individual strip LQP features on the INRIA Person dataset. The missing values correspond to features that have so few input codes that they can be directly coded without using LQP,* e.g. *H^3_9-CS has 81 codes in total.*

cell-level combination of horizontal and vertical neighbourhoods with 1Root$_f$ Latent SVM on the INRIA Person dataset, the K-Means coding with 100 centers and hence 200 histogram bins gives 65.3% AP, whereas the Random Forest one with 16 trees of 8 leaf nodes (a 128 bin histogram) gives only 57.7% AP. Increasing the maximum number of leaf nodes to 16 (a 256 bin histogram) increases the AP by just 1.3%. For this reason, we will exclusively use K-Means for the experiments below.

A.2 Results and Discussions

A.2.1 INRIA Person Dataset

Strip Layouts. Table A.1 shows AP's on the INRIA Person dataset for detectors using LQP features based on single horizontal, vertical or diagonal strips of pixels. The single-strip features turn out to be quite weak, giving significantly lower performance than existing feature sets – LBP, LTP and HOG give respectively 74.0%, 79.0% and 79.0% AP on this dataset. However it is at least clear that increasing the length of the strip increases the performance, as does replacing binary codes with ternary ones. The comparable centre-symmetric features (which compare each pixel with its diametrically opposite one, not with the centre one, thus halving the size of the input pattern) consistently give much lower performance – *e.g.* H^3-CS and V^3-CS respectively have AP's of 59.3% and 59.1% – so we will not test them further.

Cross Layouts. Unsurprisingly, including several complementary strips in the LQP neighbourhood can significantly increase the accuracy – *c.f.* Table A.2. For instance, for a 100 word dictionary the cross layout HV^3_7 gives 79.5% AP whereas the cell-level concatenation of the 100 word H^3_7 and V^3_7 LQP histograms gives only 74.6%. If split uniform LTP coding is used instead of LQP coding for the H^3_7 and V^3_7 histograms, the results are still worse – 60.4% AP for a $4 \times 33 = 132$ dimensional histogram. Clearly, the richer co-occurrence statistics that HV^3_7 LQP captures are more useful than the 100 extra

Codebook Size	50	60	80	100	150	200	300
HV_5^3	76.8	76.4	76.1	77.9	77.0	80.0	79.9
$HVDA_5^3$	77.3	77.0	79.8	78.7	79.9	81.5	82.3
HV_7^3	77.6	77.1	77.8	79.5	79.1	79.6	81.4
DA_7^3	75.1	76.1	76.0	77.0	76.3	78.4	79.1
$HVDA_7^{3*}$	80.1	81.1	80.2	80.9	81.7	82.0	82.6
$Disk_5^{3*}$	79.3	79.9	81.3	81.2	82.2	81.3	81.4

Table A.2: *The effect of different LQP geometries and codebook sizes on Average Precisions for single root latent detectors on the INRIA Person dataset.*

codewords of H_7^3+V_7^3. Despite the inclusion of only two orientations, the HV_7^3 results are already slightly better than HOG and LTP on this dataset, both of which give 79.0% AP. In contrast, DA_7^3, which combines diagonal and antidiagonal strips, gives only 77.0% AP– presumably horizontal and vertical slices are more discriminant for people than diagonal ones. Incorporating all four types of strip in a "Union Jack" pattern HVDA further improves the results, although split coding must be used in this case owing to the number of pixels in the pattern.

Disk Layouts. Disk-shaped patterns can do even better. Using 100 word codebooks on the INRIA dataset, the two-ring 24 pixel pattern $Disk_5^{3*}$ (81.2% AP) outperforms both the 16 pixel ($HVDA_5^3$, 78.7% AP) and 24 pixel ($HVDA_7^{3*}$, 80.9% AP) Union Jacks, and also LTP and HOG (both 79.0% AP) – *c.f.* Table A.2. The VOC2006 results below confirm that $Disk_5^{3*}$ has slightly better overall performance than $HVDA_7^{3*}$: presumably because dense circular sampling in a compact neighbourhood captures more of the characteristic class structure than sampling a fixed set of rays in the broader neighbourhood covered by $HVDA_7^{3*}$. In fact, $Disk_5^{3*}$ gives the best results that we are aware of (for an individual feature set and a single root detector) on the INRIA Person dataset, being only 1.6% (0.6% for the 150 word codebook) below the combined feature set LBP+LTP+HOG– *c.f.* Chapter 6 (82.8% AP).

Haar Layout. To see whether it would be useful to include multiscale information, we also tested a Haar wavelet based local pattern. We take 4×4 pixel neighbourhoods around each pixel, apply the Haar wavelet transform, discard the constant term and code the remaining 15 wavelet coefficients using LQP ternary coding. In detail, this involves taking the four 2×2-pixel corner blocks of the neighbourhood and a 2×2 block containing the average of each corner one, and applying 2×2 horizontal, vertical and diagonal Haar filters [Papageorgiou and Poggio 2000; Viola and Jones 2004] to each

τ	0	1	2	3	4	5	7	10	14
Disk_3^3	72.4	77.0	**78.7**	78.0	78.4	77.2	76.9	76.2	73.3
Disk_5^{3*}	75.4	77.1	79.9	80.9	79.9	**81.2**	81.0	80.2	78.9
HV_7^3	63.6	74.0	75.5	76.7	77.7	**79.5**	78.4	79.1	78.2
HVDA_7^{3*}	76.7	79.3	79.9	81.1	80.8	81.2	80.5	80.1	**81.8**
Disk_3^3	21.1	23.6	25.1	26.4	26.8	**27.2**	25.4	25.5	25.2
Disk_5^{3*}	23.8	25.0	26.0	28.8	30.1	31.4	**31.5**	29.7	30.4
HV_7^3	11.3	17.0	17.9	20.9	22.5	23.2	25.4	**26.9**	26.7
HVDA_7^{3*}	23.8	27.4	25.0	28.7	30.0	31.0	31.0	32.1	**32.7**
Disk_3^3	53.0	55.0	55.1	54.2	**56.0**	55.6	55.9	54.8	54.4
Disk_5^{3*}	54.0	55.3	56.1	54.7	55.9	55.9	**56.1**	56.1	55.4
HV_7^3	51.5	51.0	53.1	53.1	53.7	53.7	**54.7**	54.6	53.8
HVDA_7^{3*}	53.1	53.4	54.4	55.4	55.4	55.4	56.3	**56.4**	56.0

Table A.3: *Average Precisions of ternary LQP features on the INRIA Person and VOC2006 person and car datasets for different values of the quantization threshold τ.*

block. The resulting Haar LQP features give slightly better results than HOG and LTP: for 100 word codebooks, 80.7% AP versus 79% AP on INRIA Person, and 29.4% AP versus 25.1% and 28.9% AP on the VOC2006 person class. However the Haar patterns do not equal the performance of the best Disk and HVDA ones on these datasets.

Splitting and LQP Features. With a 118 word codebook, Disk_3^3 (the LQP form of LTP's 8-sample circle of 1 pixel radius) gives identical accuracy to traditional 118-D split uniform LTP coding. Reducing the LQP codebook size to 88 reduces the accuracy by only 0.8%. For the 16-sample radius 2 circle Circ_5^3 (the largest pattern for which unsplit ternary coding is feasible), split coding with 100 word codebooks (200-D histograms) gives 80.9% AP on INRIA whereas unsplit coding with codebook (histogram) sizes of 100 and 200 gives respectively 78.8% and 80.5% AP. Similarly, for the VOC2006 person class, split Circ_5^{3*} gives 28.3% AP while unsplit Circ_5^3 gives respectively 26.0% and 28.6% AP for 100 and 200 word codebooks, and for the VOC2006 car class split Circ_5^{3*} gives 55.0% AP while unsplit Circ_5^3 gives 55.2 % and 54.9% AP. These results in some sense validate the use of split uniform coding in the original LTP. Overall, our results consistently show that splitting causes little loss of discriminative power relative to the equivalent unsplit coding, and that it is beneficial in the sense that it allows larger spatial supports to be used, thus increasing the overall discriminative power.

Ternary Code Threshold. Table A.3 shows the effect of the ternary code threshold τ on the APs of various LQP features on the INRIA Person and VOC2006 person and car classes. For each feature there is a broad range of τ values that gives similar results, but

spatially larger patterns (notably the extended cross layouts) need larger values of τ, presumably because the typical ranges of gray-value variations increase with increasing pattern diameter. Over the full set of classes, $\tau = 5$ turns out to be the best value for both of the Disk diameters and also for LTP, whereas $\tau = 14$ is the preferred value for broad crosses such as HVDA_7^{3*}.

Choice of Codebook. Table A.2 also shows how codebook size affects the performance of various LQP features. As expected – despite some variability owing to the nonconvexity of K-Means learning – larger codebooks typically have better performance. By default we use 100 word codebooks below as they seem to offer a reasonable compromise between descriptor size and performance, but smaller ones still offer very respectable levels of performance and larger ones are often even better.

More generally, features can be quantized using either a single codebook (learned from the positives, the negatives, or the complete training set), or several concatenated codebooks – for example ones learned separately on the positive and negative training sets. Moreover, for positive training we can also learn a separate "cell level" codebook for each cell of the detection window, subsequently quantizing the pixels of each cell of the current window using that cell's positive codebook and the global negative one.

Table A.4 shows the effect of these different codebook learning schemes on the accuracy of INRIA Person detectors using HV_7^3 features. The single-codebook results are better than the multiple-codebook ones and in the multiple codebook case, learning separate cell-level positive codebooks provides only a small increase in the AP so it does not seem to be warranted given its extra complexity. Unsurprisingly, using positives alone for codebook learning is better than using negatives alone – the positive codebooks are trained on structures that are important for characterizing the object class – but (perhaps surprisingly) pooling the positives and negatives during training gives worse results than using either positives or negatives alone. For the binary and split ternary codings, initializing some of the K-Means centers at the LBP/LTP uniform patterns to encourage the latter to be well coded does not change the performance. Similarly, for global positive and negative codebooks, learning the negative codebook first and using it to initialize the positive one does not change the performance.

By default, we therefore use single 100 word codebooks obtained by running K-Means on (all of the R, G and B pixels of the annotation windows of) the positive training data.

Combination With HOG Features. Unfortunately, the LQP features are already so strong that combining them with other feature sets such as HOG only seems to provide modest improvements in accuracy on the INRIA Person dataset – *c.f.* Table A.5. Moreover, most of the improvement is observed for the comparatively weaker classes of LQP fea-

Codebook Type Cell Dimension	Positive 100	Negative 100	Combined 100	Positive & Negative 200	Cell Based 200
HV_7^3	79.5	78.8	77.8	77.0	78.0

Table A.4: *The influence of different codebook organizations on the Average Precisions of detectors using HV_7^3 LQP features on the INRIA Person dataset.*

LQP Type	HV_7^3	DA_7^3	$HVDA_5^3$	$HVDA_7^{3*}$	$Disk_5^{3*}$	$HVDA_9^3$-CS
AP for LQP	79.5	77.0	78.7	80.9	81.2	78.5
AP for HOG+LQP	81.7	80.5	82.8	82.7	82.8	81.8

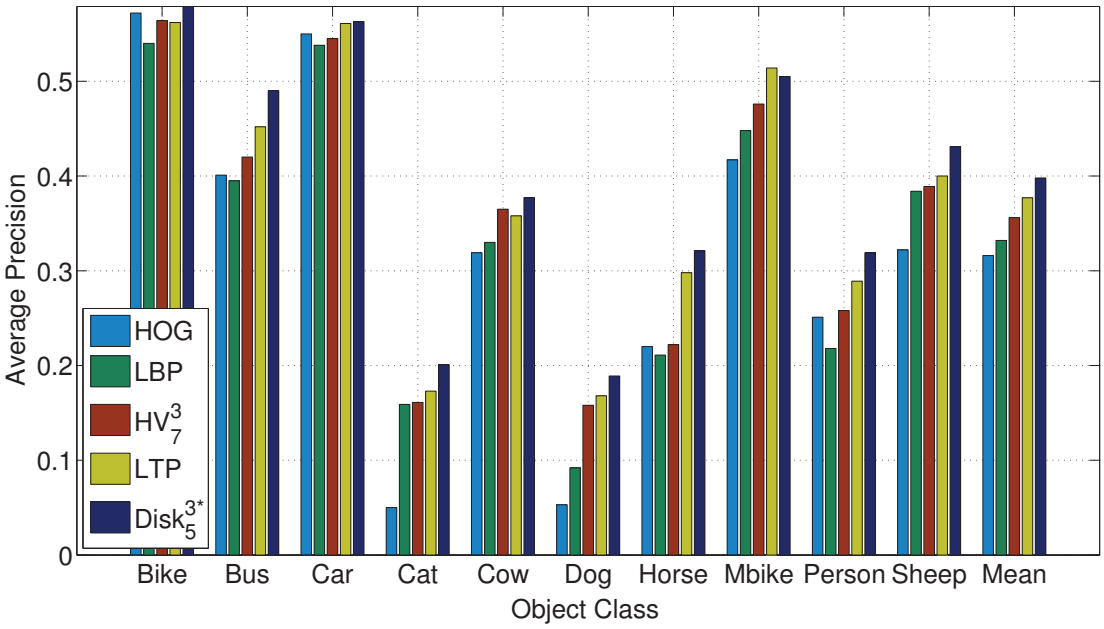
Table A.5: *Average Precisions on the INRIA Person dataset of detectors trained using HOG_{36} plus the given ternary LQP feature. The local patterns use 100 word codebooks. The AP for HOG_{36} alone is 79.0%, and for LTP+HOG_{36} it is 81.3%.*

tures such as $HVDA_5^3$, with more modest one for strong performers such as $Disk_5^{3*}$. This may well be due to the saturation of the (single root) classifiers used on this dataset because the results below show that, like other local pattern features, LQP and HOG are actually quite complementary. Note that exactly the same maximum AP of 82.8% is achieved by LBP+LTP+HOG.

A.2.2 PASCAL VOC2006

LQP features also give state of the art results on the VOC2006 dataset – *c.f.* Figure A.2. For instance, HV_7^3 outperforms HOG_{31} on 8 of the 10 classes, increasing the Mean AP by 4%, and LBP on all 10, increasing the Mean AP by 2.4%. Although LTP still outperforms HV_7^3 on 8 of the 10 classes, $Disk_5^{3*}$ outperforms LTP on 9 of the 10, increasing the Mean AP by 1.0%, and $HVDA_7^{3*}$ on 5 of the 10, increasing the Mean AP by 0.4%. For 150 word codebooks, $Disk_5^{3*}$ even outperforms LBP+LTP+HOG on 6 of the 10 classes, increasing the Mean AP by 0.2% (albeit at the expense of higher dimensionality – 300 descriptors per cell versus 208). These improvements occur for both structure-dominated classes such as cars and people, and texture-dominated ones such as cats and dogs, so LQP seems to be able to capture both types of cues. To the best of our knowledge, LTP and LBP+LTP+HOG were respectively the individual and combined feature sets with the best reported performance on both INRIA Person and VOC2006 (for single root latent detectors using only local features without additional context), so the LQP features $Disk_5^{3*}$ and $HVDA_7^{3*}$ appear to be very competitive across the board.

Finally, as with other local pattern features, combining LQP with HOG leads to



LQP Type	CB Size	Desc Size	**Mean**	Bike	Bus	Car	Cat	Cow	Dog	Horse	Mbike	Person	Sheep
HV_7^3	100	100	34.6	55.8	39.8	55.1	16.0	32.8	14.8	21.8	46.9	25.0	38.2
	150	150	35.6	56.4	42.0	54.5	16.1	36.5	15.8	22.2	47.6	25.8	38.9
$HVDA_5^3$	300	300	36.2	52.8	43.5	**56.8**	14.9	35.9	17.3	26.7	48.4	26.8	38.7
$HVDA_7^{3*}$	100	200	38.3	**58.2**	45.9	56.1	14.6	38.0	17.4	27.7	52.4	32.7	39.8
	150	300	39.4	58.0	46.5	56.5	19.0	**39.0**	17.6	28.9	**52.7**	33.0	42.6
$Disk_5^{3*}$	100	200	38.8	57.8	46.2	56.1	19.3	37.6	18.8	30.9	50.2	29.9	41.2
	150	300	**39.8**	57.9	**49.0**	56.3	20.1	37.7	**18.9**	**32.1**	50.5	31.9	**43.1**
LTP	59	118	37.8	56.2	45.2	56.1	17.3	35.8	16.8	29.8	51.4	28.9	40.0
LBP+LTP+HOG	149	208	39.6	57.4	47.6	55.7	**20.8**	38.2	17.8	31.8	51.0	**34.4**	40.8

Figure A.2: *Average Precisions of single root latent detectors on the VOC2006 test set using HOG_{31}, LBP, LTP and LQP features. Note that simple HV_7^3 outperforms both HOG_{31} and LBP but not LTP, while $Disk_5^{3*}$ outperforms all three and also their combination LBP+LTP+HOG_{31}.*

significant performance improvements on VOC2006 – *c.f.* Table A.6. As expected, the largest improvements occur for the weaker types of LQP features, with more modest improvements for the stronger ones.

A.2.3 Discussion

Given the above results, several points seem clear. Firstly, the fact that splitting ternary codes into their two binary halves leads to little performance loss both validates the use

LQP Type	HV^3_7	DA^3_7	$HVDA^3_5$	$HVDA^{3*}_7$	$Disc^{3*}_5$	$HVDA^3_9$-CS
AP for LQP	25.0	20.7	26.1	30.2	29.9	17.4
AP for HOG+LQP	33.8	30.1	33.1	33.6	34.8	28.8

Table A.6: *Average Precisions on the VOC2006 person class for detectors trained using HOG_{36} plus the given ternary LQP feature. The local patterns use 100 word codebooks. The AP for HOG_{36} alone is 22.2% and for LTP+HOG it is 33.8%.*

of splitting in LTP (and of binary coding in LBP), and suggests that the performance improvements provided by LQP are due mainly to the increased pattern sizes that lookup-table based coding permits, not per se to the absence of splitting or to the replacement of hand-specified codings with adaptive k-means ones. It also suggests that coding orders higher than ternary will give only limited further improvements and that they should be handled by splitting, but these points remain to be tested. Secondly, in agreement with the forms of existing local pattern features, patterns that sample pixels densely in a compact local neighbourhood around the centre seem to give the best performance, so disk-shaped ones are likely to be the best choice for many applications. Thirdly, even for large spatial supports, good results are obtained with quite modest codebook sizes – often even smaller than the corresponding LTP code. In any case, LQP can handle large codebooks with no loss of speed at run time – the issue is whether the subsequent classifier can handle the large histograms that result.

Despite the extra table lookup, LQP features remain very fast to train and test. For instance, HV^3_7, $HVDA^{3*}_7$ and $Disk^{3*}_5$ respectively take about 1.9, 3.1 and 4.5 seconds to scan a VOC2006 image. The extra time for $Disk^{3*}_5$ is due to the need to resample pixels on circles, which takes much longer than the LQP table lookup.

A.3 Summary

This chapter has presented Local Quantized Patterns (LQP), a generalized form of local pattern feature that replaces the traditional hand-built codebook reductions with vector quantization, using precompiled lookup tables to make coding very fast at run time. LQP inherits some of the flexibility and robustness of visual word representations while retaining the efficiency of local pattern coding. These properties allow LQP to outperform traditional local pattern features such as LBP and LTP, and also well-established feature sets like HOG. LQP can encode larger local neighbourhoods with a wider range of the topologies at slightly deeper quantization levels and with customizable output codes, with no noticeable loss of run time speed. The speed gains are provided by using table lookup for the fast vector quantization of local patterns during both training and

evaluation.

Our results on a variety of classes from two datasets validated the LQP approach and suggested that the ability to use larger spatial supports significantly increases the overall accuracy. Moreover, they validated the use of splitting in the original LTP representation of [Tan and Triggs 2010] and suggested that it remains useful for ternary LQP features owing to the larger spatial supports that it allows. Ternary features still give significantly better performance than binary ones and compact disks of pixels consistently outperform less compact arrangements such as strips and cross. The central pixel of the pattern still seems to be the most effective reference for pixel comparisons. Codebook (and hence descriptor vector) sizes remain modest, and codebook learning is fast owing to the lookup table architecture. Also using a single codebook learned from the positive class seems to be the best compromise. Our LQP features give state-of-the-art error rates. *E.g.* HV_7^3 already performs better than HOG and LBP, while Disk_5^{3*} outperforms both these and even the combination LBP+LTP+HOG. These are the best results ever reported for a single feature set on these datasets.

Currently, only a small selection of the many possible LQP configurations has been tested, and only on object detection. LQP is also likely to be useful in other visual recognition tasks such as image classification, face recognition, semantic segmentation, *etc.*, so much remains to be done.

References

S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004. pages 10, 11

T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12), 2006. pages 42, 43

T. Ahonen and M. Pietikäinen. Soft histograms for local binary patterns. *In Proceedings of the Finnish signal processing symposium, FINSIG*, volume 5, page 1. 2007. pages 19, 44

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. *In Proceedings of the Neural Information and Processing Systems, Vancouver, Canada*, 15:561–568, 2002. pages 25

S. Baker and S. Nayar. Pattern rejection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA*. 1996. pages 18

A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. *In Proceedings of the 10th European Conference on Computer Vision, Crete, Greece*, page 127–142. 2010. pages 83

O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using Hough transforms. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA*, pages 2233–2240. IEEE, 2010. pages 11, 16

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *In Computer Vision and Image Understanding*, 110(3):346–359, 2008. pages 13

I. Biederman. Recognition-by-components: A theory of human image understanding. *In Psychological review*, 94(2):115, 1987. pages 3

M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. *In Proceedings of the 9th European Conference on Computer Vision, Marseille, France*, pages 2–15, 2008. pages 13, 15

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *In Journal of Machine Learning Research*, 6:1579–1619, 2005. pages 24

A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. *In Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007. pages 13

G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, USA*, volume 1, pages 710–715. 2005. pages 10

L. Bourdev and J. Brandt. Robust object detection via soft cascade. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, USA*. 2005. pages 102

C. Cortes and V. Vapnik. Support-vector networks. *In Journal of Machine Learning Research*, 20(3):273–297, 1995. pages 14

G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22. 2004. pages 116

N. Dalal. *Finding People in Images and Videos*. Ph.D. thesis, Institut Polytechnique de Grenoble, 2006. pages 5, 10, 11, 17, 21, 22, 23, 28, 31, 32, 37, 39, 54, 74

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, USA*, pages 886–893. 2005. pages 6, 11, 12, 13, 14, 15, 19, 23, 32, 37, 38, 54, 72, 81, 82, 83, 84

S. de Jong. SIMPLS: an alternative approach to partial least squares regression. *In Chemometrics and Intelligent Laboratory Systems*, 18(3):251–263, 1993. pages 19, 58, 59

C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*. 2009. pages 17, 112

P. Dollàr, B. Babenko, S. Belongie, P. Perona, and Z. Tu. Multiple component learning for object detection. *In Proceedings of the 9th European Conference on Computer Vision, Marseille, France*. 2008. pages 10, 13, 14, 84, 85

P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. *In Proceedings of the 20th British Machine Vision Conference, London, England*. 2010. pages 83

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, Miami, USA*. IEEE, 2009. pages 33

C. Elkan. Using the triangle inequality to accelerate K-Means. *In Proceedings of the 20th International Conference on Machine learning, Washington, USA*, volume 20, pages 147–153. 2003. pages 119

A. Ess, B. Leibe, and L. V. Gool. Depth and appearance for mobile scene analysis. *In Proceedings of the 11th IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil*. 2007. pages 81, 105

A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A mobile vision system for robust multi-person tracking. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, USA*. IEEE Press, 2008. pages 81, 105

M. Everingham, L. van Gool, C. Williams, C. Winn, and A. Zisserman. PASCAL Visual Object Classes Challenge results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010a. pages 3, 4, 5

M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge. *In International Journal of Computer Vision*, 88(2):303–338, 2010b. pages 3, 34, 81, 92

M. Everingham, L. van Gool, C. Williams, and A. Zisserman. PASCAL Visual Object Classes Challenge results. http://www.pascal-network.org/challenges/VOC/voc/, 2006. pages 37

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *In Journal of Machine Learning Research*, 9:1871–1874, 2008. pages 20, 24, 65, 67

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, San Francisco, USA*, pages 2241–2248. IEEE, 2010a. pages 18

P. Felzenszwalb, R. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/~pff/latent-release4/, 2010b. pages 22, 82, 93, 94, 96, 97, 101, 103, 104, 105, 106

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. pages 6, 7, 11, 12, 13, 14, 15, 17, 19, 21, 22, 23, 26, 28, 29, 31, 32, 37, 38, 39, 40, 54, 55, 72, 82, 86, 93, 94, 96, 97, 103, 104, 107

P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, USA*, pages 66–75. 2000. pages 11

P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *In International Journal of Computer Vision*, 61(1):55–79, 2005. pages 10

P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, USA*. 2008. pages 25, 29, 93, 94, 95, 103, 104

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, Madison, USA*. 2003. pages 5, 10, 11, 14

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, San Diego, USA*. IEEE, 2005. pages 10

M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *In IEEE Transactions on Computers*, 100(1):67–92, 1973. pages 11

W. Förstner. Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *In Computer Vision, Graphics and Image Processing*, 40:273–310, 1987. pages 10

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *In Journal of Machine Learning Research*, 10:2157–2192, 2009. pages 24

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *In Proceedings of Computational Learning Theory*, pages 23–37. Springer, 1996. pages 14, 16

G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. Technical Report 02-03, Data Mining Institute, University of Wisconsin, 2002. pages 66

J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA*. IEEE, 2009. pages 10, 11, 16

C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004. pages 13, 14, 16

D. M. Gavrila and V. Philomin. Real-time object detection for smart vehicles. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, pages 87–93. 1999. pages 12, 13

D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1239–1258, 2010. pages 83

G. Gualdi, A. Prati, and R. Cucchiara. Multi-stage sampling with boosting cascades for pedestrian detection in images and videos. *In Proceedings of the 10th European Conference on Computer Vision, Crete, Greece*, page 196–209. 2010. pages 83

A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*. 2011. pages 112

R. Haralick. Statistical and structural approaches to texture. *In Proceedings of the IEEE*, volume 67, pages 786–804. IEEE, 1979. pages 13, 41, 85

H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*, pages 237–244. IEEE, 2009. pages 13, 15, 17, 18, 19, 103, 106, 112

M. Heikkilä, M. Pietikäinen, and C. Schmid. Description of interest regions with local binary patterns. *In Pattern recognition*, 42:425–436, 2009. pages 44, 46, 54

S. Hussain and B. Triggs. Feature sets and dimensionality reduction for visual object detection. *In Proceedings of the 21st British Machine Vision Conference, Aberystwyth, England*, pages 112.1–112.10. 2010. pages 42, 44, 49

S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *In International Journal of Computer Vision*, 43(1):45–68, 2001. pages 10

S. Ito and S. Kubota. Object classification using heterogeneous co-occurrence features. *In Proceedings of the 11th European conference on Computer vision: Part V*, page 701–714. 2010. pages 83, 84, 85

A. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *In Pattern recognition*, 24(12):1167–1186, 1991. pages 41

V. Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*. IEEE, 2011. pages 112

T. Joachims. Making large-scale SVM learning practical. *In* B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. The MIT Press, Cambridge, MA, USA, 1999. pages 14, 23, 82

T. Kadir and M. Brady. Scale, saliency and image description. *In International Journal of Computer Vision*, 45(2):83–105, 2001. pages 10

A. Kembhavi, D. Harwood, and L. Davis. Vehicle detection using partial least squares. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. pages 60, 63

A. Kläser. *Learning human actions in video*. Ph.D. thesis, Université de Grenoble, 2010. pages 116

A. Kläser, M. Marszalek, C. Schmid, and A. Zisserman. Human focused action localization in video. *In International Workshop on Sign, Gesture, and Activity (SGA) in Conjunction with ECCV*. 2010. pages 111

C. Lampert and M. Blaschko. A multiple kernel learning approach to joint multi-class object detection. *In Pattern Recognition*, pages 31–40, 2008. pages 17

C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, USA*, pages 1–8. ieee, 2008. pages 13, 18

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *In Journal of Machine Learning Research*, 10:777–801, 2009. pages 65

I. Laptev. Improving object detection with boosted histograms. *In Image and Vision Computing*, 27(5):535–544, 2009. pages 16

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, USA*, volume 2, pages 2169–2178. 2006. pages 13

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 86(11):2278–2324, 1998. pages 13, 14

B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. *In Workshop on Statistical Learning in Computer Vision – ECCV*, pages 17–32. 2004. pages 5

B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *In International Journal of Computer Vision*, 77(1):259–289, 2008. pages 10, 11

T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. *In Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, Greece*, pages 1010–1017. 1999. pages 116

Z. Lin and L. S. Davis. A pose-invariant descriptor for human detection and segmentation. *In Proceedings of the 9th European Conference on Computer Vision, Marseille, France*. 2008. pages 83

Z. Lin, G. Hua, and L. S. Davis. Multiple instance features for robust part-based object detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA*. 2009. pages 10, 14, 83

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *In International Journal of Computer Vision*, 60(2):91–110, 2004. pages 10, 13, 38, 39

S. Maji and A. Berg. Max-margin additive classifiers for detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA*, pages 40–47. IEEE, 2009. pages 15, 75

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, Anchorage, USA*. 2008. pages 15, 16, 33, 74, 75, 77

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, Miami, USA*, pages 1038–1045. IEEE, 2009. pages 10, 11

K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. *In Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*, volume I, pages 69–81. 2004. pages 10

A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001. pages 10, 11, 13, 15

F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1632–1646, 2008. pages 119

F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. *In Proceedings of the Neural Information and Processing Systems, Vancouver, Canada*, 19:985, 2007. pages 119

T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *In Pattern Recognition*, 29:51–59, 1996. pages 6, 41, 42, 43

T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002. pages 42, 43

A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. *In Proceedings of the 8th European Conference on Computer Vision, Graz, Austria*, pages 575–588. Springer, 2006. pages 10, 11, 14

M. Osadchy, Y. L. Cun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *In Journal of Machine Learning Research*, 2007. pages 16

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detecition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 130–136. 1997. pages 23

P. Ott and M. Everingham. Implicit color segmentation features for pedestrian and object detection. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*, page 723–730. 2009. pages 38, 84, 106

P. Ott and M. Everingham. Shared parts for deformable part-based models. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*. 2011. pages 12

C. Papageorgiou and T. Poggio. A trainable system for object detection. *In International Journal of Computer Vision*, 38(1):15–33, 2000. pages 11, 13, 14, 15, 121

D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. *In Proceedings of the 10th European Conference on Computer Vision, Crete, Greece*. 2010. pages 90, 112

M. Pedersoli, J. Gonzàlez, A. Bagdanov, and J. Villanueva. Recursive coarse-to-fine localization for fast object detection. *In Proceedings of the 10th European Conference on Computer Vision, Crete, Greece*, pages 280–293. Springer, 2010. pages 18

M. Pedersoli, A. Vedaldi, and J. Gonzàlez. A coarse-to-fine approach for fast deformable object detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*. 2011. pages 18

F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, USA*, volume 1, pages 829–836. 2005. pages 54

D. Ramanan. Using segmentation to verify object hypotheses. *In 2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. pages 17, 84, 90, 112

M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, USA*. 2007. pages 112

N. Razavi, J. Gall, and L. V. Gool. Scalable multi-class object detection. *In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1505–1512. 2011. pages 10

R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. *In Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume IV, pages 700–714. 2002. pages 10, 11

R. Rosipal and N. Kramer. Overview and recent advances in partial least squares. *In Lecture notes in computer science*, pages 34–51, 2006. pages 58

P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, USA*, pages 1–8. IEEE, 2007. pages 33

V. Sande, K. E. A. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010. pages 48

R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *In Journal of Machine Learning Research*, 37(3):297–336, 1999. pages 14, 16

C. Schmid. A structured probabilistic model for recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, USA*, volume II, pages 485–490. 1999. pages 116

H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *In International Journal of Computer Vision*, 56(3):151–177, 2004. pages 13

P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele. Discriminative structure learning of hierarchical representations for object detection. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, USA*, pages 2238–2245. IEEE, 2009. pages 10

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, San Francisco, USA*. IEEE, 2010. pages 10, 11, 14

B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002. pages 14

W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*. 2009. pages 13, 16, 19, 60, 65, 78, 83, 84, 85, 105, 106

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *In Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):411–426, 2007. pages 112

S. Shalev-Shwartz and A. Tewari. Stochastic methods for L1 regularized loss minimization. *In Proceedings of the 26th International Conference on Machine learning, Montreal, Canada*, pages 929–936. 2009. pages 66

E. Shechtman and M. Irani. Matching local self-similarities across images and videos. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, USA*, pages 1–8. IEEE, 2007. pages 13

J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. *In Proceedings of the 10th IEEE International Conference on Computer Vision, Bejing, China*. IEEE Computer Society, 2005. pages 10, 11

K. Sung and T. Poggio. Example-based learning for view-based human face detection. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998. pages 23

X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *In IEEE Transactions on Image Processing*, 19(6):1635–1650, 2010. pages 6, 7, 19, 43, 44, 45, 47, 48, 106, 109, 127

M. Tipping. Sparse bayesian learning and the relevance vector machine. *In Journal of Machine Learning Research*, 1:211–244, 2001. pages 65

A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007. pages 16, 64, 113

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *In Journal of Machine Learning Research*, 6(2):1453, 2006. pages 25

O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. *In Proceedings of the 8th European Conference on Computer Vision, Graz, Austria*, pages 589–600. Springer, 2006. pages 13

O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1713–1727, 2008. pages 13, 83, 84, 85

M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination indepence. *In Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume III, pages 255–271. 2002. pages 116

A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*. 2009. pages 10, 13, 15, 18, 19, 52, 74, 103, 104, 106

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA*. 2010. pages 15, 75, 80, 83, 112

S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*. IEEE, 2011. pages 113

P. Viola and M. J. Jones. Robust real-time face detection. *In International Journal of Computer Vision*, 57(2):137–154, 2004. pages 11, 13, 14, 16, 17, 18, 31, 37, 64, 121

B. Wandell. *Foundations of vision.* Sinauer Associates, 1995. pages 48

X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. *In Proceedings of the 12th IEEE International Conference on Computer Vision, Kyoto, Japan*. 2009. pages 13, 15, 19, 42, 43, 83, 84, 85, 106

X. Wang, X. Zhou, T. X. Han, S. Tang, G. Chen, K. Yu, and T. S. Huang. Liblinear SVM with HOG-LBP and bag of words (DHOG) features. VOC Object Detection Challenge, 2010. pages 104, 105

S. Wold, M. Sjöström, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *In Chemometrics and Intelligent Laboratory Systems*, 58(2):109–130, 2001. pages 19, 58, 59

B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *In International Journal of Computer Vision*, 75(2):247–266, 2007. pages 10

B. Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. *In Proceedings of the 9th European Conference on Computer Vision, Marseille, France*. 2008. pages 83, 84, 85

B. Wu and R. Nevatia. Detection and segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. *In International journal of computer vision*, 82(2):185–204, 2009. pages 11

H. Yu, F. Huang, and C. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *In Journal of Machine Learning Research*, pages 1–35, 2011. pages 66

Y. Yu, J. Zhang, Y. Huang, S. Zheng, W. Ren, C. Wang, K. Huang, and T. Tan. Object detection by context and boosted HOG-LBP. VOC Workshop Talk, 2010. pages 104, 105, 112

J. Zhang, K. Huang, Y. Yu, and T. Tan. Boosted local structured HOG-LBP for object localization. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA*, pages 1393–1400. IEEE, 2011. pages 104

Y. Zheng, C. Shen, and X. Huang. Pedestrian detection using center-symmetric local binary patterns. *In Proceedings of the* IEEE *International Conference on Image Processing, Hong Kong, China*, volume 1, page 1. 2010. pages 47, 53, 54

J. Zhu, A. Ahmed, and E. Xing. MedLDA: maximum margin supervised topic models for regression and classification. *In Proceedings of the 26th International Conference on Machine learning, Montreal, Canada*, pages 1257–1264. ACM, 2009. pages 58

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, San Francisco, USA*. IEEE, 2010. pages 11, 12, 15, 112

Q. Zhu, S. Avidan, M. Ye, and K. Cheng. Fast human detection using a cascade of histograms of oriented gradients. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, USA*. 2006. pages 11, 13, 18, 38, 54, 83, 85