Doctoral Thesis

Scalable Visual Recognition Using a Shared Vocabulary

Author(s):
Razavi, Nima

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007593593

Rights / License:
In Copyright - Non-Commercial Use Permitted
Scalable Visual Recognition using a Shared Vocabulary

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

Nima Razavi
M.Sc. ETH Zürich
born 2. September 1984
citizen of Iran

accepted on the recommendation of

Prof. Dr. Luc Van Gool, ETH Zurich and KU Leuven, examiner
Prof. Dr. Bernt Schiele, MPI and Saarland University, co-examiner
Dr. Jürgen Gall, MPI for Intelligent Systems, Tübingen, co-examiner
To my family.
Abstract

Building autonomous systems that enable machines to understand and interpret images is a long-standing goal in the field of computer vision. Solving this problem will have a huge impact on the society as it leads to many life-changing applications such as autonomous driving, image search, and surveillance just to name a few. Visual object detection is a key component of such systems that makes sense of an input image by establishing a correspondence of its content to previously observed data and compactly represents this information in forms of labels.

The primary goal of this dissertation is to present a scalable non-parametric visual object detector with a shared vocabulary. Scalability of object detectors with respect to the number of classes and training images is a critically important issue for applications with a large number of classes. Detecting objects with a shared vocabulary is also very attractive as it facilitates better generalization and incremental learning of the detector.

This thesis shows how to learn and utilize a shared vocabulary to detect objects accurately and in a scalable manner. To this end, it argues that the vocabulary entries need to be generalizable across instances and yet remain as discriminative as possible. This way, by matching a patch to the vocabulary, a sparse and reliable distribution of object labels and configurations can be predicted. It is proposed to learn the vocabulary by simultaneously maximizing the sparsity of the entries and controlling their complexity to ensure generalization. Also, a data-driven semantic hierarchy acting on the patch level is introduced and used to speed up the detection.

In addition to the category label and location, it is desired in many applications to retrieve auxiliary information for a detected instance such as type and pose. For this purpose, this thesis proposes to describe a detection by the configuration and appearance of its supporting patches.
Using this description, a novel occlusion-insensitive similarity measure between two detections is introduced. The proposed similarity measure is used to retrieve similar previously seen objects; transferring their information to the new detection. The support of a hypothesis is also used for tracking where the detector is adapted to the appearance of an instance.

When detecting objects in an image, it is important to only combine consistent patches into a hypothesis. Semantic properties such as pose or color can be used for detecting consistent hypotheses. However, these properties are often either unobserved in the training data or not clearly known to be optimal for enforcing consistency. This thesis proposes to treat these additional properties as global and local latent variables and discriminatively learn their optimal assignments for the training data. This way, only patches consistent in their latent assignments are combined into an object hypothesis, substantially increasing the detection accuracy.

All methods presented in this thesis are evaluated on challenging and realistic benchmark databases. The experiments confirm the benefits of using a discriminative shared vocabulary of middle complexity patches as a building block for visual recognition tasks such as scalable multi-class object detection, tracking, and object property estimation.
Zusammenfassung


Für Objekterkennung in Bildern ist es wichtig nur konsistente Patches zu einer Hypothese zu kombinieren. Semantische Eigenschaften wie Pose und Farbe können benutzt werden, um konsistente Hypothesen zu erken

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Luc van Gool. Thank you Luc for giving me the opportunity and freedom for working on this subject. Your advice and guidance has been invaluable to my research. I would also like to thank my co-examiners Prof. Bernt Schiele and Dr. Juergen Gall for accepting to be on my thesis committee.

My sincere thanks also goes to my advisors Dr. Juergen Gall and Dr. Pushmeet Kohli. It has been a real pleasure working with both of you. I have learned a great deal from you during these years and I hope our cooperation continues in the future.

I would also like to thank my previous supervisor and mentor Prof. Mehrdad Shahshahani for introducing me to the fascinating field of computer vision. Mehrdad, I have been very fortunate to be supervised by you during the years of my bachelor. Your advice and attitude towards research and life has changed me forever and in a very positive way.

My special thanks goes to Peter Gehler for discussions on scalability and Thomas Deselaers for helping with the condor scripts. Also, I would like to thank my students Pascal Bissig, Nima S. Alvar, Michael Schaffner, Xiang Fang and Daniel Wirz whose work have influenced this thesis in one way or another.

I would like to thank my friends and colleagues and all the BIWI members for making the life and research much more enjoyable during these years. I thank Julien Egger, Alex Mansfield, Andreas Ess, Thibaut Weise, Raphael Hoever, Valeria De Luca, Christian Leistner, Ali Farhadi, Masoud Alipour, Sara Saffarian, Chris Russ, Matthias Schneider, Lisa Gruebel, Stephan Verhasselt, Hossein Gorji, Payam Payamyar, Fredi Tischhauser, Gabriel Weilenmann, Fabienne Brigger, Samira Jafari, Mahmoud Jafargholi, Gregory Paul, Veronika Graml, Thomas Deselaers,
Vittorio Ferrari, Marko Ristin, Mukta Prasad, Gabriele Fanelli, Fabian Nater, Sven Hirsch, Bjoern Menze, Stephan Gammeter, Peter Gehler, Matthias Dantone, Angela Yao, Lukas Bossard, Julien Weissenberg, Peter Baki, Orcun Goksel, Matthieu Guillaumin, and all the other BIWI members. I am also thankful to the BIWI secretaries Barbara Widmer, Christina Krueger, and Fiona Matthews for their great organizational work.

Last, but not least, I would like to thank Golnoosh for her understanding and love during the past few years. Her support and encouragement was in the end what made this dissertation possible. My parents, Mahnaz and Alireza, receive my deepest gratitude and love for their dedication and the many years of support during my undergraduate studies that provided the foundation for this work.
# Contents

1 Introduction 3  
  1.1 Contributions ................................................. 6  
  1.2 Organization .................................................. 8  

2 Visual Object Detection:  
an overview 9  
  2.1 Parametrization .................................................. 9  
  2.2 Object Detection ................................................ 11  
      2.2.1 Template Matching ........................................ 13  
      2.2.2 Bag of Words ............................................ 14  
      2.2.3 Comparison ............................................... 16  
  2.3 Advances from Template Matching ....................... 19  
      2.3.1 Non-linear Template Matching .......................... 19  
      2.3.2 Part-based Models ...................................... 20  
      2.3.3 Exemplar Models ....................................... 23  
  2.4 Advances from Bag of Words ....................... 24  
      2.4.1 Spatial Pyramid Matching ............................... 25  
      2.4.2 Implicit Shape Model ................................. 26  

3 Scalable Multiclass Detection 31  
  3.1 Related Work .................................................. 35  
  3.2 Detection with an ISM ........................................ 37  
  3.3 Complexity Analysis ......................................... 39  
  3.4 Proposed Multi-Class Detector ............................. 42  
      3.4.1 Training the Shared Codebook ......................... 42  
      3.4.2 Measuring Sharing ..................................... 47  
      3.4.3 Building the Taxonomy ................................. 48  
      3.4.4 Detection with the Taxonomy ......................... 48  
      3.4.5 Verification .......................................... 50
1

Introduction

Looking at the picture in Fig. 1.1, what can you see? As humans, we can instantly recognize the animals, sky, ground and the hills. We can also tell that there are four animals in this picture and determine their pose and distance to the camera. We can even tell that there is a tree next to the animals and they are enjoying its shadow. Although we may not know the exact name of the species for animals, in this case Asiatic Cheetah (*acinonyx jubatus venaticus*), we are able to describe them as dotted cats with round ears. Despite being trivial for humans, modern computer vision algorithms are still not able to answer such questions about an image automatically.

Building autonomous systems that are able to understand and interpret images as accurate as humans is a long-standing goal in the field of computer vision. Solving this problem will have a huge impact on the society as it can lead to many life-changing applications such as autonomous driving, image search, and surveillance just to name a few.

Visual recognition is a key component of such systems that makes sense of an input image by establishing an association of its content to previously observed data. Inspired from language, this information is usually compactly represented in forms of labels. In principle, these labels can correspond to any semantic pattern, e.g. an object (person), a part of an object (leg), a group of objects (person sitting in a car), an action (person running), or even the whole scene (a football match). This thesis deals with the problem of localization and classification of these recurring semantic patterns in an image. In this thesis, we refer to this task as visual recognition.
1. Introduction

Figure 1.1: How many animals are in this photograph? Where are they located? From which species? Can you describe how they look like? As humans, we are able to instantly answer many such questions about an image. Despite recent progress, performing these tasks using computer vision remains largely an unsolved problem. In this thesis, we refer to these tasks as visual recognition and investigate a non-parametric approach for solving it.

Visual recognition is a particularly challenging problem. First of all, an image is a 2D projection of a 3D environment which introduces significant variations in appearance due to geometric distortions and lighting. The individual instances of an object category can also greatly vary in appearance and a detection algorithms needs to be able to generalize to new instances. Also, objects do not appear in isolation and interact with their environment and can be occluded by other objects in a scene.

Despite the early efforts to model the underlying 3D structure of a scene [Roberts 1963, Binford 1971], the modern approaches for object detection model the appearance of the objects in 2D. To this end, an object in an image is represented by a set of parameters, e.g. a bound-
ing box, and for each of which a set of appearance-based features are extracted. Using these features, the task of detection is then cast into classification of all set of hypotheses in an image into correct labels.

In the recent years, the performance of the state-of-the-art methods for object detection has improved substantially. This improvement in performance can be associated with the advancements in many components such as classifiers, features, or the availability of more training data. However, arguably the most important factor behind this improvement is the representation of object categories.

In this dissertation, we investigate the use a patch-based non-parametric representation for visual recognition that closely following the Implicit Shape Models [Leibe et al. 2008]. In this representation, the training images are divided into a set of patches and all information associated to each of which, such as instance id and location is stored. The task of recognition in this approach is finding the most consistent association of the training patches to the test image.

In this thesis, we show how to perform scalable detection in the number of classes by using a patch-based representation. One of the main contributions of this thesis is to analyze the computational complexity of the detector in the number of classes and show how this scaling behavior can be improved by using a shared vocabulary for recognition and training it in a particular way. A shared vocabulary, in this thesis refers to a (possibly overlapping) grouping of the training patches into visual words.

Using a patch-based representation for recognition of object categories leads to very flexible non-parametric and scalable detectors. However, it is very important that only patches that are consistent with one another are combined into a single hypothesis. The Hough-transform based detector, however, only enforces the consistency of patches in their relative locations which leads to creation of false positives with inconsistent patches. It is our understanding that this is the main reason behind the poor performance of these approaches. to overcome this problem, we propose the latent Hough transform (LHT) for enforcing consistency among patches. The general idea is to augment the Hough space with a latent space and vote for this joint space instead of only for locations. We further show how arbitrary latent spaces can be learned discriminatively
from the training data. The proposed formulation for learning an LHT model, generalizes a number of previous approaches for object detection and the latent variable models in general. Several previous works can be seen as a special case of this approach.

Moreover, the object detection is commonly just an element of a larger scene understanding problem and not the end output. To this end, it makes sense to be able to extract as much information as possible from the detector and reuse it in the other stages. For example, in a tracking scenario one is interested in following a specific person throughout a number of frames. Likewise, determining the pose of the detected objects is crucial in a robotic application and enables accurate interactions of the robot with the object. State-of-the-art approaches deal with this problem in a two stage where the object is detected at the category level and another classifier is trained to detect the views. In this thesis, we propose an alternative way where all information used for detecting an object instance is used as a signature for describing that instance. The advantage of this approach over the two-stage approaches is two fold. On the one hand, the generic knowledge of the category helps in separating the object instance from its background. On the other hand, the resulting system is faster than the two stage approach since most of the information is reused at the later stages.

1.1 Contributions

• **Scalable multi-class detection:** In this thesis, we give an in-depth complexity analysis of object detection using a shared vocabulary. Based on this analysis, we derive the necessary conditions for having sub-linear detection complexity in the number of categories. In particular, we show that the scalability can only be achieved when the individual codewords are able to discriminate between classes. We empirically demonstrate this by training a multi-class detector with a very sub-linear complexity in the number of classes and show that training such a detector does not come at the cost of reduced accuracy.

• **Data-driven class hierarchies:** By using a shared vocabulary for multi-class detection, we implicitly let the appearance of local
patches to be shared across different categories. The sharing structure reveals very interesting relations between the appearance of the categories. We use the sharing structure to create class taxonomies and use them to speed-up the detection time.

- **Patch-based description of an instance:** In the proposed multi-class detector, an object hypothesis is detected by accumulating the consistent votes from a set of training patches. In this thesis, we show how these patches and their configurations can be used to describe an object instance. Based on this description, we further define a distance between two detections and use it to retrieve auxiliary information about a detected object and track an object instance in a video.

- **Latent Hough transform:** In order to enforce higher order consistency among patches we propose to augment the Hough transform with local and global latent variables. Our experimental results suggest that, given enough training data, the detection performance can substantially increase by using these latent variables. Yet, such improvements depend very much on the latent variable used and their quantization. In this thesis, we give an optimization framework for discriminative learning of a latent space such as to directly improve detection performance. Moreover, we show how this formulation can be extended to local latent variables to further improve performance.

- **Generalized Latent Assignments:** In training a latent variable model, the values of the latent variables for training data needs to be estimated. Traditional approaches, use a hard assignment of the data points to latent variables. For training an LHT model, we generalize this notion and propose a real-valued soft memberships of data points to latent variables and represent these assignment using a latent matrix. This representation allows for modeling the uncertainty involved in these assignments and makes the model robust with respect to the number of latent variables.
1.2 Organization

The rest of this dissertation is organized as follows.

Chapter 2: An overview of the method for object detection in computer vision is given in this chapter. In particular, using a common probabilistic framework, the two approaches of template matching and bag-of-words model are first described. Recent advancements of each method and their advantages and disadvantages are then discussed in detail.

Chapter 3: The description of the proposed multi-class detector and complexity analysis of multi-class object detection with a shared vocabulary are first discussed. It is then described how to get the class hierarchy from the shared vocabulary and how to use it to further speed-up the detector. Parts of the contributions in this chapter has been appeared in [Razavi et al. 2011, Gall et al. 2012].

Chapter 4: This chapter gives the definition of the support of an object hypothesis and how it can be used to describe an object instance. Three applications of the support for estimated bounding boxes, viewpoints and tracking an instance over time is discussed. The material in this chapter has been previously published in [Razavi et al. 2010, Gall et al. 2010].

Chapter 5: The main topic of this chapter is the introduction of Latent Hough Transform as a novel latent variable model for object detection. In this chapter, we show how the optimal categories for the task of detection can be learned using this model. Further, a comparison of this model to other approaches and the way it generalizes these is discussed. The latent Hough transform has been introduced in [Razavi et al. 2012a].

Chapter 6: In this chapter, we explain how one can overcome the limitations of the Hough transform model by augmenting it with local latent variables. Further, we will show that the recognition with a visual vocabulary is a special case of such local latent variable models. Some early experiments with this model are also presented in this chapter [Razavi et al. 2012b].

Chapter 7: This chapter concludes the thesis, discusses the contributions and gives directions for the future work.
Visual Object Detection: an overview

The purpose of this chapter is to familiarize the reader to the field of object detection in computer vision. As the first step, we formally define the task of object detection and the common parametrizations used for object hypothesis. We then give an overview of the two most simple yet fundamental methods for object detection using a common probabilistic formulation, namely the bag-of-words model and template matching. Next, we review the recent advances of these two approaches and their advantages and disadvantages over one another. We conclude this chapter by discussing the remaining shortcomings of these approaches that needs to be addressed in the future.

2.1 Parametrization

Throughout this thesis, we will refer to a candidate object hypothesis by \( h \in \mathcal{H} \) where \( \mathcal{H} \) is the space of all possible hypothesis. Every hypothesis defines the spatial extent of a possible object in an image. The task of object detection is to assign a set of labels \( c_h \in \mathcal{C} \) to every hypothesis \( h \) in an image as accurately as possible. In the past, several methods have been proposed to define a hypothesis. Figure 2.2 gives an overview of three popular parametrization.

The most basic way is to parametrize a hypothesis by the position of its center. Since the objects can appear at multiple distances with respect to the camera it is also good to assign a scale to each hypothesis.
2. Visual Object Detection: An overview

Figure 2.1: This figure shows three popular methods for parametrizing the location of an object in an image. (b) An object can be parametrized by the position and scale of a reference point, e.g., center of mass. (c) A bounding box, (d) pixel level segmentation. Although a hypothesis is more accurately described by using more parameters, providing ground truth data becomes more difficult and expensive. A good trade-off is usually found by using bounding boxes [Deselaers and Ferrari 2011].
Evaluation with this criteria is usually performed using the Euclidean distance between the ground truth object and a hypothesis.

The parametrization for the center can be extended to also include the aspect ratio of the object. The combination of the aspect ratio, scale and position can be represented by a bounding box where the hypothesis is defined by the top left and the bottom right corner 2.1. The bounding box is a very lucrative choice as it very well balances the complexity of annotations and its descriptive power.

For some thin objects the bounding box does not represent the extent of the object properly. To better capture the contour, the bounding box can be extended to a polygon and eventually to a pixel-wise segmentation. The advantage of segmentation over other parametrizations is its flexibility to any shape or occlusion. However, segmenting . Evaluation of bounding box and segmentation predictions, usually performed by the Jaccard index defined as

\[ J = \frac{A \cap B}{A \cup B} \] (2.1)

where usually a predicted bounding box overlapping more than 50% with the ground truth is considered correct [Everingham et al. 2010].

### 2.2 Object Detection

Let us assume that we are given an image \( I \) and an object is parametrized by \( h \), e.g. its bounding box. Then the task of object detection is to assign every bounding box \( h \in \mathcal{H}_I \) an object label \( c_h \in \mathcal{C} \) where \( \mathcal{H}_I \) is the space of all hypotheses in image \( I \) and \( \mathcal{C} \) the space of all object categories/instances including a background category.

In probabilistic object detection, given an image \( I \) and a set of hypotheses \( \mathbf{h} = \{h \in \mathcal{H}_I\} \) in an image, one is interested in estimating the posterior distribution \( p(\mathbf{c}|\mathbf{h}, I) \) where \( \mathbf{c} \) is the vector of all labels for all hypothesis in an image, i.e. \( \mathbf{c} = \{c_h|h \in \mathcal{H}_I\} \). Using Bayes’ formula, we derive

\[
p(\mathbf{c}|\mathbf{h}, I) = \frac{p(I|\mathbf{h}, \mathbf{c})p(\mathbf{c}|\mathbf{h})}{p(I|\mathbf{h})} \] (2.2)

Given these distributions, the optimal labeling \( \mathbf{c}^* = \{c^*_h\} \) can be estimated by either the Maximum Likelihood (ML) estimator or obtained
as the Maximum A Posterior (MAP) probability of the joint distribution. In probabilistic terms, the ML estimate can be obtained from the likelihoods $p(I|c, h)$ by

$$c^* = \arg\max_c p(I|c, h)$$

(2.3)

where the $p(I|c, h)$ is the likelihood of image $I$ given all hypothesis labels. Likewise, the MAP solution is obtained from the posterior $p(c|h|I, h)$ as

$$c^* = \arg\max_c p(c|I, h)$$

(2.4)

Without going into detail, as can be seen, the difference of these estimators are in the way they handle prior information over labels $p(c|h)$ and images $p(I|h)$.

In most object detection methods these priors are assumed to be uniform and the labeling of individual hypothesis are assumed independent of one another. To this end, the labeling of each hypothesis $p(c_h|I, h)$ is obtained independently of the other hypothesis. The independence assumption of all hypothesis in an image is very inaccurate. In particular, this assumption does not hold in the case of nearby hypothesis where a feature is very likely to belong only to one of the two nearby objects. This inaccuracy is usually compensated for with various forms of heuristic non-maxima-suppression (NMS). For example, in [Leibe et al. 2008], objects are detected as modes of the posterior by mean-shift [Leibe et al. 2008]. Other approaches [Gall et al. 2011, Barinova et al. 2010], propose a greedy procedure for detection by first detecting a hypothesis and then suppressing its “nearby” hypotheses [Gall et al. 2011] or removing votes from the contributing features [Barinova et al. 2010].

In the recent years, some progress has been made to discriminatively learn the label priors $p(c|h)$ for a given hypothesis labeling, see for example [Desai et al. 2009, Pellegrini et al. 2009, Ladicky et al. 2010, Barinova et al. 2010] and the references thereof. Approaches aiming at pre-filtering the object hypothesis by image saliency [Alexe et al. 2010] or overlapping segmentation [Carreira and Sminchisescu 2010, Endres and Hoiem 2010] also fall into this category. Further, [Sadeghi and Farhadi 2011] jointly estimate the likelihoods for some combination of object hypotheses, i.e. visual phrases.
2.2. Object Detection

2.2.1 Template Matching

One of the most popular object detection strategies is using a sliding-window classifier, e.g. [Viola and Jones 2004, Dalal and Triggs 2005]. In detecting objects with a sliding window classifier, it is assumed that the label of each bounding box can be obtained independently from labels of other bounding boxes; thus, the problem of object detection is reduced to a classification task where the goal is to estimate the distribution $p(I|c_h, h)$ for each hypothesis independently. The labels of each hypothesis can then be

$$c^*_h = \arg \max_{c_h \in C} p(I|c_h, h)$$ (2.5)

Although it is in principle possible to estimate the likelihoods directly from pixel values of an image $I$, to ensure certain invariances and thus increase generalization power of the model, it is common practice to pre-process an image by extracting features. To this end, a number of features $f_i = (l_i, s_i, I_i)$ at locations $l_i \in \mathbb{R}^2$ and scales $s_i \in \mathbb{R}$ are extracted and each of which described with an appearance descriptor $I_i$.

For example, the pedestrian detector of [Dalal and Triggs 2005] is using a linear classifier with HoG descriptors. The image $I$ for the bounding box $b$ is represented by the histogram of the gradient orientations inside and a little around $b$. The spatial bins of this histogram are concatenated to form a high dimensional feature vector. Let us denote every element of this vector by $f_i$. In this case, each $f_i$ has a real valued descriptor $I_i$ and is appearing at pixel $l_i$ at scale $s_i$ of the scale pyramid. Under the naive Bayes assumptions, the likelihood term is written as the product of the individual feature likelihoods

$$p(I|c_h, h) = \prod_i p(f_i|c_h, h)$$ (2.6)

where the mapping from $f_i$ to $p(f_i|c_h, h)$ is learned during training. As can be seen, the mappings in this model not only depends on the appearance $I_i$ of a feature but also, and critically so, on its location inside the bounding box. In a multi-resolution approach like [Park et al. 2010] this mapping can also depend on the scales $s_i$. In these models, the emphasis is more on the rigid location than the appearance as the estimated likelihoods can greatly vary for features appearing at different locations in the bounding box yet with a similar descriptor value.
2. Visual Object Detection: An overview

(a) A pedestrian  (b) HOG features  (c) pos. template  (d) neg. template

Figure 2.2: This figure shows the detector introduced [Dalal and Triggs 2005]. This detector is an example of detection on template matching. In the case of this detector, a template is learned based on Histogram of Oriented Gradients (HOG) features using a linear SVM. (c-d) positive and negative weights for different features in the template. The picture is reprinted from [Dalal and Triggs 2005] with permission.

2.2.2 Bag of Words

Borrowed from the natural language processing community [Gerard and Michael 1983], the bag-of-words model [Csurka et al. 2004, Sivic et al. 2005, Fei-Fei and Perona 2005] is another very powerful set of methods for image classification with independent features. Similar to template matching, in BoW models the image likelihoods given a hypothesis are obtained using the product of individual feature likelihoods

\[ p(I|c_h, h) = \prod_i p(f_i|c_h, h) \] (2.7)

Unlike the detector of [Dalal and Triggs 2005], in the bag-of-words model the spatial relationships of the features inside \( h \) are discarded and instead the emphasis is put on the appearance of the features. In particular, the appearance of every feature \( I_i \) is considered to be a patch around its location \( l_i \) instead of being a simple gradient orientation. In particular,
the feature likelihoods are decomposed into an appearance term and a spatial term.

\[
p(f_i|c_h, h) = \underbrace{p(I_i|c_h, h)}_{\text{appearance}} \cdot \underbrace{p(l_i, s_i|c_h, h)}_{\text{location}}.
\] (2.8)

The location term in this model only takes the presence of a feature in the hypothesis bounding box

\[
p(l_i, s_i|c_h, h) = \begin{cases} 
1 & \text{if } (l_i, s_i) \in h \\
0 & \text{otherwise}
\end{cases}
\] (2.9)

In the bag-of-words model, the appearance term \(p(I_i|c_h, h)\) is estimated indirectly by first matching them to a visual vocabulary. This visual vocabulary is obtained by clustering the visual appearances of training data into a set of codewords. We denote this vocabulary by \(\Omega = \{\omega_1, \ldots, \omega|\Omega|\}\) and refer to the words by \(\omega_j\) indexed by the letter \(j\). In this model, each codebook entry \(\omega_j\), also called visual word, is representing a particular appearance. The appearance of each feature, \(I_i\) is matched to each word \(\omega_j\) to obtain the appearance likelihood \(p(I_i|c_h, h)\):

\[
p(I_i|c_h, h) = \sum_{j=1}^{|
\Omega|} p(I_i|\omega_j, c_h, h)p(\omega_j|c_h, h)
\] (2.10)

where the \(p(I_i|\omega_j, c_h, h)\) determines the likelihood of the appearance \(I_i\) to be generated from codebook \(\omega_j\) for hypothesis \(h\) with label \(c_h\) and the \(p(\omega_j|c_h, h)\) is the prior distribution over encounter of this word given \(c_h\) for hypothesis \(h\).

Matching of a feature to the codebook as explained in Eq.(2.10) requires evaluating the distance of a feature to every codeword that is computationally expensive. To enable faster recognition, many codebook-based approaches [Lazebnik et al. 2006, Fei-Fei and Perona 2005, Leibe et al. 2008] a hard matching procedure is proposed in which the likelihoods are only estimated from the most likely codebook entry

\[
\hat{\omega}_i = \arg \max_{\omega_j \in \Omega} p(I_i|c_h, h, \omega_j)
\] (2.11)

\[
p(f_i|c_h, h) \approx p(I_i|c_h, h, \hat{\omega}_i)p(l_i, s_i|h, c_h)p(\hat{\omega}_i|c_h, h)
\] (2.12)
2. Visual Object Detection: an overview

Figure 2.3: Features of different object classes can share appearance although they do not necessarily also share their location. For instance, the legs of a person and a horse share both appearance (bounding boxes) and location (arrows) whereas the wheels of a bus and car are similar in appearance but not in location (red/blue arrows). Although the location is used as the main source of discrimination in template matching methods, the appearance is used as the only cue in the bag-of-words methods by discarding the location for detection.

by replacing this into Eq.(2.7) we arrive at the equation for obtaining the hypothesis posterior in the bag-of-words model under the independent feature assumption

\[ p(c_h | I, h) \propto \prod_i p(I_i | h, c_h, \omega^i) p(l_i, s_i | h, c_h) p(\hat{\omega}^i | h, c_h). \] (2.13)

In chapter 6, we show that the codebook can be interpreted as a local latent variable model where each codeword indicate a possible assignment of this latent variable. Without going into details, we show how the hard-matching and soft-matching strategies can be seen as maximization and marginalization over this latent variable. For further details, we refer an interested reader to the aforementioned chapter.

2.2.3 Comparison

The comparison of the bag-of-words model and the template matching as the two most basic approaches for recognition is of great importance.
Figure 2.4: Complexity in appearance of a local patch, and thus its discriminative power, can be altered by varying its relative size/resolution to the object. Using small regions of an image only contain basic edge information and are not alone informative for either a possible object’s position (localization) or its class label (classification). Yet, by increasing the size of a feature its discriminative power rapidly increases leading to more precise predictions even specific to an instance or situation.

Despite their similarities and their common independence assumptions, the most important difference between these approaches is the way they treat the spatial relationships of features. On the one hand, in the bag-of-words models [Csurka et al. 2004], the spatial relationships between features is completely discarded leading to models very well suited to handle deformations but incapable of capturing the spatial correlations among features. On the other hand, in the detector of [Dalal and Triggs 2005], a rigid template is learned that makes these models very sensitive to deformations but learning the spatial information much easier. From these models we draw three conclusions.

The first take home message of this comparison is to note two different sources of discrimination in object detection: location and appearance. The appearance term of an object category is based on how visually similar pixels/patches of an instance is matching those of the model of an object category. The location is taking into account how similar the structure/configuration of pixels/patches are to a category’s model. Figure 2.3, gives a visualization of these two sources for discrimination.
The second observation is the relationship between the patch size and complexity and its effect on reliance on location or appearance for object detection. Although in detecting an object with a rigid template as in [Dalal and Triggs 2005], very simple/generic features are extracted and matched to an image putting the emphasis on the location of the features for discrimination, the bag-of-words models achieve similar performance [Lampert et al. 2008] on deformable categories like animals by using larger image patches and relying on their discriminative power in appearance. Figure 2.4 gives an illustration of the effect. The issue of the size and complexity and the information within is discussed in [Ullman et al. 2002] and the merits of middle-complexity features are illustrated in the context of image classification. In Ch. 3, further analysis of the feature size in the context of scalable object detection will be given.

As can be seen in Eq.(2.10), the way the feature likelihoods are estimated in template matching is conceptually very different in a bag-of-words model. Although the likelihoods in template matching are estimated directly from comparing a feature with a template, in Eq.(2.10), the likelihoods are obtained in a more flexible way by using more than one template (code word) and by marginalizing over the conditional likelihoods of patches given the code words. The mixture template models introduced in [Felzenszwalb and McAllester 2010, Yang and Ramanan 2011] progress toward this direction where instead of marginalizing over the codewords to obtain feature likelihoods, the template assignment is considered as a configuration parameter and the most consistent code-word/template with the rest of the object is selected. A similar idea is proposed in [Woodford et al. 2011].

In this thesis, we are aiming at an alternative approach where we would like to determine the code word/template that matches a feature independent of the object configuration. Although this approach may not lead to the best explaining hypothesis, we argue that determining the codebook assignment using the configuration of the rest leads to over-smooth solutions also observed in hallucinations in part-based models [Fergus et al. 2003, Felzenszwalb and Huttenlocher 2005]. Determining the assignment, or a number of likely assignments, of features to codebook entries independent of the object configuration offers many computational advantages leading to scalable detection (discussed
in Chapter 3) and at the same time enables retrieval of specific appearances and gives cues for occlusions (in Chapter 4). The down side of this is that one should go for strong feature matching and a large and discriminative vocabulary which enables more reliable matching. The issue of learning the better codebook is discussed in Chapter 3, and getting better probability estimates is discussed in Chapter 6.

As mentioned earlier, the template matching and bag-of-words models represent the two extreme paradigms in object detection. In the recent years, the gap between these two methods has been filled by many hybrid approaches from both sides. The next two sections give an overview of the advancements from either side.

### 2.3 Advances from Template Matching

The template matching approach to object detection is making strong assumptions about the rigidity of an object by only allowing small local deformations and appearance changes. Nevertheless, many object in the world are articulated and even for those with rigid structure their appearance can greatly change locally. Previously, many approaches have been proposed to deal with these issues. We give an overview of the most important ones below.

#### 2.3.1 Non-linear Template Matching

In linear matching of a single template for object detection, one measures the distance of a model to an observation by linearly summing the matching score of individual features. The features in these approaches are thus treated independent of one another and the distance of a features/pixel to the corresponding feature in is measured irrespective of other features. This independent assumption is making the detection very efficient yet it may lead to inaccuracies due to the following two reasons:

- Treating the features independently fails to model their relations in a test image due to co-occurrences of appearance or smooth deformations among them.
• By using a single template for detection, complex inter-dependence among previously observed instances (e.g. due to pose changes) cannot be retained.

The non-linear methods for object detection aim at resolving these issues by modeling the inter-dependence of training images as well the features in a single image. In particular, given the features of a test image and the features and labels of previously observed instances, they aim at implicit learning of the hidden structure to make better predictions.

Many examples of non-linear classifiers has been proposed for sliding window classification. The cascade of [Viola and Jones 2004], is an example of a non-linear matching in which the matching is done in a simple decision tree. To this end, first a linear classifier is trained by feature selection to separate the negative set from the positive set with a low false positive rate. The decision of this classifier is then refined by consequent classifiers reducing the false positives rate of the combined classifier.

Vedaldi et al. [Vedaldi and Zisserman 2009] pose the problem as a Structured Output Regression [Tsochantaridis et al. 2004] problem when in addition to the bounding box a number of nuisance parameters like its truncation and a set of simple transformations needs to be estimated.

In [Lehmann et al. 2011], the task of determining an score for a bounding box is cast as a multi-task learning problem where a number of non-linear SVMs are trained to classify subsets of bounding boxes. Starting from image classification, the space of possible bounding boxes is gradually refined by subtasks to iteratively determine the most likely bounding box in an image. The inference with these models is done similar to the Efficient Subwindow Search [Lampert et al. 2008] where the bound is replaced by the score of a task classifier.

2.3.2 Part-based Models

Inspired from the language, a very successful approach for allowing deformations in matching templates is the part-based models. First proposed in [Fischler and Elschlager 1973] and followed by pioneering works [Yuille 1991, Brunelli and Poggio 1993, Lades et al. 1993], this powerful family of models have been an active area of research with dozens of pa-
2.3. Advances from Template Matching


In the part-based models, an object is represented by a fixed number of rigid-templates referred to as parts and the deformations are handled by modeling the spatial relations among them. The task of object detection with these models is defined as finding the optimal assignment of the parts in an image by maximizing the patch likelihoods and minimizing deviations from the spatial model due to deformations.

The spatial model can be arbitrarily structured as in [Fergus et al. 2003] or tree-structured [Felzenszwalb and Huttenlocher 2005, Felzenszwalb et al. 2009] with certain conditions on the deformation models for fast inference [Felzenszwalb and Huttenlocher 2006]. The main idea behind these approaches is that the local regions of objects have rigid structures and the deformation in an object is due to varying positions of these rigid structures on the object. For example, in the pictorial structure model of [Fischler and Elschlager 1973], as visualized in Fig. 2.5 a face is modeled as a nose, two eyes, a mouth, hair and two edges. Similarly, in [Marr and Nishihara 1978, Felzenszwalb and Huttenlocher 2005] a human is modeled as a set of hierarchical parts that represents the limb structure and connections. Figure 2.5 shows two examples of these models for faces and humans.

In a part-based model, a hypothesis $h$ is described as a fixed number of auxiliary variables $\mathbf{z} = (z_1, \ldots, z_{|\mathbf{z}|})$ which encodes the configuration of an object. The variable corresponding to each part $z_k \in \mathcal{Z}$ is indexed by $k$ and is a part of state space $\mathcal{Z}$, e.g. the locations in an image. The problem of object detection with a part-based model is defined as the task of estimating the posterior distribution $p(h|I)$. Detecting objects is then done using MAP inference in Eq.(2.4). Using the Bayes’ rule we can write the posterior

$$p(h|I) = \frac{p(I|h)p(h)}{p(I)}$$  \hspace{1cm} (2.14)

where the $p(I|h)$ is the likelihood of an image given part configurations and the $p(h)$ is a prior over part configurations. Similar to template
2. Visual Object Detection: An overview

Figure 2.5: Two examples of pictorial structure models. (a) The face model of [Fischler and Elschlager 1973] with loops. (b) The tree structured pictorial structure of [Felzenszwalb and Huttenlocher 2005]. The figures are re-printed with permission.

matching, in a part-based model the appearance of the parts is considered to be independent of one another thus

\[ p(I|h) \propto \prod_k p(I|z_k) \]  \hspace{1cm} (2.15)

yet unlike the template matching approaches a non-uniform prior \( p(z) \) is assumed on the configuration of the parts. For example, in the pictorial structure model of [Felzenszwalb and Huttenlocher 2005], the prior is modeled as the tree-structured Markov random field. Given this prior the posterior becomes

\[ p(h|I) \propto \prod_k p(I|z_k) \prod_{k,k'} p(z_k, z_{k'}) \]  \hspace{1cm} (2.16)

where the \( p(z_k, z_{k'}) \) encodes the pairwise relationships of state assignments of parts \( k \) and \( k' \). The prior distribution is usually represented by a Markov random field graph with nodes being the parts and the edges their pairwise relationships. Various topologies have been considered for prior graphs of the part-based models: star-shaped [Fergus et al. 2005, Felzenszwalb et al. 2009], tree [Felzenszwalb and Huttenlocher 2005],
hierarchical tree [Bouchard and Triggs 2005, Zhu et al. 2010], hierarchical with intra-level loops [Pedersoli et al. 2011], sparse flexible models [Carneiro and Lowe 2006], loopy graphs [Crandall et al. 2005], full graph [Fergus et al. 2003]. Also, the structure of the graph and the priors are learned from the training data, either generatively [Fergus et al. 2003] or discriminatively [Felzenszwalb et al. 2009].

The inference complexity on these graphs in terms of the number of possible states for each part, $|Z|$, is increasing with the tree-width (maximum clique size) of the graphical model with the minimum of $O(|Z|^2)$ for tree structured models. Although this complexity is a general complexity with an arbitrary pair-wise relationship, for particular types of pair-wise functions efficient inference algorithms has been developed. Two notable optimizations is the polynomial algorithms for inference with submodular functions using Graph-Cuts [Boykov et al. 2001] and using generalized distance transforms [Felzenszwalb and Huttenlocher 2004, Felzenszwalb and Huttenlocher 2005, Felzenszwalb and Huttenlocher 2006] for optimizing tree-structured Markov random field graphs with Mahalanobis distance as pairwise functions.

The part-based models learn an explicit representation of the object category which has a clear advantage in detecting objects which are well represented by the priors of the model. However, this explicit representation with strong deformation priors has limitations in representing the unusual poses, abnormal objects or objects with significant occlusions. In the recent years, more sophisticated rule-based spatial models are proposed for object detection by means of Image Grammars [Zhu and Mumford 2007, Felzenszwalb and McAllester 2010]. Yet, scaling these models to the number of categories and perhaps more importantly dealing with unknown or new categories is still a challenge.

### 2.3.3 Exemplar Models

Another class of recognition methods are the exemplar models. In an exemplar model, an object hypothesis is compared to all training images by means of a similarity measure and the classification is done by using non-parametric classifiers, e.g. the k-Nearest Neighbors. Several successful examples of this model has been used previously for texture classification [Varma and Zisserman 2004], shape matching [Belongie et al. 2001,
2. Visual Object Detection: An Overview

Felzenszwalb and Schwartz 2007], object detection [Chum and Zisserman 2007, Malisiewicz et al. 2011], and non-parametric scene parsing [Liu et al. 2009].

As an alternative to the part-based models, non-parametric approaches have been previously proposed for implicit modeling of the joint variations in shape and appearance of the object categories. An advantage of these models over the parametric approaches is that they make far fewer assumptions about the structure of the objects whereas the parametric models are limited by the model structure. However, non-parametric approaches may require a lot of data to able to capture the variability in the data which is a drawback. To illustrate this issue, consider the part-based model of a pedestrians. With a part-based model a single model needs to be trained for the appearance of a part and a shape model for modeling the deformation model of the whole body. In this model, a single part can be used for detecting a standing person with the left arm up, another person with the right arm up and the persons with both arms up and both of them down. In an exemplar model, a training image should exist for all of these combinations to be detected. To resolve this issue, some approaches have tried to model random pieces of object [Divvala et al. 2010] or parts of training data representing similar poses [Bourdev and Malik 2009].

2.4 Advances from Bag of Words

As mentioned in Sec. 2.2.2, in the bag of words methods the feature likelihoods are obtained by only relying on the patch appearances and without using their locations. Further, unlike template matching these likelihoods are obtained indirectly using a visual vocabulary of patches. Although, this approach allows arbitrary deformations in the object instance, as discussed earlier, it reduces the discriminative power of this method. Recently, several approaches have addressed this issue by introducing deformation models in the context of bag-of-words models [Lazebnik et al. 2006, Leibe et al. 2008, Karlinsky et al. 2010]. In addition, many approaches have addressed the issues regarding learning the codebook of visual appearances [Leibe et al. 2008, Fulkerson et al. 2008, Boiman et al. 2008, Gall and Lempitsky 2009] and learning spatial dis-
2.4. Advances from Bag of Words

Figure 2.6: Overview of Spatial Pyramid Matching as an extension of bag-of-words models. The first level of the spatial pyramid is a bag-of-words with no location information. By dividing the hypothesis bounding box into spatial regions in subsequent levels the structure of the object is captured in more and more rigid models. The figure is reprinted from [Lazebnik et al. 2006].

Contributions for them [Gall and Lempitsky 2009, Maji and Malik 2009, Zhang and Chen 2010]. In the following, we will give an overview of deformation models proposed for object detection with a codebook which is the main drawback of the bag-of-words models.

2.4.1 Spatial Pyramid Matching

The spatial pyramid matching has been introduced in [Lazebnik et al. 2006] as an extension of bag-of-words models to capture the spatial structure of a category. In this work, a hypothesis $h$ is recursively split into smaller spatial regions $\mathcal{R}$ organized in a spatial pyramid, see Fig. 2.6, and each region is represented by a bag-of-words histogram. The first level of the spatial pyramid is essentially the bag-of-word representation of the whole image. Given all the regions $R \in \mathcal{R}$ in the spatial pyramid
and using a linear classifier, the likelihood of the whole image for the hypothesis $h$ is estimated from the likelihoods of regions $p(I|R, h)$ as

$$p(I|h) \propto \prod_{R \in R} p(I|R, h) \quad (2.17)$$

$$\propto \prod_{R \in R} \prod_{f_i \in R} p(f_i|R, h) \quad (2.18)$$

[Lazebnik et al. 2006] proposed to use a pyramid match kernel (PMK) [Grauman and Darrell 2005] for directly estimating $p(I|R, h)$ instead of a linear kernel which leads to substantial increase in performance. It has also been shown how to do efficient inference with this kernel [Maji et al. 2008] for object detection [Maji and Malik 2009].

2.4.2 Implicit Shape Model

The Implicit Shape Model (ISM) of [Leibe et al. 2004, Leibe et al. 2008] is another important extension of the bag-of-words models. In this work the location $l_i$ and scale $s_i$ of a feature $f_i$ in addition to its appearance $I_i$ is used to obtain feature likelihoods. The Eq.(2.10) for an ISM can be written as

$$p(f_i|c_h, h) = p(I_i, l_i, s_i|c_h, h)$$

$$= \sum_{\omega_j \in \Omega} p(I_i, l_i, s_i|\omega_j, c_h, h)p(\omega_j|c_h, h) \quad (2.19)$$

where for calculating $p(I_i, l_i, s_i|\omega_j, c_h, h)$ this distribution is assumed to be factored as

$$p(I_i, l_i, s_i|\omega_j, c_h, h) = p(I_i|\omega_j, c_h, h)p(l_i, s_i|\omega_j, c_h, h) \quad (2.21)$$

where, similar to a bag-of-words model in Sec 2.2.2, the first term, $p(I_i|\omega_j, c_h, h)$ is the matching likelihood of appearance $I_i$ to codeword $\omega_j$ given $c_h$ as the label of hypothesis $h$. The second term, $p(l_i, s_i|\omega_j, c_h, h)$, encodes the spatial likelihood of the location $l_i$ at scale $s_i$ given codeword $\omega_j$ and labeling $c_h$ of hypothesis $h$. 
Given this factorization and assuming independence of features, the hypothesis likelihoods $p(I|h)$ in an ISM can be written as

$$p(I|h) \propto \prod_i p(f_i|c_h, h)$$ (2.22)

$$\propto \prod_i \sum_{\omega_j \in \Omega} p(I_i|\omega_j, c_h, h)p(l_i, s_i|\omega_j, c_h, h)$$ (2.23)

To retrieve an outlier insensitive model, the summation hack [Minka 2003] is used to approximate image likelihoods $p(I|c_h, h)$ by the sum of individual feature likelihoods instead of their product

$$p(I|c_h, h) \propto \prod_i p(f_i|c_h, h) \approx \sum_i p(f_i|c_h, h)$$ (2.24)

By substituting the product in Eq.(2.22) by this sum we get the final probabilistic ISM model of [Leibe et al. 2008] as

$$p(I|c_h, h) \approx \sum_i \sum_{\omega_j \in \Omega} \underbrace{p(I_i|\omega_j, c_h, h)}_{\text{matching likelihood}} \underbrace{p(\omega_j|c_h, h)}_{\text{word prior}} \underbrace{p(l_i, s_i|\omega_j, c_h, h)}_{\text{spatial prior}}$$ (2.25)

which comprises three terms matching likelihood, word prior and spatial prior. The first two terms depend on the appearance of the local feature $I_i$ and the last term on its location. The main difference of the ISM with the bag-of-words model is in the way spatial priors for every code-word $\omega_j$ are encoded. Unlike the bag-of-words model in which only the presence of a feature in a bounding box is considered, in an ISM a much more discriminative spatial prior is learned for each word. Another difference of the ISM model is in the way a hypothesis is parametrized. Whereas in the bag-of-words and template matching approaches an object is commonly parametrized by the bounding box, in an ISM it is parametrized by a reference point $h = (l_h, s_h)$ with location $l_h$ and scale $s_h$ although other parametrizations like the bounding box and even segmentation [Marszalek and Schmid 2007] are also possible.

The spatial distribution in the Implicit Shape Model of [Leibe et al. 2008], and its variants [Gall et al. 2011], is encoded non-parametrically from the location and scale of the training patches clustered to each
visual word. For every training patch \( f_{occ} \) that belongs to the training hypothesis \( h_t \) with label \( c_t \) and is assigned to visual word \( \omega_j \), a relative position \( \hat{l}_{occ}^{j} \) and scale \( \hat{s}_{occ}^{j} \) with label \( \hat{c}_{occ}^{j} \) is calculated as

\[
\hat{l}_{occ}^{j} = \frac{l_{occ} - l_{ht}}{s_{ht}} \quad (2.26)
\]

\[
\hat{s}_{occ}^{j} = \frac{s_{occ}}{s_{ht}} \quad (2.27)
\]

\[
\hat{c}_{occ}^{j} = c_t \quad (2.28)
\]

and is stored for that visual word. The spatial prior for the patch \( f_i \) at position \( l_i \) and scale \( s_i \) given the visual word \( \omega_j \), label \( c_h \) and hypothesis \( h \) at location \( l_h \) and scale \( s_h \) are estimated from the occurrences by a balloon density estimator [Comaniciu et al. 2001]. To this end, first the relative location and scale of the patch with respect to the hypothesis is calculated as

\[
\hat{l}_{i,h} = \frac{l_i - l_h}{s_h} \quad (2.29)
\]

\[
\hat{s}_{i,h} = \frac{s_i}{s_h} \quad (2.30)
\]

Secondly, the spatial prior of \( f_i \) is estimated using Kernel density estimation from the occurrences of code word \( \omega_j \) as

\[
p(l_i, s_i|h, c_h, \omega_j) = \sum_{occ, c_{occ} = c_h} K\left(\frac{\left(\hat{l}_{i,h}, \hat{s}_{i,h}\right) - \left(\hat{l}_{occ}^{j}, \hat{s}_{occ}^{j}\right)}{b(s_h)}\right) \quad (2.31)
\]

where \( K(.) \) is a radially symmetric kernel, usually a Gaussian, with an scale adaptive band width \( b(s_h) \).

Inspired from the Hough transform [Hough 1962, Ballard 1981], and unlike the bag-of-words model of [Fei-Fei and Perona 2005] or the spatial pyramid matching [Lazebnik et al. 2006], the ISM exploits the linearity of the likelihoods for obtaining the likelihoods of all object hypothesis. For efficiency reasons, it proceeds by first looping over the image features instead of object hypotheses [Lehmann et al. 2009]. The ISM model can also be thought of as a special case of spatial pyramid matching where only one level is used in the spatial pyramid with infinitely small bin sizes and a Gaussian Match Kernel [Grauman and Darrell 2005].
Once the posterior probability over all object hypotheses is estimated, the objects are detected as local maxima of this distribution by mean-shift mode seeking [Leibe et al. 2004, Woodford et al. 2011] or iteratively found by detecting the most probable hypothesis and removing its features [Leibe et al. 2008, Barinova et al. 2010] leading to a new estimation of the posterior distribution. As explained in [Barinova et al. 2010], this feature removal technique is in fact enforcing a Minimal Description Length (MDL or otherwise called Occam’s Razor) prior on the assignment of the features to hypotheses where each feature is only allowed to be assigned to a single hypothesis.
Performance of object detection methods has improved substantially in the recent years [Everingham et al. 2010]. As the number of detectable object classes increases, so does the need for scalable methods for detection. However, scaling multi-class object detection remains a challenging task. The main difficulty is that object categories can have a very large intra-class variability in pose, appearance, and shape. In a multi-class setup, this poses a challenge since the detection method should be discriminant to inter-class and clutter variations and invariant to intra-class variations.

As explained in Chapter 2, in order to deal with these issues, several successful methods represent an object hypothesis with a set of features/parts and learn a shape model to determine the class label of this hypothesis. In this scenario, every part has an appearance and a location. The location is generally defined as the relative position and scale of a feature with respect to one or several reference points on the object, e.g., center of mass. In a multi-class setup, the features (or the combination of them) need to be able to discriminate classes from one another. Yet, to enable scalable detection, previous approaches have proposed to share both the appearance and location of parts among classes; thus reducing the amortized detection cost per class.

The multi-class detection approaches [Torralba et al. 2007, Fidler and Leonardis 2007] have focused on sharing the appearance of the features for better generalization while discriminating objects from the clutter. In [Torralba et al. 2007], a set of features at fixed locations are selected using a boosting procedure to discriminate objects from clutter and classes from each other. Without enforcing sharing, this procedure
Figure 3.1: Features of different object classes can share appearance although they do not necessarily also share their location. For instance, the legs of a person and a horse share both appearance (bounding boxes) and location (arrows) whereas the wheels of a bus and car are similar in appearance but not in location (red/blue arrows).

overfits to the training data and selects a set of very specific discriminative features with large spatial support with poor generalization. Hence, the authors have proposed a solution to this problem by enforcing sharing of features across classes. Another advantage of sharing is the reduced cost of feature extraction. The selected features, however, become very generic since the location is fixed. These generic features are generalizing well but they are very weakly discriminating between classes and therefore detection with them requires evaluating a model for every class that scales linearly in the number of classes [Stefan et al. 2009]. Similarly in [Fidler and Leonardis 2007], a set of shape features are learned by combining generic simple edge features. However, since the location is fixed in the constellation model, the model of every class needs to be evaluated [Fidler et al. 2010].

In several other approaches [Griffin and Perona 2008, Marszalek and Schmid 2008, Fergus et al. 2010, Fidler et al. 2010, Bengio et al. 2010], the object classes are organized in a hierarchy for the sake of scalable detection. In these method, a classifier is trained at each level of the hierarchy and used to classify a hypothesis as belonging to one group of classes in one branch. For example, in [Griffin and Perona 2008] a binary tree is used as the hierarchy. Provided that this is a balanced tree, a hypothesis needs to be evaluated only logarithmic times in the number of
Figure 3.2: Complexity in appearance of a local patch, and thus its discriminative power, can be altered by varying its relative size/resolution to the object. Using small regions of an image only contain basic edge information and are not alone informative for either a possible object’s position (localization) or its class label (classification). Yet, by increasing the size of a feature its discriminative power rapidly increases leading to more precise predictions even specific to an instance or situation.

classes in contrast to the linear complexity of the one-vs-the-rest detectors. However, since the classifiers at the top levels need to discriminate among many classes, these methods usually lead to significant accuracy loss. Learning the optimal hierarchies is also an NP-complete problem, although there has been several attempts to learn them optimally, for example refer to [Bengio et al. 2010]. Since the nodes of the hierarchy need to make hard decision, the errors at the higher levels of the hierarchy propagate through the hierarchy and cannot be corrected at later nodes. In addition, due to these hard decisions, the confidence of the other classes are not retrieved in these methods.

In this chapter, we propose a voting based approach for scalable multiclass detection of object categories. The core idea of our approach is to learn discriminative mid-level features in appearance and location. Using a cost-analysis of detection, we argue that ideally, the appearance of a local feature should give as much information as possible about the class label and location of probable object hypotheses. In other words, to facilitate scalable detection, the individual features need to be specific and have a sparse class and spatial distribu-
3. Scalable Multiclass Detection

However, as shown in the previous work [Torralba et al. 2007, Varma and Ray 2007], there is a trade-off between the discriminative power of a feature and its generalization capability. In particular, features with larger spatial support and resolution tend to be more discriminative and less generalizable whereas smaller spatial support and lower resolution results in more generic and thus less discriminative features. Figure 3.2, illustrates the effect of the feature size on the complexity of local features.

In order to reliably detect object categories and at the same ensure generalization to new instance, we propose to control the complexity of the features by their size and learn mid-level features which are both discriminative and able to generalize. This is achieved by treating the location and appearance of features differently. Our approach is also motivated by psychological evidence suggesting that the localization and classification of objects follow different pathways in the human visual system [McCloskey and Palmer 1996, Ungerleider and Haxby 1994]. Such separation implicates also possible computational advantages as discussed in [Rueckl et al. 1989].

In our approach, we focus on the features of intermediate complexity which are optimal for classification [Ullman et al. 2002]. In particular, the appearance of our features is shared across some categories providing good generalization, yet remain discriminative for a subset of classes. However, when the appearance is combined with location, the features become discriminative for the individual classes. For instance in Fig. 3.1, the wheel discriminates the bus from the horse and the person, but not from the car. However, the location of the wheel in the bus image gives another detection of the object than the wheel of the car. Hence, we first classify features and obtain a set of likely categories and then do localization to gather evidence for the position of the most likely object class.

For building a shared vocabulary, we extend the single-class approach of [Gall and Lempitsky 2009] and introduce a novel optimality criteria to achieve the right balance between feature sharing and discrimination in the context of multi-class detection. The proposed multi-class detector scales better than a combination of single-class detectors and, most importantly, is similar in detection accuracy. The detection is based on the implicit shape model [Leibe et al. 2008] where codebook entries vote
for the object position and class. Due to the sharing of features, the number of votes that need to be cast for detection also increases only sub-linearly with respect to the number of classes. Furthermore, we build a taxonomy of object classes from the sharing distribution among classes.

We show that the derived taxonomies have a semantic interpretation and that the taxonomies increase scalability by reducing the number of votes. Since the detection might result in overlapping bounding boxes that are ambiguous in class label, we perform an additional classification after detection. The detection reduces not only the number of potential bounding boxes to a very small number, it also provides a class label such that each bounding box is only classified with respect to a single object class and not all classes.

The rest of this chapter is organized as follows. After an overview of the related work for scalable detection, we give a detailed cost analysis of object detection with a shared vocabulary. For this purpose, we divide the cost of object detection into three components matching cost, voting cost, and localization cost and analyze the scaling behavior of each of which in the number of classes. Next, the multi-class detection with the Hough Forests [Gall et al. 2011] as an instance of the Implicit Shape Model is described. Using the cost analysis, we then determine the desired criteria of a shared vocabulary and show how to learn the vocabulary by optimizing these criteria. Finally, in our experiments we compare the performance and efficiency of the resulting multi-class detector to that of a battery of one-vs-the-rest detectors.

### 3.1 Related Work

Several approaches have addressed the problem of feature sharing in the context of multi-view or multi-class object detection [Torralba et al. 2007, Fidler and Leonardis 2007, Razavi et al. 2010, Shotton et al. 2008a]. Following a sliding window approach for detection, [Torralba et al. 2007] proposes a boosting procedure to explicitly enforce sharing and shows that the number of features grows sub-linearly with the number of classes. The classification in [Torralba et al. 2007] is done in a one-vs-the-rest approach and scales linearly with the number of classes,
however. [Stefan et al. 2009] has reduced joint-boost recognition at classification time to nearest neighbor search in a vector space to scale the joint boosting for large multi-class problems. However, this work is limited to the joint-boosting and requires many similar classes.

Similar to [Krempp et al. 2002], [Fidler and Leonardis 2007] introduced a method for learning a scalable hierarchy of parts by using the statistical co-occurrence of generic features. For shape-based object detection, a constellation model of these parts is independently evaluated for each class. This work has been recently extended [Fidler et al. 2010] for speeding up multi-class detection by introducing a coarse-to-fine representation of contour features and constellation models. In [Torralba et al. 2007, Fidler et al. 2010], it is assumed that constellations of similar classes are similar and can therefore be grouped together. However, this assumption is too restrictive as it forces the features in the constellation to simultaneously share appearance and location. As a result, this method yields shallow hierarchies with many disjoint groups of classes and requires again coarsening of features which are once joined in training to increase discriminative power.

Hierarchical taxonomies of object categories have been used previously for scalable image classification as well [Griffin and Perona 2008, Marszalek and Schmid 2008, Fergus et al. 2010]. In [Griffin and Perona 2008], a taxonomy is built by clustering the confusion matrix of one-vs-the-rest detectors. At each node of the hierarchy, a classifier is trained and combined to a multi-class classifier. The approach has been extended in [Marszalek and Schmid 2008] by allowing overlapping labels in disjoint branches of the hierarchy. The degree of overlap gives a trade-off between accuracy and efficiency. Object hierarchies have been used in [Fergus et al. 2010] to transfer the label into an unseen class by sharing labels between semantically similar classes. Although these works are using hierarchies and are therefore related, they are not applicable to object detection.

Building a codebook of parts and learning appropriate weights for them is also addressed in the literature. [Leibe et al. 2008, Fidler and Leonardis 2007] follow a generative approach for clustering the patches whereas [Shotton et al. 2008a, Maji and Malik 2009, Zhang and Chen 2010, Gall and Lempitsky 2009] pursue a discriminative one. [Maji and Malik 2009, Zhang and Chen 2010] build the codebook in a generative way but
learn appropriate weights for them discriminatively using a max-margin framework. [Shotton et al. 2008a, Gall and Lempitsky 2009] learn a direct mapping from the patch appearances to weights in a random decision forests framework. In [Shotton et al. 2008a], only class labels are used for discriminative training where both class labels and spatial location of features are used in [Gall and Lempitsky 2009]. In [Ommer and Malik 2009], scale-invariant features of training data are stored as the codebook without any quantization [Boiman et al. 2008] and used to cast voting lines for detection. Since our features are not scale-invariant, [Ommer and Malik 2009] does not directly apply to our approach.

Recently, several approaches have considered generic object detection [Alexe et al. 2010, Endres and Hoiem 2010, Carreira and Sminchisescu 2010]. These approaches are related to this work as they also separate the localization and classification. In contrast to our work, these approaches do not make use of class information during detection. This implies that detected bounding boxes need to be evaluated for all classes yielding a linear complexity in the number of classes. In this work, we use the label information encoded in the detection for scalable multi-class detection where we do not need to run the final expensive classifier for all classes.

### 3.2 Detection with an ISM

As described in Chapter 2, in an ISM a number of features $f_i = (I_i, l_i, s_i)$ are extracted from an image $I$ prior to detection. Each feature is described with an appearance model $I_i$ and the location $l_i$ and scale $s_i$ of its occurrence in $I$. During detection, each feature is matched to a visual vocabulary to determine the matching likelihood $p(I_i|\omega, c_h, h)$ and retrieve the word prior $p(\omega|c_h, h)$ and spatial prior $p(l_i, s_i|\omega, c_h, h)$ for every label $c_h$ of an object hypothesis $h$. As in other Hough transform-based methods, the matching likelihoods are obtained independent of the hypothesis $h$, $p(I_i|\omega, c_h) = p(I_i|\omega, c_h, h)$. Further, the word priors only depend on the hypothesis labels $p(\omega|c_h) = p(\omega|c_h, h)$. The spatial priors are obtained by non-parametric density estimation from the occurrences of training features at each codebook.

As derived in Eq. (2.25), the image likelihoods given a hypothesis and label, $p(I|c_h, h)$, are obtained by a linear sum over the features and
the visual words. By substituting Eq.(2.31) in (2.25), we can write the likelihoods as

\[
p(I|c_h, h) \approx \sum_i \sum_{\omega_j \in \Omega} p(I_i|\omega_j)p(\omega_j|c_h) \sum_{occ, c_{occ}=c_h} K \left( \frac{(\hat{l}_{i, h}, \hat{s}_{i, h}) - (\hat{l}_{occ}^j, \hat{s}_{occ}^j)}{b(s_h)} \right)
\] (3.1)

This way, the image likelihoods for all object hypothesis can be obtained by looping over all object hypotheses, labels and features. Since in an ISM the matching likelihoods and the word priors are obtained independent of the object hypothesis, this can be performed using Hough transform. To this end, every feature is first matched to the visual vocabulary and its votes for all object hypothesis are cast in a voting space denoted by \( V \in \mathbb{R}^{H \times C} \). After the votes of all features are accumulated in \( V \), each element of \( V \) stores the image likelihoods given a hypothesis and label.

The spatial distribution \( p(l_i, s_i|h, c_h, \omega_j) \) in an ISM is encoded non-parametrically using density estimation from the training occurrences in the word \( \omega_j \). In particular, the kernel density estimation is carried out by convolving a radially symmetric Kernel (usually a Gaussian) which is a linear operator. The linear nature of the convolution allows the votes of individual training occurrences to be cast as Kronecker delta’s and the density estimation to be carried out collectively for all votes by a single convolution. To this end, for each occurrence \( (\hat{l}_{occ}^j, \hat{s}_{occ}^j, c_{occ}^j) \in \omega_j \) and feature \( f_i = (I_i, l_i, s_i) \), the best explaining hypothesis \( \hat{h} \) at which the kernel is centered is selected as

\[
\hat{h} = \arg \min_{h \in H} \left( (\hat{l}_{i, h}, \hat{s}_{i, h}) - (\hat{l}_{occ}^j, \hat{s}_{occ}^j) \right)
\] (3.2)

and the weighted vote \( \zeta \cdot p(I_i|\omega_j) \cdot p(\omega_j|c_{occ}) \cdot \delta_{\hat{h}, c_{occ}} \) is cast to the voting space. \( \zeta \) is a normalization constant of \( p(l_i, s_i|h, \omega_j, c_{occ}^j) \) and is calculated as

\[
\zeta \leftarrow \frac{1}{\sum_{occ'} \mathbb{I}(c_{occ}', c_{occ}^j)}
\] (3.3)

where \( \mathbb{I}(\cdot, \cdot) \) is 1 if \( c_{occ}' = c_{occ}^j \) and 0 otherwise. Algorithm 1 is summarizing the voting procedure in an ISM. In the next section, we will analyze the computational cost of multi-class object detection with an Implicit Shape Model [Leibe et al. 2008].
Algorithm 1 Voting procedure in an ISM [Leibe et al. 2008]

\[ \mathcal{F} \leftarrow \text{extract features from } I \]
\[ \mathcal{V} \leftarrow 0 \]
\[ K \leftarrow \text{initialize kernel} \]

//loop over all features
\[ \text{for } i = 1 \rightarrow |\mathcal{F}| \text{ do} \]
\[ \text{for } j = 1 \rightarrow |\Omega| \text{ do} \]
\[ p(I_i|\omega_j) \leftarrow \text{match } f_i \text{ to } \omega_j \]
\[ \text{for all } (\hat{l}_{occ}, \hat{s}_{occ}, c_{occ}) \in \omega_j \text{ do} \]
\[ \hat{h} = \arg\min_{h \in \mathcal{H}} \left( (\hat{l}_{h}, \hat{s}_{h}) - (\hat{l}_{occ}, \hat{s}_{occ}) \right) \]
\[ \zeta \leftarrow \frac{1}{\sum_{occ} I(l_{occ} = c_{occ})} \] // norm. constant of \( p(l_i, s_i | \hat{h}, \omega_j, c_{occ}) \)
\[ \mathcal{V} \leftarrow \mathcal{V} + \zeta \cdot p(I_i|\omega_j) \cdot p(\omega_j|c_{occ}) \delta_{\hat{h}, c_{occ}} \]
\[ \text{end for} \]
\[ \text{end for} \]
\[ \mathcal{V} \leftarrow \mathcal{V} \ast K \] // convolving with the kernel

### 3.3 Complexity Analysis

As described in Alg. 1, estimating the image likelihoods \( p(I|h, c_h) \) for all hypotheses and labels in an Implicit Shape Model comprises four steps. First, a set of features \( f_i \in \mathcal{F} \) are extracted from the image \( I \). Second, every feature is matched to the visual vocabulary, incurring the matching cost, to obtain codewords with non-zero matching probability \( p(I_i|\omega_j) \) and retrieve a number of occurrences stored at each codeword. Third, the retrieved occurrences for every matching of feature \( f_i \) to a codeword \( \omega_j \) with positive matching probability \( p(I_i|\omega_j) \) are cast in the voting space incurring a voting cost. The cast votes are then used for kernel density estimation which has a linear complexity in the number of votes. In the following, we analyze the complexity of the detection with an ISM denoted by \( L \), in various scenarios.

For discussing the complexity in terms of the number of classes \( |C| \), we assume that the number of features per test image is scaling linearly with the number of pixels in an image \( O(|I|) \) and the cost of extracting a single feature is constant. Furthermore, we assume that the number of
training images per class, \( N \), is equal for all classes and the number of features per training example is constant. The total number of features occurrences, denoted as \( |F^{\text{train}}| \), stored in the vocabulary \( \Omega \) is then assumed to increase both with the number of classes and training images, \( i.e. \ O(|C| \times |N|) \). Further, the number of occurrences stored in each codeword is assumed to be equal at \( \frac{|F^{\text{train}}|}{|\Omega|} \).

The first scenario is that of soft matching of every feature with every codeword (see Sec. 2.2.2). In this case, since the soft matching is used, we can assume that the number of codewords with non-zero matching probabilities \( p(I_i|\omega_j) \) is increasing with the size of the visual vocabulary, \( i.e. \ O(|\Omega|) \). The detection complexity for the soft-matching can be written as

\[
\mathcal{L} = O \left( \frac{|I| \times |\Omega| + |I| \times |\Omega| \times \frac{|F^{\text{train}}|}{|\Omega|}}{\frac{|F^{\text{train}}|}{|\Omega|}} \right) = O \left( |I| \times |C| \times |N| \right) \tag{3.4}
\]

which is dominated by the voting cost and increases with the number of classes \( O(C) \). Given this analysis it is clear that, in order to scale well with the number of classes, the number of matching entries with positive \( p(I_i|\omega_j) \) should be constant or increase sub-linearly with \( |\Omega| \). In the following, we assume a constant number of matching entries e.g. as done in the hard-matching in Eq.(2.11).

Previous approaches have proposed both linear \( O(|\Omega|) \) [Leibe et al. 2008, Torralba et al. 2007] and logarithmic \( O(\log|\Omega|) \) [Boiman et al. 2008, Gall and Lempitsky 2009] matching costs in the size of the codebook \( |\Omega| \). We give an analysis of the detection complexity with both linear and sub-linear matching costs in the following.

When the cost of matching is scaling linearly with the size of the codebook \( O(|\Omega|) \), the cost of detection can be written as

\[
\mathcal{L} = O \left( \frac{|I| \times |\Omega| + |I| \times \frac{|F^{\text{train}}|}{|\Omega|}}{\frac{|F^{\text{train}}|}{|\Omega|}} \right). \tag{3.5}
\]
3.3. Complexity Analysis

As can be seen, if the size of the vocabulary $|\Omega|$ is increasing logarithmically with the number of categories $O(\log|C|)$ as proposed in [Torralba et al. 2007, Fidler and Leonardis 2007], the detection cost will be dominated by the voting cost and will scale linearly $O(|C|)$. This logarithmic increase denotes a strong sharing among features of different categories that adversely affects the scalability of the detector. Alternatively, by having class-specific codewords, if $|\Omega|$ is increasing linearly with the number of classes $O(|C|)$, the detection cost will also scale with $O(|C|)$ as it will be dominated by the matching cost. The optimal complexity in this scenario is achieved by having the size of the vocabulary scaling with $\sqrt{|C|}$ where the detection complexity will increase at the same rate. In order to scale the detection, one can either decrease the complexity of matching, use less training images, or reduce the number of voting elements which currently is $O(|C| \times |N|)$. Neither reducing the number of training images nor the number of votes is favorable as it impairs appearance and geometry models of a category leading to a reduced detection accuracy.

Several approaches [Gall and Lempitsky 2009, Boiman et al. 2008] proposed matching schemes with logarithmic cost in the size of the vocabulary $O(\log(|\Omega|))$. In this scenario, the complexity of detection is

$$\mathcal{L} = O \left( |I| \times \log |\Omega| + |I| \times \frac{|F_{\text{train}}|}{|\Omega|} \right). \quad (3.6)$$

where the best scaling behavior is achieved when the size of the codebook is increasing with the number of classes which results in a logarithmic detection cost in the number of classes $O(\log|C|)$. However, our experiments confirm the observation of [Torralba et al. 2007, Fidler and Leonardis 2007, Ullman et al. 2002] that in order to afford generalization and handle intra-class variability, the features cannot be very specific to a certain class as implied by $|\Omega| = O(|C|)$. Hence, in order to ensure generalization, the size of the codebook shall scale sub-linearly in $|C|$ yet not logarithmically $|\Omega| = O(\log |C|)$ since the detection cost becomes

$$\mathcal{L} = O \left( |I| \times \log \log |C| + |I| \times \frac{|N| \times |C|}{\log |C|} \right) = O (|C|) \quad (3.7)$$

that increases linearly with the number of categories.
Using the above analysis, we conclude that in order to be able to scale the detection cost with the number of classes one needs to find a trade-off between feature sharing and class-specificity. Further, this trade-off depends on the matching scheme. Although by using logarithmic matching costs, the feature should rather be as specific as possible, i.e. as long as they generalize well, the linear matching implies finding a fragile balance between sharing and discrimination. Yet, it is worthwhile to note that the logarithmic matching of a feature, e.g. using approximate nearest neighbor search, is not error free and is not as accurate as linear matching.

3.4 Proposed Multi-Class Detector

3.4.1 Training the Shared Codebook

To train the visual vocabulary \( \Omega \) with entries \( \{ \omega_1 \ldots \omega_{|\Omega|} \} \), first a set of features \( f_i^t \in \mathcal{F}_{\text{train}} \) are extracted from the training images \( t \in \mathcal{T} \) and to each of which a hypothesis and a label are assigned. The training features are sampled from a set of bounding-box annotated positive images and a set of background images. Similar to the Generalized Hough transform [Ballard 1981], each hypothesis \( h \) is parametrized by the location \( l_h \) and scale \( s_h \) of its center (initially set as the center of the bounding box) and assigned a label \( c_h \). Let us assume that every feature \( f_i^t \) at location \( (l_i^t, s_i^t) \) and appearance \( I_i^t \) is assigned to the hypothesis \( \hat{h} \) with label \( c_i^t = c_h \) if it is inside the bounding box of hypothesis \( h \) at the same scale. We consider \( h \) as its best explaining hypothesis for voting as calculated in Eq. (3.2). In addition, for every feature we also record additional training data information \( \theta_i^t \), e.g. the identity of the training image it is sampled from as a pointer to the additional properties that might be available for that training image.

The task of training a visual vocabulary is to cluster the training features into (disjoint/overlapping) clusters such as the features with similar properties (e.g. appearance, color, location, etc.) are clustered together. For training the visual vocabulary, we use a multi-class extension of the class-specific Hough forests [Gall and Lempitsky 2009]. Although we are
Figure 3.3: Prior to training a visual vocabulary with the multi-class Hough Forests, a set of background and foreground training images are collected. The foreground images are also assigned a label determining their class/viewpoint (coded by color). For training every tree, a set of features are extracted from the training images \( t \in T \) and are recursively split such as to optimize the discriminative power of the resulting nodes. The leaves of the forest are treated as codewords \( \omega_j \) and store the occurrences of the training data and their offset \( l_i^t \), class label \( c_i^t \) and auxiliary variables \( \theta_i^t \). The visualized occurrences for every leaf node are the actual patches in the trained vocabulary.

Not limited to this choice, this framework provides us with the flexibility to analyze several issues regarding sharing, classifier complexity and feature size in a multi-class setup. In addition, this method allows a dense sampling of the image which has shown to outperform to sparse samplings.

Hough Forests are random forests [Breiman 2001] which are trained discriminatively to boost the voting performance. During training, using an optimality criteria, a binary test is assigned recursively to each node of the trees that splits the training patches into two subsets. Splitting is continued until the maximum depth is reached or the number of remaining patches in a node is lower than a predefined threshold. Unlike
3. Scalable Multiclass Detection

**Figure 3.4:** Patches clustered in two sample leaves of our multi-class detector trained for VOC’06 dataset. Although the appearance of our middle complexity patches is shared among some classes, it is yet discriminative to other classes. For instance, when a patch is assigned to leaf (a), it is difficult to determine if it belongs to a dog or a person but we can say that it is not coming from a car or a motorbike. This property of middle-complexity features enable building of part-based category hierarchies introduced in Sec. 3.4.3.

...training the decision trees, no pruning is performed after the training. The leaves of the decision trees, i.e. codewords \( \omega_j \), store the arrived training patches \( F_{\text{train}}^j \).

Class-specific Hough forests discriminatively learn the codebook \( \Omega \) from the training data \( F_{\text{train}} \). To this end, starting from the root with all features, a binary test is assigned recursively to each node \( n \) of the trees that optimally splits features of that node \( F_{\text{train}}^n \) into two sets \( F_{\text{train}}^{n_{\text{left}}} \) and \( F_{\text{train}}^{n_{\text{right}}} \) according to certain criteria. The optimal binary tests are selected as to minimize either location or class uncertainties. As explained earlier, we can consider each node as a codeword \( \omega_n \) at which the probabilities \( p(\omega_n|c_h) \) and \( p(l, s|\omega_n, h, c_h) \) for each hypothesis \( h \) and its labeling \( c_h \) can be calculated non-parametrically using features \( F_{\text{train}}^n \). We can define class and location uncertainties for multi-class
as a natural extension of the criteria introduced in [Gall and Lempitsky 2009] for single class as

\[ U^c(n) = \left| \mathcal{F}_n^{\text{train}} \right| \sum_{c \in \mathcal{C}} -p(c|\omega_n) \cdot \log p(c|\omega_n) \]  

(3.8)

\[ U^l(n) = \sum_{f_i^t \in \mathcal{F}_n^{\text{train}}} \left\| \hat{l}_i^t - \bar{l}_{c_i^t} \right\|^2_2 \]  

(3.9)

where \( p(c|\omega_j) \) is the density of class \( c \) given \( \omega_n \) and \( \bar{l} = \frac{1}{|\mathcal{F}_n^{\text{train}}|} \sum_i l_i^t \) is the mean vector of locations.

While (3.8) enforces the features to be discriminative between objects and background, (3.9) favors features sharing their location. Optimizing the class and location uncertainties indeed prefers features at each leaf to satisfy both properties. It is important to note, however, that the second criterion (3.9) is not forcing similar locations as in [Torralba et al. 2007] as it only tries to minimize location uncertainty as much as possible. Nevertheless, this requirement can be restrictive as mentioned earlier (see Fig. 3.5). Hence, we relax the second criteria to be class dependent letting features to have different locations for different classes:

\[ U^l(n) = \sum_{f_i^t \in \mathcal{F}_n^{\text{train}}} \left\| \hat{l}_i^t - \bar{l}_{c_i^t} \right\|^2_2 \]  

(3.10)

where \( \bar{l}_c = \sum_{i,c_i^t = c} l_i^t \).

The class uncertainty measure in (3.8) treats the background class like any other class and does not ensure well separation of the background from other classes (see Fig. 3.5(b)). We thus propose a special treatment of the background class by having a cost function that measures the information gain for separating background from any other class given by

\[ U^{bg}(n) = \left| \mathcal{F}_n^{\text{train}} \right| \sum_{c \in \{c_o, c_{bg}\}} -p(c|\omega_n) \cdot \log p(c|\omega_n) \]  

(3.11)

where \( p(c_o|\omega_n) = \sum_{c \in \mathcal{C}, c \neq c_{bg}} p(c|\omega_n) \)  

(3.12)
Figure 3.5: In this figure, the effect of training with various optimization criteria is illustrated. (a) When training according to (3.8) and (3.9). (b) By relaxing the location with (3.10), more distinction is appearing between dissimilar classes but the separation of background class is problematic. (c) If we only separate the foreground and background using (3.11), the background gets well separated but the classes remain mixed. (d) This is the sharing matrix obtained from (3.14) for which the classes and background are both well separated. As can be seen, the classes are separated similar to (b) and the foreground and background similar to (c). (best viewed in color)
and another cost function measuring the information gain for separating object classes from each other

$$U^o(F) = |F| \sum_{c \in C, c \neq bg} -p(c|\omega_n) \cdot \log p(c|\omega_n). \quad (3.13)$$

These two cost functions are then mixed to form the final cost function

$$U^c(F) = U^o(F) + \lambda U^{bg}(F) \quad (3.14)$$

where the mixing coefficient $\lambda$ can be varied to tune the amount of class versus background discrimination. Figure 3.5 shows sharing matrices of vocabularies trained with varying this parameter.

### 3.4.2 Measuring Sharing

For each codeword $\omega_j$, let us assume that in addition to $p(c|\omega_j)$, we have precalculated the offsets $l_{occ}^j$, their labelings $c_{occ}^j$, and their associated weights for voting $w_{occ}^j$. For every codebook, we can obtain a sharing matrix $S : |C| \times |C|$, which indicates the degree of sharing between different classes. The sharing matrix can be calculated by considering only appearance of features, or both appearance and location. An element $S^a(c, c')$ of a sharing matrix for appearance is calculated by

$$S^a(c', c'') = \frac{1}{\zeta} \sum_j p(c'|\omega_j) \sum_{c_{occ} = c''} w_{occ}^j. \quad (3.15)$$

where $\zeta = \sum_{c \in C} S(c', c)$. Similarly, we can calculate a sharing matrix of sharing both appearance and location by

$$S^{al}(c', c'') = \frac{1}{\zeta} \sum_j p(c'|\omega_j) \sum_{c_{occ} = c'} w_{occ}^j \sum_{c_{occ'} = c''} w_{occ'}^j K(\hat{l}_{occ'}^j, \hat{l}_{occ}^j). \quad (3.16)$$

where $\zeta$ is again a normalization factor and $K(\hat{l}_{occ'}^j, \hat{l}_{occ}^j)$ is a radially symmetric kernel which is set to a disc with the radius of 10 pixels in our implementation. Hence, two features of different classes are considered shared only when they are assigned to the same codewords and their location is within a certain distance of another (10 pixels in our implementation).
3.4.3 Building the Taxonomy

We automatically build a taxonomy of classes by clustering the appearance sharing matrix defined in (3.15). The appearance sharing matrix is asymmetric and its elements are affinities. For clustering, we transform it into a symmetric dissimilarity matrix $D$ by

$$D = 1 - \frac{1}{2}(S + S^T).$$  \hfill (3.17)

The taxonomy $T$ is obtained by clustering $D$ using the complete-linkage agglomerative clustering. Sharing matrices using (3.15) and (3.16) and their corresponding taxonomies are shown in Fig. 3.6.

In contrast to [Griffin and Perona 2008], the hierarchy is derived from the sharing matrix and not from the confusion matrix. This is more efficient since the similarity can be directly computed from the trained codebook without an additional expensive validation procedure that is needed for the confusion matrix. Furthermore, taking the confusion matrix means that the location is also used whereas we would like to have sharing only based on the appearance. Although it is our objective to automatically obtain a taxonomy from the sharing matrix, it is still possible to use WordNet [Fellbaum 1998]-style hand-crafted semantic taxonomies. In principle, semantic taxonomies enable sharing labels of the learned categories to unseen categories [Fergus et al. 2010]. This transfer is beyond the scope of this dissertation and we leave it for future work. However, as we will see in the experimental section, the hierarchies that we obtain by clustering the sharing matrix are indeed semantically meaningful; therefore, we do not expect to have a performance drop by using semantic hierarchies.

3.4.4 Detection with the Taxonomy

As described in Sec. 3.3, the cost of detection with the described detector scales with the number of cast votes to the voting space. In order to reduce this number, we use a taxonomy that has been automatically
Figure 3.6: Sharing matrices for VOC’06 dataset using our proposed multi-class criteria in Eqs. (3.10) and (3.14) for training. (a) Sharing matrix of feature appearances using (3.15) and its corresponding taxonomy which is automatically obtained by clustering the sharing matrix. When calculating sharing of both appearance and location with (3.16), the sharing matrix in (b) is obtained. When discriminativity in appearance is combined with location, our multi-class training leads to a very good separation of classes. (c) Speed-up achieved using the taxonomy in (a). The more discriminative features lead to more speed-up. (best viewed in color)
obtained from the sharing matrix; see Fig. 3.6(a) for an example taxonomy. The object classes are the leaves of this taxonomy and every internal node \( t \) has a subset of classes \( C_t \). We use the taxonomy to efficiently retain a small subset of similar categories for each feature and only vote with those classes. For this purpose, we only look at the appearance of the features. At training time, for every node \( t \) and each codeword \( \omega_j \), we pre-calculate a weight \( \psi^t_j = \sum_{c \in C_t} p(c|\omega_j) \) and normalize it by the weight at the root and store it in the codebook. At the detection time, a dense set of features are extracted from an image and are assigned to a codebook entry. For every feature \( f \), the taxonomy tree is traversed using breath first search and all the leaves with weights greater than a threshold are retained and the voting is done using those leaves only. Since the weight at each node is the sum of the weights at its children, retaining the leaves with weights bigger than the threshold can be done efficiently and in sub-linear time. The threshold in our work is node specific and is set to \( \alpha \times \frac{\text{linkage}}{|C|} \) where \( \text{linkage} \) is the linkage cost of the parent node (set to one at the root) and \( \alpha \) is set to 1.5 in our implementation. In order to increase robustness, we obtain the weights at each node by averaging weights of all the features in the neighborhood of \( f \). In our experiments, we used a neighborhood of 8×8 pixels.

### 3.4.5 Verification

The above mentioned detection algorithm has a very good recall, but this usually comes at the cost of low precision. The main underlying reason for this is the independent assumption which is too crude for classification but vital for fast detection. In addition, the detection system looks only at the features which are well localizable. For example, seeing an elephant’s skin tells little about the elephant’s center, however, it is useful for classification. To this end, we re-score the hypothesis obtained from the previous step using a more sophisticated classifier (we used [Felzenszwalb et al. 2009] for which the source code is kindly provided by the authors). Unlike other methods, e.g., [Maji and Malik 2009], that exhaustively search for a set of bounding boxes in the neighborhood of a detection, the re-scoring in our system is performed only for a single bounding box and a single class, which makes it scal-
3.5 Experiments

For our multi-class detection experiments, we use the popular PASCAL VOC 2006 and 2007 datasets [Everingham et al. 2010]. We also consider two baselines for the performance analysis. The first baseline is detection with one-vs-the-rest Hough Forests using joint-voting and then followed by the verification step using [Felzenszwalb et al. 2009]. The second baseline is the output of the classifiers [Felzenszwalb et al. 2009] using sliding window for detection.

The multi-class detector presented in Sec. 3.4 is single-scale. The detection on the multi-scale VOC datasets is carried out at multiple scales by rescaling images. We have used the scales: $\sqrt{2}^i$ where $i \in \{-3, -2, \ldots, 8\}$. The range of scales appearing in VOC datasets are different for different classes. For efficiency reasons, a range of scales from the a-forementioned range is selected for each class using the validation set.

**Figure 3.7:** This figure gives an overview of the multi-class detection with a verification classifier.
3. Scalable Multiclass Detection

Figure 3.8: The sharing matrix of appearance and its automatically built taxonomy for the VOC’07 dataset. (best viewed in color)

3.5.1 Training

Prior to training of the forest, the training objects are cropped from the bounding box annotated images and rescaled to have approximately the same size. A small region around the bounding boxes with 10% of the original bounding box width is added to the cropped image. The rescaling is carried out to have the maximum of width and height not less than 100 pixels and the minimum of them not smaller than 50 pixels. The coordinates at the center of the bounding box of each object is considered as its center. For training each tree, 200 objects per class are randomly selected (when available) where 250 patches are extracted from each. 250 patches are extracted from non-object regions of 200 randomly selected images and used as background patches. The patch size is set to $16 \times 16$ pixels which can be considered of intermediate complexity (see Fig. 3.4). Same image features as [Gall et al. 2011] are used. A class label is assigned to each feature and the additional view annotations are ignored. Each tree is trained until the depth 20 for VOC’06 and 25 for VOC’07 is reached, or until less than 10 patches for one class are left.
### 3.5. Experiments

#### Methods

| Methods              | $|\Omega|$ | Match. in $|\Omega|$ | No. of Votes | Training |
|----------------------|----------|---------------------|--------------|----------|
| ISM                  | linear   | linear              | linear       | quadratic|
| Hierarchy of parts   | sublinear| linear              | linear       | linear   |
| Coarse-to-fine       | linear   | sublinear           | linear       | linear   |
| HoughForest          | linear   | sublinear           | sublinear    | linear   |
| This paper           | sublinear| sublinear           | sublinear    | linear   |

**Feature-based methods**

<table>
<thead>
<tr>
<th>Sliding Window Methods</th>
<th>Joint-boost</th>
<th>DPM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sublinear</td>
<td>constant</td>
</tr>
<tr>
<td></td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td></td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td></td>
<td>linear</td>
<td>linear</td>
</tr>
</tbody>
</table>

**Table 3.1**: Complexity of different methods for object detection in the number of classes. For [Fidler and Leonardis 2007, Fidler et al. 2010], the constellation model of contour features is used. Although both our method and [Fidler et al. 2010] are sub-linear, [Fidler et al. 2010] uses coarse-to-fine representation of contour features at fixed locations which leads to accuracy loss.

The effect of different optimization functions for training is shown in Fig. 3.5.

#### 3.5.2 Scalability

Table 3.1 compares the complexity of our approach to our baselines and the state-of-the-art. Since we are using random decision forests for matching a patch to the codebook, our matching cost is logarithmic in the number of codebook entries. However, as discussed in Sec. 3.3, the complexity of detection also depends on the size of the codebook and the total number of cast votes. Figure 3.9 compares the codebook size and the number of votes for one-vs-the-rest detectors and our multi-class approach with and without using taxonomies. For this experiment, different number of classes are chosen at random from the VOC’06 dataset and a multi-class detector, with the same settings as above, is trained for them. Although the number of votes scales sub-linearly in the multi-class approach, using taxonomies adds further reductions. Figure 3.6(c) shows the per class speed-ups when using the taxonomy shown in Fig. 3.6(a).
3. Scalable Multiclass Detection

Figure 3.9: Our scalable method shows a sub-linear growth in the size of codebook (a) and in the number of votes (b) as opposed to linear in one-vs-the-rest. The taxonomy further reduces the number of votes. (20* uses VOC’07)

3.5.3 Detection

For detection at every scale, a dense set of features is extracted and matched to the forest and their votes are cast to a voting space similar to [Gall and Lempitsky 2009]. When using the taxonomy, the votes are cast only for the categories with weights bigger than the threshold as described in Sec. 3.4.4. The voting space is implemented by having a separate accumulator for every class, but since the number of cast votes to this accumulator scales sub-linearly, so does the detection. All hypotheses up to a certain threshold for each class are detected and their bounding boxes are estimated using backprojection similar to [Razavi et al. 2010] and no further non-max-suppression is performed before verification. For verification, the final verification classifier [Felzenszwalb et al. 2009] is applied to every hypothesis. For fairness in comparison with [Felzenszwalb et al. 2009], the same bounding boxes and non-maxima suppression are used after verification. Tables 3.2 and 3.4 summarize the accuracy of our two-stage multi-class detector. Table 3.3 compares the number of windows passed to the verification for both VOC’06 and VOC’07 datasets. Since in our system the number of evaluated windows is three orders of magnitude lower than a sliding window
### Table 3.2: Performance comparison of our multi-class method (MC) with the baselines in average-precision for VOC’06 dataset. The first block shows the detection without verification and without non-maxima suppression. MC outperforms one-vs-the-rest. The taxonomy not only reduces the amount of voting (Fig. 3.9), it also gives a slight improvement. In the second block, verification is performed with [Felzenszwalb et al. 2009]. By using a two-stage method, we are not losing accuracy compared to [Felzenszwalb et al. 2009]. The number of performed verifications is given in Table 3.3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Bicycle</th>
<th>Bus</th>
<th>Car</th>
<th>Cat</th>
<th>Cow</th>
<th>Dog</th>
<th>Horse</th>
<th>Motorbike</th>
<th>Person</th>
<th>Sheep</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Non-maxima Suppression</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One-vs-the-rest</td>
<td>.16 .13</td>
<td>.07</td>
<td>.04</td>
<td>.18</td>
<td>.03</td>
<td>.15</td>
<td>.16</td>
<td>.11</td>
<td>.12</td>
<td></td>
<td>.114</td>
</tr>
<tr>
<td>MC</td>
<td>.37 .12</td>
<td>.11</td>
<td>.02</td>
<td>.14</td>
<td>.05</td>
<td>.08</td>
<td>.21</td>
<td>.05</td>
<td>.12</td>
<td></td>
<td>.127</td>
</tr>
<tr>
<td>MC + Tax.</td>
<td>.38 .13</td>
<td>.12</td>
<td>.05</td>
<td>.15</td>
<td>.03</td>
<td>.11</td>
<td>.12</td>
<td>.05</td>
<td>.12</td>
<td></td>
<td>.132</td>
</tr>
<tr>
<td>Using LSVM for Verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSVM</td>
<td>.64</td>
<td>.62</td>
<td>.23</td>
<td>.46</td>
<td>.45</td>
<td>.61</td>
<td>.38</td>
<td>.45</td>
<td></td>
<td></td>
<td>.459</td>
</tr>
<tr>
<td>one-vs-the-rest+verif.</td>
<td>.67</td>
<td>.62</td>
<td>.23</td>
<td>.46</td>
<td>.46</td>
<td>.62</td>
<td>.35</td>
<td>.43</td>
<td></td>
<td></td>
<td>.461</td>
</tr>
<tr>
<td>MC+verif.</td>
<td>.68</td>
<td>.64</td>
<td>.65</td>
<td>.20</td>
<td>.47</td>
<td>.14</td>
<td>.44</td>
<td>.63</td>
<td>.38</td>
<td></td>
<td>.465</td>
</tr>
<tr>
<td>MC+T.+verif.</td>
<td>.66</td>
<td>.64</td>
<td>.66</td>
<td>.22</td>
<td>.47</td>
<td>.14</td>
<td>.44</td>
<td>.64</td>
<td>.36</td>
<td>.42</td>
<td>.463</td>
</tr>
</tbody>
</table>

In this chapter, we have presented a patch-based method for scalable detection of object categories. The computational complexity of the...
Table 3.3: Our multi-class detection approach reduces the number of windows for verification per image by three orders of magnitude. Unlike sliding window approaches, our method assigns a class to each returned window. This eases the verification as one classifier should be evaluated per window, and this without compromising accuracy; see Tables 3.2 and 3.4.

Our approach in linear for training and very sub-linear at testing that is necessary for detecting large number of categories efficiently. The method makes use of a shared vocabulary of local patches specifically trained to optimize the detection performance.

By providing a complexity analysis of non-parametric detection using an Implicit Shape Model [Leibe et al. 2008], we have shown that the entries of this vocabulary need to be of middle-complexity, i.e. establishing a trade-off between discriminative power and generalization. In fact, the codebook entries need to be able to discriminate between most classes yet be shared among a small number of classes at the same time. This is achieved by treating location/configuration and appearance of the patches differently. While the appearances of the patches in the codebook can be shared, when combined with location the patches need to discriminate between all classes. We have proposed a new objective function for maximizing the discriminative power of the words while training the codebook and suggested to control the amount of sharing between words by the size and resolution of the patches, i.e. patch complexity.

Our approach also benefits from an automatically built category taxonomy for robust scalability without compromising accuracy. The proposed detector is very efficient, yet the over all precision of the detector is not satisfactory. We have shown how to address this issue by running a verification classifier on top of the detections in a scalable fashion. In chapter 5, we will show how the accuracy of the detector can be improved by introducing latent variables in the Hough transform.
In the current implementation only simple intensity and HoG features have been used which are not appropriate for textured classes like cats and dogs. In addition, although high resolution training data is provided, the training completely discards this by rescaling all training images to relatively small sizes. In the future, we are planning to overcome these shortcomings by using texture features and building a multi-resolution representation of patches similar to [Park et al. 2010].

**Table 3.4**: Performance evaluation on VOC’07 dataset in average precision (AP). Our scalable method with verification (MC+verif.) achieves very similar results to the baseline [Felzenszwalb et al. 2009] although using LSVM on significantly lower number of subwindows.
In the previous chapter, we have discussed how to train a multi-class detector for scalable detection of object categories. Yet, the visual detection of the objects is only an initial step in many applications. In these applications it is also necessary to determine additional properties such as pose or 3D geometry for a detection or track an object instance over time. For example, consider a robotic application for autonomous driving. In this scenario, at every frame the pedestrians and cars need to be detected. However, the robotic car needs to also estimate the orientation of the cars and the pose of the the pedestrians. These objects in the scene also need to be tracked to enable prediction of their future moves.

A common way of determining object properties is to train competing detectors for each property by treating them as separate classes [Leibe et al. 2007, Seemann et al. 2006, Felzenszwalb et al. 2009, Torralba et al. 2004, Thomas et al. 2006, Ozuysal et al. 2009]. Some recent approaches e.g. [Malisiewicz et al. 2011] even go as far as to train a detector for each object instance in the training data. However, these works suffer from a serious shortcoming. In fact, in these approaches, the training data is split into subgroups according to the object attributes like viewpoint, pose, etc. This splitting of the training data leads to few training images for some attribute combinations which becomes problematic. As we show in this chapter, although the approach of splitting the training data into subgroups performs better than training a generic detector when the training data is abundant, the detection with this model becomes unreliable when the number of training data per group is low. In the latter case, training a joint detector for all object poses performs significantly better.
Common approaches for tracking objects in a video rely on a motion model and an appearance model of an object. In these approaches the object is tracked by using mean shift [Comaniciu et al. 2000], Kalman [Wren et al. 1997], or particle filtering [Isard and Blake 1996] where the appearance of the object is modeled by manually selected or learned image features. The appearance model is usually updated over time to cope with changing lighting conditions, background changes, and object transformations in the 3D space [Jepson et al. 2003, Ross et al. 2008]. However, modeling the object appearance only based on the instance does not take into account the validity of the resulting appearance model with respect to the appearance model of the category; thus, the update of the appearance can lead to a significant drift, *i.e.* the appearance adapts to another object or background.

In this work, we propose to use the the supporting votes of a detection, or in short *support*, to describe an object instance. The support of a detection contains all information used for separating an object from the background and thus we argue that it can be used as a rich source

---

**Figure 4.1:** Object detection using Implicit Shape Models: Features are matched against the codebook $C$ casting votes to the voting space $V$. The local maxima of the voting space is localized and the votes contributing to it are identified (inside the blue circle). The contributing votes are backprojected to the image domain creating the backprojection mask. At right, visualization of the backprojection mask blended with the image. Note that the mask does not include the area occluded by a pedestrian. The image is taken from UIUC cars dataset [Agarwal and Awan 2004].
for describing an instance. In fact, the backprojection of the support has been previously used successfully to extract auxiliary information as well, yet it has not been fully explored. For example, in [Leibe et al. 2008], provided pixel-accurate segmentation of the training data and storing this information in the codebook, the backprojection is used for top-down segmentation and verification. In addition, in [Thomas et al. 2009, Seemann et al. 2007], the annotations of the training data encoded in the votes is backprojected to the image domain for meta-data transfer. Our approach is different from the previous work in that we use all the votes in the support to densely describe a detection instead of only backprojecting the annotations of the training data to the image domain.

In this chapter, we investigate a broader question of ’what can the support tell us about the detection?’ We show how the information in the support can be used to fully describe an object instance and be backprojected to the image domain to estimate aspect ratios or object bounding boxes. We investigate two additional applications of this descriptions for multi-view object detection and view estimation based on nearest neighbor retrieval (see Fig. 4.2) and tracking an object instance in a video by online adaptation of a class-specific codebook to an instance (see Fig. 4.3).

4.0.1 Multi-view Detection and Pose Estimation

For the task of multi-view object detection and view retrieval, we firstly show that the different views of an object can be handled by a single codebook similar to multi-class detection and even a single view-independent voting space since the support and its backprojection encode enough information to deal with the viewpoint variations. The proposed training procedure also makes better use of the training data by sharing features of different views. Having the shared vocabulary and depending on the availability of view annotations and amount of training data, two voting schemes are proposed which outperform the battery of one-vs-the rest detectors both in terms of accuracy and computational complexity, in particular, when only few training examples are available.

Secondly, we not only detect objects under large view variations with a single shared codebook but also use the support for estimating the continuous parameters like object’s pose or view-point. This estimation is
Figure 4.2: In this chapter we propose a two stage approach for estimating auxiliary variable like view point of a detection. (a) The object instances in an image are detected by running the detector on the image. (b) Using their support, the most similar examples of a set of view annotated objects are retrieved. (c) The view point is estimated by transferring the view of the nearest examples and based on a single image and without using any motion model.
Figure 4.3: In order to track an instance of a class, like a certain person from the class pedestrians, we adapt on-line a class-specific codebook to the instance. This has been done by creating an instance model using its supporting votes. (a) Blue box indicates the instance of interest. (b) Voting image obtained by an off-line trained codebook for pedestrians. (c) Voting image obtained by the proposed instance-specific codebook. Despite the presence of other pedestrians and the confusing background, the peak at the center of the instance is clearly visible and votes for background objects are reduced.

carried out by retrieving the nearest examples with annotations of view or pose and transferring these annotations to the detected object by majority voting. To this end, we introduce a similarity measure based on the support that works by establishing dense feature correspondences between two detections and can be used to measure the distance of any two detections. This measure is particularly interesting as it is part-based and naturally takes the spatial relationships of the parts into account. This similarity measure also inherits nice properties like good generalization and insensitivity to partial occlusions from the ISM model. We show its power in the task of view-retrieval in the presence of partial occlusions.

4.0.2 Instance Tracking

The tracking by detection approaches rely on a class-specific detector that is applied independently to all frames. The detections are then as-
sembled to tracks, *e.g.* by using the detections or the confidence of the
detector as observation for a particle filter. An off-line trained detector,
however, detects an instance of the class and does not consider the spe-
cific appearance model of the instance nor its background. Consequently,
it tries to solve a much more difficult task than object tracking, namely
to identify any instance of the class in any image. Since there is not a
perfect object detector, the detectors provide even high confidence for
parts of the background, *i.e.* false positives, and cannot distinguish be-
tween several instances of the same class. Even given a perfect detector
multiple instance of the object of interest might be present in an im-
age that brings about the problem of finding the correct associations for
tracking. A straightforward workaround is the combination of an off-line
class-specific detector with an on-line learned identifier that distinguish
the target from other instances or from background objects with a high
detector confidence. This, however, does not address the core of the
problem, namely that detectors are not designed for tracking.

In this work, we demonstrate that an off-line trained class-specific de-
tector can be transformed into an instance-specific detector on-the-fly.
To this end, we make use of a codebook-based detector [Gall and Lem-
pitsky 2009] that is trained on an object class. A codebook models the
spatial distribution and appearance of object parts. When matching an
image against a codebook, a certain set of codebook entries is activated
to cast probabilistic votes for the object. For a given object hypothe-
sis, one can collect the entries that voted for the object. In our case,
these entries can be regarded as a signature for the target of interest.
Since a change of pose and appearance can lead to an activation of very
different codebook entries, we learn the statistics for the target and the
background over time, *i.e.* we learn on-line the probability of each part
in the codebook belonging to the target. By taking the target-specific
statistics into account for voting, the target can be distinguished from
other instances in the background yielding a higher detection confidence
for the target, see Fig. 4.3. To cope with the multi-modal probability for
the location of the object, we couple the codebook with a particle filter.
4.1 Related Work

4.1.1 Multi-view Detection and Pose Estimation

Codebook-based detectors learn the mapping from image features into a Hough space where detection hypotheses are obtained by local maxima. To this end, a codebook of local appearance is trained by clustering a training set of image features and storing their relative location with respect to the object center. The spatial distribution can be estimated by a non-parametric Parzen estimate [Leibe et al. 2008] or a mixture of Gaussians [Lehmann et al. 2009]. In [Maji and Malik 2009], a max-margin framework re-weights the votes for better detection. While [Leibe et al. 2008] clusters the sparse image features only based on appearance, the spatial distribution of the image features is used as cue for the clustering in [Opelt et al. 2008, Shotton et al. 2008b]. Hough forests [Gall and Lempitsky 2009] use a random forest framework [Breiman 2001] instead of clustering for codebook creation. Random forests have also been used for real-time tracking [Lepetit et al. 2005] where the forest is trained on a single target object. The approach, however, is object-specific and does not generalize to other objects.

While the work of [Leibe et al. 2008] uses a non-parametric Parzen estimate for the spatial distribution and thus for the determination of the weights, the spatial distribution can also be estimated by a mixture of Gaussians to speed up the hypothesis generation process [Lehmann et al. 2009]. In [Maji and Malik 2009], a max-margin framework re-weights the votes for better detection. The voting structure has been addressed in [Ommer and Malik 2009], where voting lines are proposed to better cope with scale-location ambiguities.

In [Leibe et al. 2008], the backprojection has been used for verification. To this end, the training data is segmented and the local foreground-background masks are stored with the codebook. When a maximum is detected in the voting space, the local segmentation masks are used to infer a global segmentation for the detection in the image. The global segmentation is then in turn used to improve recognition by discarding votes from background and re-weighting the hypothesis. In [Thomas et al. 2009, Seemann et al. 2006], the individual parts of an object (e.g. front wheel of a motorbike) are also annotated in the training and used
to infer part-labels for a test instance. In this work, we neither focus on hypothesis verification nor require time consuming segmentation and part annotation of the training data.

The handling of multiple object views has also been addressed in the literature, see e.g. [Selinger and Nelson 2001]. Based on the ISM and a silhouette-based verification step [Leibe et al. 2008], the model can be extended by handling the viewpoint as an additional dimension. While [Seemann et al. 2006] uses the segmented training data to cluster the shapes, [Thomas et al. 2009, Thomas et al. 2006] trains a codebook for each annotated view and links the views together by appearance. Other approaches use the statistical or geometric relationships between views in the training data to reason about the 3D structure of the object [Kushal et al. 2007, Liebelt et al. 2008, Savarese and Fei-Fei 2007]. Since these approaches need many viewpoints of an object, they are very expensive in data. Although some of the missing data can be synthesized from existing data by interpolation [Chiu et al. 2007, Savarese and Fei-Fei 2008], the interpolation still requires a certain number of views in the training data. In [Farhadi et al. 2009], a discriminative approach is proposed to handle aspects more general than for a specific class. To this end, latent variables that model the aspect are inferred and the latent discriminative aspect parameters are then used for detection.

### 4.1.2 Tracking

One benefit of codebook detectors is the robustness to occlusions. Since only a small set of local patches is required to locate the object, the detection is still reliable when the object is partially occluded. The idea of voting has been exploited for tracking in [Adam et al. 2006] where the template of the object is represented by a set of local patches. Each patch is tracked independently and the patches vote for the center of the object. While [Adam et al. 2006] use a static template model, it has been shown that updating the template is necessary in many scenarios [Jepson et al. 2003, Ross et al. 2008].

Modeling the appearance of an object has been addressed in several works. Most relevant are the tracking-by-detection approaches that train a model to separate the object from the background via a discriminative
4.2. Detection Support

The approaches mainly differ in the classifier and the update strategy. For instance, an adaptive ensemble of classifiers based on support vector machines [Avidan 2007], adaptive feature selection [Collins et al. 2005, Wang et al. 2005, Woodley et al. 2007], incremental subspace learning [Ross et al. 2008], on-line boosting [Grabner et al. 2008], or multiple instance learning [Babenko et al. 2009] have been proposed. A combination of several classifiers has been used in [Verma et al. 2003, Tang et al. 2007, Li et al. 2008, Yu et al. 2008, Stalder et al. 2009, Breitenstein et al. 2009]. The classifiers have different properties such that the weakness of one is compensated by the others, e.g. by using off-line and on-line trained classifiers or by training the classifiers on different features. The fusion, however, is often performed in a non-probabilistic manner using a weighted sum or taking the maximum of the confidences. Our approach differs from these approaches in that it does not combine several binary classifiers but adapts a single class-specific codebook in a probabilistic manner.

4.2 Detection Support

In the section 3.2, we have introduced the multi-class detection with the codebook. Before explaining how the support of a detection is used to describe an instance, let us review some of the notations. To detect objects of a test image, the features $f_i$ are matched to the codebook $\Omega$ to obtain the probabilities $p(\omega_j|c)$, $p(I|\omega_j)$, and $p(l_i,s_i|c,h,\omega_j)$ for every matching entry $\omega_j$ and hypothesis $h$. As mentioned earlier, the spatial distributions $p(l_i,s_i|c,h,\omega_j)$ in an Implicit Shape Model are encoded non-parametrically by a Parzen-window estimation on votes from training occurrences $(\hat{l}_j^i, \hat{s}_j^i)\,$ stored in the entries of vocabulary.

For a hypothesis $h$ and label $c$, the contribution of occurrence $(\hat{l}_j^i, \hat{s}_j^i)\,$ stored at entry $\omega_j$ from feature $i$, denoted by $v_{j,occ}^i(h,c)$ can be written as

$$v_{j,occ}^i(h,c) = p(I|\omega_j)p(\omega_j|c)K \left( \frac{\hat{l}_j^i, \hat{s}_j^i - \hat{l}_i^j, \hat{s}_i^j\,}{b(s_h)} \right)$$  (4.1)
where $K(.,.)$ is the band-width adaptive kernel used for density estimation. We define the support of the hypothesis $h$ as the collection of all the votes with a positive contribution to it

$$S_h^c = \{v_{j,occ}^i | p(\omega_j|c) > 0 \land v_{j,occ}^i(h,c) > 0\}. \quad (4.2)$$

Since in an ISM each vote is originating from an occurrence of a training patch in the codeword $\omega_j$, the support encodes the information in the appearance and relative location of the hypothesis patches in terms of the occurrences of previously observed training patches. In this chapter, we show how this information can be used to infer auxiliary information about an object instance.

### 4.2.1 Backprojection of the Support

**Bounding Box Estimation**

As described earlier, in many object detection approaches the location of an object hypothesis is explained by its bounding box. The bounding box aspect ratio and its scale can vary greatly among instances, however. Instances of an object category appear in different scales in an image and can be viewed from different angles which due to perspective effects can in turn affect the aspect ratio. In addition, object instances can be partially visible in an image due to occlusions or truncations leading to varying aspect ratios.

Despite its definition as the “bounding box” of an object instance, most approaches (e.g. [Leibe et al. 2008, Leibe et al. 2007, Gall and Lempitsky 2009]) estimate the extent of the object’s presence by placing the average bounding box of all training images scaled and translated to the detection center. Although this measure is sufficiently accurate for the rather generous standard evaluation criteria like [Everingham et al. 2006b], this measure is not applicable to multi-view detection with a single detector where aspect ratios of different views widely vary. Inspired by [Leibe et al. 2008], we propose using the backprojection of the supporting features for this purpose. The backprojection can be thought of as a backward mapping $B$ of the votes to the image domain leading to a real valued mask $\mathcal{M}$:

$$B : S \mapsto \mathcal{M} \quad (4.3)$$
4.2. Detection Support

Figure 4.4: Some example cars from the UIUC dataset together with their backprojection masks. Although only bounding box annotations are used for training, the backprojection can lead to a relatively accurate segmentation mask; already telling us about the occlusions.

where every element of $M_{h,i}^c$ holds the total contribution of feature $f_i$ to the hypothesis $h$ with label $c$

$$M_{h,i}^c = \sum_{j,occ} v_{j,occ}^i(h, c). \quad (4.4)$$

In our work, the backprojection mask is simply thresholded by an adaptive threshold (set to half the value range) to form a binary mask. And the tightest bounding box encompassing this mask is used as our bounding box estimate. Of course this is an oversimplification and there is still the possibility of more sophisticated bounding box estimations, e.g. [Blaschko and Lampert 2008], but simple thresholding suffices to obtain reasonable bounding box estimates.

Backprojection to Training Examples

Similar to [Seemann et al. 2006, Thomas et al. 2009], by conditioning the back-projection of a hypothesis' support $S_h^c$ to the votes coming from a single training example with identity $t \in T$, we can obtain the supporting votes of $h$ from $t$, $S_h^t$. Similar to the previous section, a mask $M_{h,i}^t$ can be obtained from backprojection of votes originating from $t$ for hypothesis $h$

$$M_{h,i}^t = \sum_{j,occ} v_{j,occ}^i(h, t) \quad (4.5)$$

where $v_{j,occ}^i(h, t)$ is the vote to hypothesis $h$ from feature $i$, code word $\omega_j$ and training occurrence $occ$ originating from training image $t$. In
addition, by summing over the weight of votes originating from \( t \), one can measure how much \( t \) contributes to the detection of \( h \). We denote this contribution by \( V(h, t) \) and compute it as the sum of the contributing votes from \( t \)

\[
V(h, t) = \sum_{j,occ} v_{j,occ}^i(h, t).
\]

The contribution from training example \( t \) to hypothesis \( h \), i.e. \( V(h, t) \), can be used as a holistic measure of similarity between the hypothesis \( h \) and the training image \( tr \). In principle, by introducing additional constraints, one can enforce more specific similarity measures, e.g. similarity of the left side of \( h \) to the left side of \( t \). Since we sample only a sparse set of patches from the training examples during training, this measure establishes correspondences between sparse locations of the detection and a training example. This sparse correspondence although fast for calculation is not very accurate and is only applicable as a distance between training images and a hypothesis. Later in this chapter, we will introduce a dense similarity measure between any two detections which overcomes these limitations.

### 4.2.2 Multi-view Detection

As mentioned earlier, similar to multi-class detection, multi-view detection in most approaches is carried out by first splitting the training examples according to a number of quantized views and training a separate detector for each view. For detection, these detectors are run simultaneously on the test image and the objects are detected as their maximum
response. The first reason for running a battery of competing detectors for the views is the varying bounding box aspect ratios of different views. The appearance of the object across views can also change considerably which is another reason for separating the views.

Since by backprojecting the supporting votes to the image domain, bounding box aspect ratio can be well estimated, we can now evaluate the effect of appearance change by comparing the performance of detecting views separately by also voting for the views in addition to the class which we call as separate voting, or voting only for the object class, joint voting.

Separate Voting:

Voting in separate voting is performed in a 4D voting space where in addition to the position and scale and the class $c$ the viewpoint/aspect, $a \in A$, of the object is also used for voting. Similar to the multi-class case, this is implemented by recording the view annotations for each training occurrence in the vocabulary and voting with them to obtain the image likelihoods

$$p(I|h, c, a) \approx \sum_i \sum_{\omega_j \in \Omega} p(I_i|\omega_j)p(\omega_j|c, a)p(l_i, s_i|\omega_j, c, a, h)$$ (4.7)

where the word probability $p(\omega_j|c, a, h)$ and the location prior $p(l_i, s_i|\omega_j, c, a, h)$ are obtained exactly the same as before by performing density estimation on the training occurrences. In this voting scheme, only votes from training examples of a particular viewpoint can contribute to a hypothesis of that view. In this sense, the view is a global condition on the independent assumption which ensures certain consistencies among votes. We will get back to this in Chapter 5 where we discuss how the optimal global condition for enforcing vote consistency can be learned.

Joint Voting:

In the joint voting, we are interested in obtaining the likelihoods $p(I|h, c)$ without using the aspect annotations for voting. One way of obtaining these likelihoods is to simply ignore the aspects for voting. Yet, this
likelihood can also be calculated by marginalizing the aspect likelihoods over all possible aspects.

\[ p(I|h, c) = \sum_{a \in A} p(I|h, c, a)p(a|h, c) \]  

(4.8)

where \( p(a|h, c) \) determines the prior of seeing an aspect given a certain class and hypothesis. An advantage of this way of voting is that the aspect priors are easier to set where typically we assume a uniform prior over the aspects. In this voting scheme, votes from different training examples and from different views can contribute to a hypothesis and detection is performed without using the aspect annotations.

4.3 Support as Instance Signature

As mentioned earlier, the support contains all information used for separating an object from its background and determine its class, aspect, bounding box, etc.. In this section, we show how the support can be used as a signature to describe an object instance. This description is used in two applications i) to measure the similarity between two detections and ii) to track an instance in a video.

4.3.1 Support Intersection as a Similarity Measure

We propose to measure the similarity of two object instances by intersecting their supports. Let us assume that we have two hypotheses \( h_1 \) and \( h_2 \) for which we have gathered their supporting votes \( S_{h_1} \) and \( S_{h_2} \). We define the support intersection of two hypotheses \( h_1 \) and \( h_2 \) as:

\[ S_{h_1} \cap S_{h_2} = \frac{\sum_{j,occ} \zeta_{occ}^j p(\omega_j|c_{occ}^j)\mathbb{I}(h_1, j, occ)\mathbb{I}(h_2, j, occ)}{\sum_{j,occ} \zeta_{occ}^j p(\omega_j|c_{occ}^j)\mathbb{I}(h_1, j, occ)}. \]  

(4.9)

where \( \zeta_{occ}^j \) is the normalization factor calculated from Eq.(3.3) and \( \mathbb{I}(h, j, occ) \) indicates whether an occurrence is contributing to a support or not

\[ \mathbb{I}(h, j, occ) = \begin{cases} 1 & \text{if } \sum_i \sum_{c \in C} v_{i,occ}^j(h, c) > 0 \\ 0 & \text{otherwise.} \end{cases} \]  

(4.10)
Note that the similarity measure is not symmetric due to the normalization factor. Yet, this factor is important as it makes the measure independent of the detection weight and that itself accounts for the occluded regions. There is a close link between the support intersection in Eq. (4.9) and the histogram intersection kernel used in bag-of-words image classification [Lazebnik et al. 2006] and the ISM [Leibe et al. 2008]. This said, there are also substantial differences between the two since the proposed measure is not defined on the level of code words but votes and the deformation cost is limited to zero-one to put more emphasis on the appearance of the detections. Since the detection is done with ISM, the support of the detection takes the spatial relationships of the features into account; therefore, there is no need for a fixed grid on top of the bag-of-words representation as in the spatial pyramid kernel [Lazebnik et al. 2006]. In addition, this metric is part-based and benefits from the generalization capabilities of these models and their insensitivity to occlusions, as will be shown in our experiments.

It is worthwhile to note an important difference between the similarity measures in Eqs. (4.6) and (4.9). The similarity measure in Eq. (4.6) can only be used to find the similarity between sparse patches of a detection and a training example, i.e. only matching to the same patches sampled during training. On the other hand, support intersection establishes a dense feature correspondence between any two detections. Due to the dense correspondences in Eq. (4.9), this similarity measure has a computational cost in the order of the number of votes in its support. However, this is about the same cost it takes to consider sparse correspondences to all training examples in Eq. (4.6) and thus is more costly yet more accurate.

### 4.3.2 Instance Specific Tracking

The class-specific detector described previously aims at estimating the image likelihood $p(I|h,c)$ for a particular class $c$. For tracking, however, one is not interested in the evidence for any instance of the class. In this application, the likelihood of the presence of a particular instance $\tau$ of class $c$, i.e. $p(I|h,c,\tau)$, needs to be estimated. Similar to the multi-
Figure 4.6: The probability \( p(v_{j,\text{occ}}|\tau) \) for 2500 occurrences in the vocabulary. The probability is on-line estimated by Equation (4.13) and gives an instance-specific signature that can be used to improve tracking, see Fig. 4.3. While occurrences with probability > 0.5 are specific to the instance \( \tau \), occurrences with probabilities < 0.5 vote for non-instance hypotheses. The estimates shown are from two sequences after 100 frames. The occurrences with probability equal to 0.5 are mainly patches that have not been activated during tracking.

As explained in Chapter 3, the word likelihoods \( p(\omega_j|c,\tau) \) and location priors \( p(l_i, s_i|\omega_j, c, \tau) \) are estimated non-parametrically from the training occurrences stored in the vocabulary. To this end, the likelihoods can be written as the sum of votes from these occurrences

\[
p(I|h, c, \tau) \approx \sum_i \sum_{\omega_j \in \Omega} p(I_i|\omega_j)p(\omega_j|c, \tau)p(l_i, s_i|\omega_j, c, \tau, h) \quad (4.11)
\]

As explained in Chapter 3, the word likelihoods \( p(\omega_j|c, \tau) \) and location priors \( p(l_i, s_i|\omega_j, c, \tau) \) are estimated non-parametrically from the training occurrences stored in the vocabulary. To this end, the likelihoods can be written as the sum of votes from these occurrences

\[
p(I|h, c, \tau) \approx \sum_i \sum_{\omega_j \in \Omega} \sum_{\text{occ}} v_{j,\text{occ}}^i(h, c, \tau) \quad (4.12)
\]

where \( v_{j,\text{occ}}^i(h, c, \tau) \) is calculated using Eq. (4.1) given the labeling of occurrences for \( c \) and \( \tau \). Although in the class specific vocabulary the class label of every occurrence is annotated, for tracking we need to determine the instance label of every occurrence. To this end, given initial detections of an instance in multiple frames, we first collect a set of hypotheses \( \mathcal{H} \) and label the ones belonging to the instance \( \tau, \mathcal{H}^\tau \). For
the occurrence $occ \in \omega_j$, we then first count the number of times a particular occurrence is contributing to the support of target hypotheses, 
\[ \sum_{h \in H^\tau} \mathbb{I}(h, j, occ), \] and the number of times it belongs to other hypotheses, 
\[ \sum_{h \notin H^\tau} \mathbb{I}(h, j, occ), \] and then use this information to determine the probability of a vote belonging to the instance, 
\[ p(v_{j,occ}|c, \tau) = \begin{cases} 0.5 & \text{if } \sum_{h \in H} \mathbb{I}(h, j, occ) = 0 \\ \frac{\sum_{h \in H^\tau} \mathbb{I}(h, j, occ)}{\sum_{h \notin H^\tau} \mathbb{I}(h, j, occ)} & \text{otherwise} \end{cases} \] \hspace{1cm} (4.13) where $\mathbb{I}(h, j, occ)$ is calculated according to Eq.(4.10). The first row assumes a fifty-fifty chance for an occurrence to belong to the target instance if it has not been observed in the previous frames yet. Figure 4.6, shows the value of the instance probabilities of two target instances for 2500 occurrences of the vocabulary. Using the instance probability of a vote, we calculate the votes for an instance as
\[ v_{i,j,occ}(h, c, \tau) = p(v_{j,occ}|\tau)v_{i,j,occ}(h, c). \] Note that the update neither changes the offset of the occurrences nor does it add new patches to the leaves. The update only performs a re-weighting of the votes. On the one hand, the localization accuracy does not suffer from the updates as it might be the case for other online learning approaches, where the appearance model drifts away from the target due to poor localization in the previous frames. In the worst case, the tracker is confused by other instances of the class or better to say by objects that are very similar according to the off-line trained detector. On the other hand, instances that are not localized a-priori by the codebook detector cannot be tracked since new observations are not added. In our experiments, we will demonstrate that the tracker handles difficult classes like humans that have a high frame-to-frame variation due to pose and many similarities between instances due to similar clothing.

In order to compute (4.13) for the current frame, we split the voting space into three clusters. This procedure is illustrated in Fig. 4.7. Since the scale is not important for this step, we only consider the votes at scale $\hat{s}_{t-1}$. However, the full voting space could also be clustered if necessary. For clustering, we label each $v_{i,occ}(h, c)$ by 1 or $-1$ if we are confident that it either belongs to the instance $\tau$ or it does not. If we are not confident, we ignore the vote and assign it to no hypothesis.
4. Backprojection

Figure 4.7: On-line adaption of the codebook. (a) After updating the particles, the multi-modal posterior distribution is approximated. The weights of the particles are indicated by color (yellow: high, red: low). The target is marked by a blue dot. (b) Based on the posterior, the voting space is clustered (blue: foreground, red: background, green: uncertain). Note that the intensity of the background (red) has been increased for a better visibility. (c) Votes that contributed to the detected local maxima are used to update the instance-specific statistics. Note that there is no local maximum for the most left person of the foreground (blue).

4.3.3 Tracking

For tracking, we couple the codebook with a particle filter [Doucet et al. 2001]. In the following, we first describe the dynamical model and how it is combined with the instance likelihoods for tracking.

Prediction

We estimate the position $x$, velocity $\beta$, acceleration $\alpha$, and the scale $s$ of the object in pixels. Assuming a constant acceleration model, we have:

$$x_t = x_{t-1} + \Delta t \cdot \beta_{t-1} + \frac{1}{2}(\Delta t)^2 \cdot \alpha_{t-1} + \eta_x(\hat{s}_{t-1})$$  \hspace{1cm} (4.15)

$$s_t = s_{t-1} \cdot (1 + \eta_s)$$  \hspace{1cm} (4.16)

$$\beta_t = \beta_{t-1} + \Delta t \cdot \alpha_{t-1} + \eta_\beta(\hat{s}_{t-1})\alpha_t = \alpha_{t-1} + \eta_\alpha(\hat{s}_{t-1})$$  \hspace{1cm} (4.17)
The noise terms $\eta$ depend on the estimated scale $\hat{s}_{t-1}$ of the previous frame and are modeled by zero mean Gaussians with standard deviation

$$\sigma_x = \max\{4(\hat{s}_{t-1}/s_u), 4\}, \sigma_\beta = 0.5 \Delta t \cdot \sigma_x, \sigma_\alpha = \sigma_\beta^2, \text{ and } \sigma_s = 0.04,$$

where $s_u$ is the scale of the training data.

**Update**

To get the posterior probability, the hypotheses $h_k$ associated with each particle $k$ are weighted by the likelihood $p(I^t|h_k, \tau, c)$, where $I^t$ is the current image. An example of the posterior approximated by a weighted set of particles is given in Figure 4.7 (a). Before the posteriors are normalized and the particles are re-sampled [Doucet et al. 2001], the un-normalized weights are used for updating the codebook.

### 4.4 Experiments

In order to assess the performance of the multi-view detectors described in Sect. 4.2.2, we use three datasets. The multi-view Leuven-cars dataset [Leibe et al. 2007] contains 1471 training cars annotated with seven different views and a sequence of 1175 images for testing. The multi-view Leuven-motorbikes dataset [Thomas et al. 2006] contains 217 training images annotated with 16 quantized views and 179 test images. And the PASCAL VOC’06 cars datset [Everingham et al. 2006a]. Further experiments for nearest neighbor retrieval are carried out on the TUD-pedestrians dataset introduced in [Andriluka et al. 2008] and the cars datasets. The TUD-pedestrians dataset provides 400 training images and 250 images for testing. Throughout the experiments, only bounding box annotations of the training images are used. The segmentation masks that are provided for some datasets are discarded. For a quantitative evaluation of tracking, we use two standard datasets i-Lids [Branch] and PETS09 [Ferryman et al.] that have been recorded in an underground station and a public place. The sequences contain several instances (persons) of the class (pedestrians).
Figure 4.8: (a) After a hypothesis is detected, its support is backprojected to the image domain to estimate its bounding box. (b) Estimating the bounding boxes from backprojection (Multi-Aspect Ratio) leads to more accurate estimation than using average training aspect ratio (Average Aspect Ratio).

4.4.1 Multi-view Detection

As a baseline comparison for the multi-view detection, we consider the popular one-vs-the-rest detector. For each view, the training is carried out with the positive training images of a view versus random patches from the Caltech 256 clutter set plus all the positive training images of the other views. An edge detector has been carried out both for training and testing and only features with their center on an edge are considered.

In order to make fair comparisons, the training and detection parameters are kept the same throughout the experiments. In particular, the number of trees in each forest is set to 15. From each training image 100 patches are i.i.d sampled and the number of background patches is kept constant at 20000 patches. For detection, the kernel used for the density estimation is a Gaussian with $\sigma = 2.5$ and the first 20 local maxima per image are considered. When the backprojection is not used for bounding box estimation, non-maxima suppression is done by removing all detections whose centers are within the bounding box (with 90% of its size) of another detection with a higher weight. When using backprojection, the hypothesis with the highest weight is included and its features are removed from all other hypotheses, thereby decreasing their weights.
4.4. Experiments

Figure 4.9: (a) Detection performance for the Leuven-cars dataset and comparison to Leibe et al [Leibe et al. 2007]. Separate voting with bounding boxes estimated from backprojection (bp) achieves the best performance. Joint voting without using view annotations gives competitive results. (b) Performance comparison of joint-voting with state-of-the-art approaches on PASCAL VOC 2006 dataset.

Figure 4.10: (a) Joint voting achieves better results than separate voting because of insufficient training data per view and finely quantized views although it does not use view annotations. Estimating the bounding box from backprojection even leads to a small improvement. (b) Sharing of features across views of motorbikes. (c) Performance comparison to state-of-the-art multi-view detectors for the motorbikes dataset.
The results of this experiment are shown in Fig. 4.9. As can be seen, separate voting with the help of backprojection performs best and estimating the bounding box with backprojection slightly increases the performance of the system. Joint voting also shows a competitive performance. It is worthwhile to note that the superior performance of separate voting is mainly due to the abundance of training images per view in this dataset and the presence of additional view information not used by joint voting. By sharing features across views, as e.g. shown in the work of Torralba et al. [Torralba et al. 2007], one expects to benefit mainly when the training data is limited. In order to verify this, we did the following experiment.

We compare the performance of joint voting, separate voting, and a battery of independent one-vs-the-background classifiers for 50, 25, 10, and 5 training images per view (7 views). In all the experiments, the full background set from Caltech 256 clutter is used and the set of training images for all three detectors is identical. Joint voting and separate voting use an identical shared codebook whereas a separate codebook is trained per view for the independent detector (see Fig. 4.11). With fewer training examples, as expected, the performance of all three detectors degrades, but that of joint voting far more gently. In particular, the comparison of separate voting and joint voting for few training images is very interesting. Although an identical codebook is used, joint voting significantly outperforms separate voting. This performance gap seems to narrow by using several codebooks (one for each view) and thus more codebook entries for the independent detector but the performance of joint voting is still superior in terms of accuracy as well as training and detection time.

In order to assess performances on a more challenging dataset, we evaluated joint voting and separate voting for the Leuven-motorbikes dataset [Thomas et al. 2006] where the test set is provided by the PASCAL VOC Challenge [Everingham et al. 2006b]. The motorbikes have more variability in their appearance and the views are quantified finer because of the larger variability in aspect ratios. For the sake of a fair comparison the same training and test settings as in [Thomas et al. 2006] is used. The results of this experiment are shown in Fig. 4.10(a). Note that the detection result with joint voting is obtained only using the bounding box annotations for the training data and using no view annotations.
4.4. Experiments

(a) 350 images

(b) 175 images

(c) 70 images

(d) 35 images

Figure 4.11: The effect of training size on the performance of joint voting, separate voting, and a battery of independent one-vs-the-background classifiers: With the abundance of training data per view, the separate-voting works best. The advantage of sharing is significant with lower number of training examples, especially compared to the separate voting with an identical codebook although no view annotations are used. Joint and separate voting outperform the independent detector in efficiency and/or accuracy.
important to note that the aim of the experiments is to show improvements over our baselines with the same parameters throughout experiments. The performance of joint voting and other state-of-the-art approaches is shown in Fig. 4.10(b) to give the reader an idea of the performance of other approaches compared to ours on this dataset. Note that in Thomas et al. [Thomas et al. 2006] and [Savarese and Fei-Fei 2007] pixel accurate segmented training data is used. In contrast to our approach and [Thomas et al. 2006], the sliding window approach of Savarese et al. [Savarese and Fei-Fei 2007] explicitly uses the geometrical relationships of different views. Although these relationships seem to be useful (better recall) for detection it comes at high computational costs which makes this approach not scalable to large datasets. In particular, testing with this approach has linear complexity in the number of training examples compared to logarithmic in our implementation. And training complexity is quadratic in the number of training images (linear in our case). In addition, although this work does not use view annotations, unlike our approach it needs many views of several training examples which are expensive to acquire. Some example detections on the Leuven cars and Motorbikes are shown in 4.12 and 4.13 respectively.
4.4. Experiments

Figure 4.13: Some example detections of our multi-view detector on motorbikes dataset. Similar to 4.12, the ground truth annotations are drawn in blue and false detections and correct detections are in red and green respectively.

Sharing features across views

One of the main advantages of training multiple views jointly is the sharing of features. In order to evaluate the capability of our method in doing so, we are creating a sharing matrix of size $n_{views} \times n_{views}$. Each element of this matrix shows, on average, how many features of the column view are used for detection of the row view. Since the test set of none of the datasets is annotated for views, this experiment is done on the set of training data with a leave-one-out strategy. When running the detector on a training instance, we are removing all the occurrences that are originating from that instance from the forest. The sharing matrices for the Leuven-cars and Leuven-motorbikes datasets are shown in Fig. 4.14.
Figure 4.14: Sharing in appearance and location among views for the motorbikes and cars as calculated in 3.16. As can be seen, the position and appearance of patches in both categories is shared across views. Due to the rapid change in appearance in the frontal and back views, these views share less with the side views.

**Figure 4.15:** (a) Viewpoint retrieval with the nearest neighbor using (4.9) (b) Moderate improvement by retrieving views by majority voting of 35 nearest neighbors. As can be seen, the nearest neighbor works already very good and the confusion appears to be limited to nearby views. In particular, comparing the sharing pattern in Fig. 4.14(a) and view confusions is interesting; e.g. front and back views share many features but they have been separated well for view retrieval. This shows the presence of additional information in the support that is exploited.
4.4.2 Estimating View-point with Nearest Neighbors

As described in Sect. 4.3.1, support intersection can be used as metric to compare two detections. In order to assess the quality of this metric, we use it to retrieve the viewpoint of the detected cars in the Leuven and PASCAL VOC’06 cars datasets. To this end, we have hand-annotated the viewpoint of the full Leuven-cars test set. For the PASCAL VOC’06 cars set, the ground truth annotations were used. For the Leuven-cars, we have run the detector on the positive set of the training data and collected a set of detections. For the VOC’06 set, the same procedure is carried out but on the validation set. All detections are done with joint voting (see Sect. 4.2.2) and not using view annotations. By comparing the support of a test detection to the support of all positive collected detections using (4.9), the nearest neighbor is retrieved and the estimated view of it is assigned to the test detection. This has been done for all the true positives in the test set and their estimated viewpoint is stored. By comparing the estimated viewpoint with the ground truth annotations, the confusion matrix in Fig. 4.15(a) (with average diagonal of 43%) is created where the rows indicate the ground-truth viewpoints and columns are the estimated viewpoints. In order to see if retrieving more nearest neighbors would add robustness to this process, this experiment is repeated by retrieving the 35 nearest training detections for each test detection and assigning the viewpoint of the majority to it (with average diagonal of 50%). The results for the VOC’06 are given in Fig. 4.15(c,d). As can be seen, most confusion is happening between very similar views. Note that the features used in our detection system are relatively invariant with respect to small viewpoint changes and the training is done without using viewpoint annotations and in a way to optimize detection performance. In addition, there is a relatively large overlap in the annotation of nearby views due to the difficulty of precise viewpoint estimation even for humans. A video showing the estimated views for the entire Leuven-cars dataset is available under http://www.vision.ee.ethz.ch/~nrazavi. Figure 4.17 shows the estimated views and the nearest neighbors for two images of the Leuven sequence. The view retrieval performance on the PASCAL VOC 2006 dataset and its comparison to the state-of-the-art is shown in Fig. 4.16.
Figure 4.16: (c) View-point classification for detected cars in the VOC’06 dataset. As can be seen, the confusion appears to be limited only to similar views. (d) Comparison to state-of-the-art [Su et al. 2009, Sun et al. 2009] for view classification on VOC’06 (for a fair comparison detections up to 43% recall are considered; average accuracy 82%). Note that our nearest neighboring approach leads to superior performance and more balanced estimation. Comparing the sharing pattern and view confusions is also interesting; e.g. front and back views share many features but their view have been separated well. This shows the presence of additional information in the support.

The effect of occlusion:

In order to assess the quality of the support intersection similarity metric in the presence of occlusions, we have annotated all the cars in every tenth frame of the Leuven-cars sequence based on the amount of occlusion: not occluded, 10%, 20%, 30%, 40%, and > 50% occluded regions. In this experiment, first the detector, with the same settings as in the multi-view experiment, Sect. 4.4.1, is applied to all the images and a number of detections are retrieved for each image. Then for each correct detection, its viewpoint is estimated as described above. For each occlusion set, we have evaluated how accurately the viewpoint is estimated. The results in Fig. 4.18(b) show the robustness of this nearest neighbor metric with respect to partial occlusions.
Figure 4.17: View point estimation using nearest neighbor retrieval for two frames of the Leuven sequence. The nearest neighbors are shown in the right column.

4.4.3 Retrieving Nearest Training Examples

In Sect. 4.2.1, we have explained how backprojection can be used as a similarity measure between object hypothesis and the training examples. In the following experiment, we are using such information to estimate the distance between the ankles of pedestrians as an indicator of their pose; see Fig. 4.20. We carried out our experiments on the TUD-pedestrians dataset. Training data of this dataset has annotations of the joint positions and this information is exploited for estimating the Euclidean distance (in pixels) between the ankles of a test instance.

For the sake of evaluation, we have produced the same annotations for the test set. The distance between the ankles of the test instance is then estimated as the median of this distance in the $k$ NNs. Figure 4.19(b) shows the deviation of the estimated distance from the ground truth for different values of $k$. As a baseline, we also show the deviation from the
Figure 4.18: (a-b) The view retrieval performance using (4.9) together with the proportion of the cars detected depending on the amount of occluded regions for a subset of the Leuven-car sequence. (the last two sets, 50% and > 50%, have very few instances). The recall and view retrieval performances were calculated independently for each occlusion set. Interestingly, although the detection performance deteriorates from large occlusions (a), the viewpoint retrieval performance is affected very little (b) which shows robustness of this similarity measure to occlusions.

Figure 4.19: Distance between the ankles estimated by the median of the k nearest training images using (4.6) compared to mean and median as baselines. The estimation is robust even at high recall rates.
4.4. Experiments

Figure 4.20: Two test detections from TUD-pedestrians dataset and their top ten nearest training examples (top row; nearest examples ordered from left to right) and backprojections of detection support to them (bottom row) using (4.6). The blue box shows the estimated bounding box from the backprojection mask (blended). Note the similarity of the poses between the test instances and retrieved nearest training images.

ground truth if the distance is estimated by the mean or median distance of the whole training set.

4.4.4 Tracking

For comparison, we apply the proposed tracker with on-line adaption, a particle filter without any adaption, \textit{i.e.} using the class-specific codebook only, and a third version where the voting space is clustered only in foreground and background, \textit{i.e.} the codebook is always updated. As class-specific codebook, we use 5 pre-trained trees for the TUD pedestrian dataset [Andriluka \textit{et al.} 2008] which are public available [Gall and Lempitsky 2009]. All trackers run with 50 particles and are initialized by a given bounding box. As an estimate, we take the strongest
mode of the posterior. The accuracy is measured by the PASCAL VOC criterion\(^1\) that takes scale and position of the bounding box into account. The results in Figs. 4.21 and 4.22 show the benefit of the on-line adaption of a class-specific codebook to the target instance. While simple sequences without ambiguities can be handled by a class-specific codebook, more complex scenes with several instances cannot be tracked without the proposed on-line adaption. Note that the proposed approach is also faster, see Table 4.2. The results for some on-line boosting approaches [Babenko et al. 2009, Grabner et al. 2006, Grabner et al. 2008] are given in Table 4.1. However, we have to emphasize that the public available implementations [Babenko et al. 2009, Grabner et al. 2006, Grabner et al. 2008] neither handle scale nor make use of any off-line training.

We have also applied our algorithm to face tracking. To this end, we trained a codebook with 5 trees on a face dataset [Kumar et al. 2008] where we used clutter as background images [Griffin et al. 2007a]. The sequences we have tested on are shown in Fig. 4.23. Our approach successfully tracks some standard sequences that have been used in

\(^1\)\text{Accuracy}(est) = (A_{gt} \cap A_{est})/(A_{gt} \cup A_{est})$, where $A_{est}$ and $A_{gt}$ are the areas of the estimated and the ground truth bounding box, respectively. When the areas do not overlap the value is 0, when they are identical the value is 1.
4.5. Conclusions

Figure 4.22: Tracking accuracy for the sequence S2.L1 (view 1) of PETS09 [Ferryman et al.]. Neither the off-line codebook nor the update-always-strategy are able to track the person since they are misled by other pedestrians in the scene (a). The proposed approach tracks the target (blue box) despite of occlusions and similarity with other instances. Two frames are shown in (b) and (c). Note that the accuracy is zero for a few frames where the strongest mode of the multi-modal posterior distribution does not overlap with the target, see Fig. 4.7 (a).

In this chapter, we have shown how the support can be used to describe a detection and be used to infer auxiliary information about it. In particular, we have shown that the support of a detection and its backprojection the literature [Ross et al. 2008, Birchfield 1998, Adam et al. 2006, Babenko et al. 2009]. Occlusions and changing lighting conditions do not impose a burden on our algorithm. Although the training data contains only frontal faces, the 360 degree head rotation in the sequence Girl is well captured. Sometimes the tracker follows the face of the girl with a slight delay which is a limitation of our dynamical model. Since these research sequences are not as challenging as real world data, we have taken a few sequences from YouTube\textsuperscript{2}. The sequences cover a variety of challenges like low resolution, strong compression artifacts, motion blur, occlusions, and extreme lighting conditions ranging from very dark to very bright. In some frames, the face is very difficult to recognize without context. In addition, we have run the tracker on a very long sequence with 14166 frames and sequences with multiple faces.

4.5 Conclusions

In this chapter, we have shown how the support can be used to describe a detection and be used to infer auxiliary information about it. In particular, we have shown that the support of a detection and its backprojection

\textsuperscript{2}\url{www.youtube.com}. The sequences are part of the supplementary material.
Figure 4.23: From top to bottom. Standard test sequences: i-Lids hard [Branch] (surveillance, crowded scene); David Indoor [Ross et al. 2008] (changing lighting conditions); Girl [Birchfield 1998] (fast movements, 360 degree head rotation); Occluded Face [Adam et al. 2006] (occlusions); Occluded Face2 [Babenko et al. 2009] (strong occlusions). YouTube test sequences: Klaus Kinski 1971 (camera zoom, 14166 frames); David Attenborough Night (very dark); Desert (strong shadow); Walrus (fast movement); Top Gear (multiple faces); Top Gear Ariel Atom (face deformations); Monty Python Military (multiple faces); Florian Silbereisen (fast movements, low resolution). The last two are from public street parades (occlusions, multiple faces, low resolution, difficult lighting conditions).
4.5. Conclusions

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>i-Lids(easy)</th>
<th>i-Lids(med.)</th>
<th>i-Lids(hard)</th>
<th>PETS09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>67.4 ± 13.5</td>
<td>65.4 ± 12.2</td>
<td>65.9 ± 15.0</td>
<td>60.3 ± 15.3</td>
</tr>
<tr>
<td>Update All</td>
<td>69.2 ± 13.9</td>
<td>60.8 ± 14.7</td>
<td>61.4 ± 19.6</td>
<td>15.6 ± 29.0</td>
</tr>
<tr>
<td>No Update</td>
<td>66.9 ± 12.8</td>
<td>45.9 ± 33.9</td>
<td>53.2 ± 15.2</td>
<td>31.3 ± 29.9</td>
</tr>
</tbody>
</table>

[Grabner et al. 2006] 25.1 ± 21.3 23.1 ± 31.0 21.3 ± 32.4 8.1 ± 20.5
[Grabner et al. 2008] 0.8 ± 8.6 6.6 ± 21.1 6.8 ± 21.7 12.0 ± 24.6
[Babenko et al. 2009] 28.5 ± 18.9 35.9 ± 36.6 34.8 ± 37.7 8.4 ± 22.2

Table 4.1: Mean and standard deviation of the tracking accuracy.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>180msec.</td>
<td>0.3msec.</td>
<td>235msec.</td>
<td>63msec.</td>
<td>851msec.</td>
</tr>
</tbody>
</table>

Table 4.2: Since votes with zero probability are not cast, voting with on-line adaption is 2.8 times faster than voting with the class-specific codebook.

can be exploited to estimate the extent of a detection, retrieve nearest training examples, and establish an occlusion-insensitive similarity measure between two detections. In addition, we have demonstrated that a class-specific codebook can be transformed into a more instance-specific codebook in a probabilistic manner. Coupled with a particle filter, the codebook becomes already a powerful instance tracking method without the use of additional classifiers to distinguish several instances during tracking. Compared to a class-specific codebook, the accuracy is not only increased but the computation time is also reduced. Compared to on-line learning approaches, tracking is much more reliable subject to an off-line trained codebook. Although this prevents tracking arbitrary objects, it is not a practical limitation since the objects of interest usually belong to a well defined class. Real-time performance is not yet achieved, but it seems to be feasible by speeding-up feature extraction and voting.
Latent Hough Transform

In the previous chapters, we have presented a Hough transform based multi-class detector with scalable training and run-time behavior in the number of classes. In addition, we have shown how the support and its backprojection can be used to infer auxiliary properties of a detection. However, as shown in our experiments the accuracy of the resulting detector is lower than other competing approaches [Vedaldi et al. 2009, Felzenszwalb et al. 2009]. Although this performance gap can be filled by using a verification classifier, due to the probabilistic nature of our detector, it is inconvenient to use these verification classifiers in applications such as tracking where the detector needs to be seamlessly integrated and thus needs to be more accurate.

Detecting objects using the Hough transform is very efficient, however, it often fails to detect consistent hypotheses. As an instance of the Implicit Shape Model [Leibe et al. 2008], the proposed detector represents an object by the relative locations of specific image features with respect to a reference point. For object detection, the image features vote for the possible locations of this reference point. Although this representation allows parts from different training examples to support a single object hypothesis, it also produces false positives by accumulating votes that are consistent in location but inconsistent in other properties like pose, color, shape or type. For example, features taken from a frontal view image of a training example and a side view image of another training example might agree in location, but an object can not be seen from frontal and side views at the same time.

To improve the detection performance, researchers have proposed enforcement of consistency of the votes by estimating additional param-
eters like aspect ratio [Gall et al. 2011] or pose [Leibe et al. 2007, Seemann et al. 2006, Seemann et al. 2007, Razavi et al. 2010, Marszałek and Schmid 2007]. While the use of more parameters obviously improves consistency, it also increases the dimensionality of the Hough space. However, Hough transform-based methods are known to perform poorly for high-dimensional spaces [Stephens 1991]. Consistency can also be enforced by grouping the training data and voting for each group separately. Such a grouping can be defined based on manual annotations of the objects, if available, or obtained by clustering the training data. While this does not increase the dimensionality of the voting space, the votes for each group can become sparse due to a limited number of training examples, which impairs the detection performance. Even if annotations are available for the training examples, it is not clear which properties to annotate for an optimal detection performance since the importance of properties differs from case to case. For instance, pose is important for detecting airplanes but far less so for detecting balls.

In this chapter, we propose to augment the Hough transform by global latent variables to enforce consistency of the votes. This is done by only allowing votes that agree on the values of the latent variables to support a single hypothesis. We discriminatively learn the optimal assignments of the training data to these variables to improve object detection in the context of Hough transform-based methods. To this end, starting from a random assignment, the training examples are reassigned by optimizing an objective function that approximates the average precision on the training set. In order to make the optimization feasible, the objective function exploits the linearity of voting approaches. Further, we extend the concept that training images can only be assigned to a single latent value. In particular, we let training images assume multiple values and further allow these associations to be weighted, i.e. modeling the uncertainty involved in assigning a training example to them. This generalization makes the learned latent space continuous which reduces quantization artifacts and increases the robustness of the latent Hough transform with respect to the number of quantized latent values.

Experiments on the Leuven cars [Leibe et al. 2007] and the PASCAL VOC 2007 benchmark [Everingham et al. 2010] show that our latent Hough transform approach significantly outperforms the standard Hough transform. We also compare our method to other baselines with unsu-
supervised clustering of the training data or by manual annotations. In our experiments, we empirically demonstrate the superior performance of our latent approach over these baselines. The proposed method performs better than the best Hough transform based methods. And it even outperforms state-of-the-art detectors on some categories of the PASCAL VOC 2007.

5.1 Related Work

A number of previous works have investigated the idea of using object properties to group training data. For instance, the training data is clustered according to the aspect ratios of the bounding boxes in [Felzenszwalb et al. 2009]. Other grouping criteria like user-annotated silhouettes [Marszałek and Schmid 2007], viewpoints [Razavi et al. 2010, Torralba et al. 2004, Thomas et al. 2006, Ozuysal et al. 2009], and pose [Seemann et al. 2006] have been considered as well. To enforce consistency, the features vote for objects’ depth in addition to locations in [Sun et al. 2010], the close-by features are grouped in [Yarlagadda et al. 2010] to vote together. In [Girshick et al. 2011], two subtypes are trained for each part by mirroring the training data. The location of each part with respect to its parent joint is also used in [Yang and Ramanan 2011] to train sub-types. Instead of grouping the training images, [Malisiewicz et al. 2011] train a model for every single positive instance in the training data. While all of these works divide the training data into disjoint groups, [Torsello et al. 2008] proposes a generative clustering approach that allows overlapping groups.

Latent variable models have been successfully used in numerous areas of computer vision and machine learning to deal with unobserved variables in training data, e.g. [Hofmann 2001, Farhadi et al. 2009, Wang and Mori 2010, Bilen et al. 2011]. Most related to our work, [Felzenszwalb et al. 2009, Zhu et al. 2010] learn a latent mixture model for object detection by discriminatively grouping the training images and training a part model consisting of a root filter and a set of (hierarchical) parts for each group. In contrast to these works, our approach is a voting based method where we assume that we are given the parts in form of a shared vocabulary. As has been shown in chapter 4 and pre-
vious approaches [Torralba et al. 2004, Razavi et al. 2011], the shared vocabulary allows better generalization when learning a model with few examples as it makes use of the training data much more effectively. Given this vocabulary, we aim at learning latent groupings of the votes which lead to consistent models and improve detection accuracy. The advantage of this approach is that we need to train the parts only once and not re-train them from scratch as done in [Felzenszwalb et al. 2009, Zhu et al. 2010].

Several approaches have been proposed for training the parts vocabulary. Generative clustering of the interest points are used in [Leibe et al. 2008, Seemann et al. 2007, Maji and Malik 2009] as the codebook whereas [Gall et al. 2011, Razavi et al. 2010] discriminatively train a codebook by optimizing the classification and localization performance of image patches. Discriminative learning of weights for the codebook has been addressed before as well. In [Maji and Malik 2009], a weight is learned for each entry in a Max-Margin framework. [Zhang and Chen 2010] introduced a kernel to measure similarity between two images and used it as a kernel to train weights with an SVM classifier. Although increasing the detection performance, those works differ from our approach in that they train weights within a group.

Detecting consistent peaks in the Hough-space for localization has been the subject of many investigations as well. While Leibe et al. [Leibe et al. 2008] used a mean-shift mode estimation for accurate peak localizations, [Woodford et al. 2011] utilize an iterative procedure to “demist” the voting space by removing the improbable votes from the voting space. [Barnova et al. 2010] pose the detection as a labeling problem where each feature can belong to only a single hypothesis and propose an iterative greedy optimization by detecting the maximum scoring hypothesis and removing its votes from the voting space.

Learning discrete latent groupings of training data has a close link to the problem of categorization in psychology as well. It is well known that humans have a categorical perception of even continuous phenomena like color, or sound [Harnad 1990]. An important side effect of this phenomena is the uncanny valley effect where the subjects would feel uncomfortable when asked to classify an example at the border of two categories [Cheetham et al. 2011].
Some recent approaches have also proposed to share labels \cite{Fergus2010}, training examples \cite{Lim2011}, or knowledge \cite{Salakhutdinov2011} among categories for more robust detection. These works are very related to our latent variable formulation as it can be shown that the sharing in the context of these works is equivalent to learning a set of latent categories and can be represented by a latent matrix.

\section{Detection with the Hough transform}

In this section, we briefly describe Hough transform based object detection approaches that learn a codebook of voting elements, which can be some image features \cite{Leibe2008} or simply dense image patches \cite{Gall2011}. Since the procedure that follows does not require any retraining of the codebook, we refer for the details of the codebook learning to the Chapter 3. Having such a codebook, the features of an image $f_i \in F$ are extracted and matched against the codebook to cast weighted votes $V(h, c|f_i)$ for an object hypothesis $h \in H$ and class label $c \in C$. The hypothesis $h$ encodes the position and scale of the object in the image. For a given $h$ and $c$, the votes of all features are accumulated to obtain the hypothesis score:

$$V(h, c) = \sum_i V(h, c|f_i)$$  \hspace{1cm} (5.1)

i.e., the accumulated weights of the votes that agree on the location and scale of the object. Without loss of generality we assume that we are dealing with a single class detection and thus omit the class label $c$ from equations. The votes of each feature are estimated non-parametrically from the votes from training occurrences $occ$ stored at its matching code word $\omega_j \in \Omega$:

$$V(h|f_i) = \sum_{\omega_j \in \Omega} \sum_{occ} v^i_{j,occ}(h).$$  \hspace{1cm} (5.2)

For detecting multiple objects, following the probabilistic approach of \cite{Batinova2010}, the maximum scoring hypothesis is localized and its supporting votes are removed from the voting space. This process is iterated until the desired number of objects are detected or a confidence threshold is reached.
Since in an Implicit Shape Model [Leibe et al. 2008], the votes are estimated from training patches in a non-parametric way, the score of each hypothesis is linear in the votes and can be written as sum of votes from training images $t \in T$:

$$V(h) = \sum_{t \in T} \sum_i V(h|t, f_i)$$  \hspace{1cm} (5.3)

where $V(h|t, f_i)$ denotes the votes of feature $f_i$ from occurrences originating from the training image $t$. Although the votes originating from a single training example are always consistent, this formulation accumulates the votes over all training examples even if they are inconsistent, e.g., in pose or shape. In the next section, we show how to use latent variables to enforce consistency among votes.

### 5.3 Latent Hough transform

In detecting objects with the latent Hough transform, we augment the hypothesis space by a latent space $Z$ to enforce consistency of the votes in some latent properties $z \in Z$. The score of a hypothesis in the augmented space can be determined as

$$V(h) = \max_{z \in Z} \sum_{t \in T} \sum_i V(h, z|t, f_i)$$  \hspace{1cm} (5.4)

where $V(h, z|t, I_i)$ are the votes of an image element $I_i$ from training image $t$ to the augmented latent space $H \times Z$. For instance, $Z$ can be quantized viewpoints of an object. Voting in this augmented space allows only votes that are consistent in viewpoint to support a single hypothesis.

Similar to other latent variable models [Felzenszwalb et al. 2009, Farhadi et al. 2009, Wang and Mori 2010, Zhu et al. 2010], one can associate each training image $t$ to a single latent assignment $z$. This association groups the training data into $|Z|$ disjoint groups. Note that the number of these groups is limited by the size of training data $|T|$.

The grouping of the training data by latent assignments can be represented by a binary $|Z| \times |T|$ matrix, which we denote by $W$ and refer
to as latent matrix. The elements of $W$ are denoted by $w_{z,t}$, where $w_{z,t} \in \{0, 1\}$ and $\sum_z w_{z,t} = 1, \forall t \in \mathcal{T}$. Observe that every $W$ that satisfies these constraints defines a disjoint grouping of the training data. Given a latent matrix $W$, we can rewrite the hypothesis score as

$$V(h|W) = \max_{z \in \mathcal{Z}} \sum_{t \in \mathcal{T}} w_{z,t} V(h|t), \quad \text{where} \quad V(h|t) = \sum_i V(h|t,f_i). \quad (5.5)$$

The term $V(h|t)$ is the sum of the votes originated from the training example $t$. This term will be very important while learning the optimal $W$ as we will discuss in Sec. 5.4.

### 5.3.1 Generalized Latent Assignments

The association of the training data to a single value $z$ does not make use of the training data effectively. We therefore generalize the latent assignments of the training data by letting a training image assume multiple latent values. To this end, we relax the constraints on $W$ and allow $w_{z,t}$ to be real-valued in $[0, 1]$ and non-zero for more than a single assignment $z$. This can be motivated by the uncertainties of the assignments for the training images, in particular with an increasing quantization of $\mathcal{Z}$ where two elements of $\mathcal{Z}$ can become very similar. As we show in our experiments, this generalization makes the latent Hough transform less sensitive to the number of quantization levels, i.e., $|\mathcal{Z}|$.

### 5.3.2 Special Cases of the Latent Matrix

The most basic special case of the latent matrix is when $|\mathcal{Z}| = 1$ and $w_{z,t} = 1, \forall t \in \mathcal{T}$ which is equivalent to the original Hough transform formulation in Eq.(5.1). Another interesting special case is where all $w_{z,t}$ are uniformly set to a non-zero value in which case the detector is again equivalent to the original Hough transform since the response of every row is identical. If $W$ is a diagonal square matrix with the number of rows equal to the number of training images $|\mathcal{Z}| = |\mathcal{T}|$, and all elements on the diagonal are set to a non-zero constant, the model transforms into an exemplar detector similar to [Malisiewicz et al. 2011] where the responses of different training data compete with one another.
Other splitting of the training data between the uniform and exemplar cases, by manual annotations or unsupervised clustering, are also other special cases of the latent matrix where each row of the matrix represents one cluster.

For splitting the training data, we have considered manual annotations of the viewpoints and two popular methods for clustering. Namely, agglomerative clustering of the training images based on their similarity and k-means clustering of the aspect ratios of ground truth bounding boxes. Similar to [Zhang and Chen 2010, Razavi et al. 2010], we define the similarity of two training images as the $\chi^2$ distance of their occurrence histograms. Groupings of the training data based on the similarity measure and manual view annotations are visualized in Figs. 5.1(a) and 5.3(b) respectively. As shown in these figures, the groupings based on annotations or clustering might be visually very meaningful. However, as we illustrate in our experiments the visual similarity alone may not be optimal for the detection. In addition, it is not clear what similarity measure to choose for grouping and how to quantize it. These problems underline the importance of learning the optimal latent matrix for detection.

## 5.4 Learning the Latent Matrix

Learning the latent matrix to optimize the detection performance is very challenging. First, the number of parameters to be learned is proportional to the number of training images in the codebook which is usually very large. Another problem is that to be faithful to the greedy optimization in [Barinova et al. 2010], with every update in the weights, one needs to run the detector on the whole validation dataset in order to measure the performance.

In practice, we make an approximation and deal with the detection problem by sampling a sparse set of hypotheses from the validation set assuming that the position of the detections remain the same. To this end, we run the detector on the validation set once and collect a large number of false positives and true positives, which we denote by $R = \{(h, y)\}$. The variable $y \in \{0, 1\}$ indicates whether this sample is a true positive ($y = 1$) or a false positive ($y = 0$). To increase the number of positive
5.4. Learning the Latent Matrix

Figure 5.1: (a) Visualization of the $\chi^2$ similarity metric using Isomap. The two ellipses show the clustering of the training images of the 'Aeroplane' category of PASCAL VOC 2007. As can be seen the clustering splits the data into a very meaningful groups. (b) The importance of the first twenty intrinsic dimensions. The visualization is accurate since the first two dimensions cover most of the variation in the data.

hypotheses, we also generate object hypotheses from the positive training examples. For each hypothesis $h$, we pre-compute the contribution of every training image to that hypothesis, i.e., $V(h|t)$ (5.5). It is actually the linearity of the Hough transform based approaches in Eq.(5.3) that allows this pre-computation, which is essential for learning the latent matrix $W$ in reasonable time.

We formulate the problem of learning the latent matrix such as to maximize the objective function

$$\hat{W} = \arg\max_W O(W, R). \quad (5.6)$$

where our objective is the average precision calculated as

$$O(W, R) = \frac{1}{\sum_m y_m \sum_{d,y_d=1} \sum_m \mathbb{S}(h_t, h_m)} \sum_{d,y_d=1} \sum_m \mathbb{S}(h_t, h_m) \quad (5.7)$$

$$\mathbb{S}(h_t, h_m) = \begin{cases} 1 & \text{if } V(h_t|W) \geq V(h_m|W) \\ 0 & \text{otherwise} \end{cases} \quad (5.8)$$

The objective function in Eq. 5.7 is non-convex and is not even continuous and thus it is not possible to optimize it with a gradient-based approach. For optimization, we used the Interacting Simulated Annealing (ISA) [Gall et al. 2007]. ISA is a particle-based global optimization
Algorithm 2 Interacting Simulated Annealing (ISA) [Gall et al. 2007] with cross-validation.

\[
\{R^s\} \leftarrow \text{sample}(R, \text{maxNeg}, \text{maxPos})
\]
\[
\epsilon \leftarrow 0.6
\]
\[
\text{for } p = 1 \rightarrow n \text{ do}
\]
\[
W_p \leftarrow \text{initialize } W \text{ at random}
\]
\[
\text{end for}
\]
\[
\text{for } \text{epoch} = 1 \rightarrow \text{maxEpochs do}
\]
\[
\{R^s\} \leftarrow \text{sample}(R, \text{maxNeg}, \text{maxPos})
\]
\[
c \leftarrow \text{getMaxPerturbations}(\text{iter, epoch}) \ // \text{adaptively reduce perturbation}
\]
\[
\text{for } \text{iter} = 1 \rightarrow \text{maxIter do}
\]
\[
W \leftarrow \text{perturb}(W, c) \ // \text{perturb } c \text{ elements of } W \text{ at random}
\]
\[
\text{for } p = 1 \rightarrow n \text{ do}
\]
\[
o_p \leftarrow O(R^s, W_p)
\]
\[
\text{end for}
\]
\[
\beta \leftarrow 20 \times (\text{epoch} \times \text{maxIter} + \text{iter})
\]
\[
\{W\} \leftarrow \text{selection}\{W\}, \{o\}, \beta, \epsilon
\]
\[
\text{end for}
\]
\[
\text{end for}
\]

method similar to the simulated annealing. Starting from an initial set of weights for \(n\) particles, it iteratively, i) perturbs the weights of selected particles ii) evaluates the objective value for each particle, exponentiate these values with the algorithm parameter \(\beta\), and normalizes them to create a probability distribution. iii) randomly selects a number of particles using this distribution. This process is continued until a strong local optimum is reached. The perturbation of \(W\) at each iteration is done by selecting a random number of elements \(c\) (maximum 10) and changing their weights randomly. \(c\) is adaptively decreased at each epoch by the factor \(\frac{1}{\sqrt{\text{epoch}}}\).

Algorithm 2 gives an overview of the optimization with ISA. To avoid overfitting effects due to the large number of parameters to be estimated, we run a cross-validation loop inside the global optimization. For cross-validation, we use a random subset \(R^s\) of \(R\) at each epoch. In practice we have kept all the positive examples and 5% of the negatives for training at each epoch. We have also found well performing solutions, in detection
5.5 Experiments

We have evaluated our latent Hough transform on two popular datasets, namely the Leuven cars dataset [Leibe et al. 2007] and the PASCAL VOC 2007 [Everingham et al. 2010]. As a baseline for our experiments, we compare our approach with the marginalization over latent variables by voting only for locations ("Marginal"), unsupervised clustering of aspect ratios ("AR clustering") and image similarities ("Similarity clustering"), and the manually annotated viewpoints ("View Annotations") provided for both Leuven cars and PASCAL VOC 2007.

In all our experiments, the codebook of the ISM is trained using the Hough forests [Gall et al. 2011] with 15 trees and the bounding boxes for a detection are estimated using backprojection. The trees are trained up to the maximum depth of 20 such that at least 10 occurrences are remained in every leaf. For performing detection on a test image, we have used the greedy optimization in [Barinova et al. 2010]. The multi-scale detection was performed by doing detection on a dense scale pyramid with a \(\frac{1}{\sqrt{2}}\) resizing factor. In addition, instead of penalizing the larger hypotheses by adding a negative bias as in [Barinova et al. 2010], similar to [Leibe et al. 2008], we allow larger deformations by increasing the standard deviation of the smoothing kernel proportional to the scale. The smoothing kernel is chosen as a Gaussian with \(\sigma = 1.25\) at scale one.

In every test, 40 bounding boxes are detected and the hypothesis score in Eq.(5.4) is assigned as the confidence of every detection. Standard precision/recall curves used for the evaluations and the average precision measure (AP) was calculated according to [Everingham et al. 2010].

Prior to learning the latent groups, we have collected a set of positive and negative detections by running the detector on the whole validation set of a category and detecting 100 bounding boxes from each image. The bounding boxes with more than 60% overlap with ground truth were considered as positive and the ones with less than 30% as negatives.

**Leuven cars:** For the Leuven cars dataset, 1471 cropped training images of cars are provided. The viewing angle is divided into 14 views and
Figure 5.2: This figure illustrates the result of using view annotations and unsupervised clustering for grouping training data of “Aeroplane”, “Bicycle” and “Sheep” categories of PASCAL VOC 2007. Groupings based on aspect ratio are shown in the first row, similarity clustering in the second, and the manual view annotations in the third row. Although clustering increases the performance for aeroplanes, it is reducing it for the “Sheep”. Also the AR clustering is performing better than similarity clustering for aeroplanes and bicycles, yet clustering similarities leads to better results for the sheep. By using four clusters, the results are deteriorating in all three categories which is due to the insufficiency of the number of training data per cluster.
Figure 5.3: (a) The performance comparison of our latent Hough transform with the baselines on the Leuven cars dataset. As can be seen in the red curve, AP is clearly increased by manually splitting the data to 14 views. By learning 14 latent groups, the performance is significantly improved over both baselines, the magenta curve. Learning the generalized latent matrix (green) and increasing the number of groups (cyan) improves the results further. (b) Examples of the 14 views manually annotated in the training data.

training images are annotated for 7 viewpoints. The training data of the other 7 viewpoints is obtained by mirroring the training images, creating the total of 2942 training images annotated for 14 viewpoints. Prior to the training, all positive objects in the training images are cropped and scaled to have the height of 70 pixels. In addition, for the background category, we are using the clutter set of Caltech 256 [Griffin et al. 2007b]. A third of positive images and 200 negative images and from each of which 250 patches are randomly sampled for training each tree. As the validation set for learning the latent groupings, the Amsterdam cars dataset [Leibe et al. 2007] and the Graz02 cars [Opelt et al. 2006] are used. The Leuven sequence [Leibe et al. 2007] is used as the test set and the detection was performed on 12 scales starting from 2.7.

PASCAL VOC 2007: A separate forest is trained for each category. The training is carried out by using all the positive examples and their mirrors in the “trainval” set of a category as the positive set and the images not containing the category as the negative set. The partial view annotations are ignored for the training. Similar to the cars, the positive training images are cropped and normalized to have the maximum height or width of 100 pixels. For training each tree, 200 training im-
ages from the positive set and 200 from the negative set and from each of which 250 patches are sampled at random. The “trainval” set of a category was used as the validation set for learning the latent groupings. The performance of the method is evaluated on the “test” set and in accordance with “competition 3” of PASCAL VOC Challenge. The multi-scale detection is done with 18 scales starting from 1.8.

To evaluate the benefits of the discriminative learning against unsupervised clustering and manual view annotations, we compared the results of learning on the Leuven cars and PASCAL VOC 2007 datasets. For a fair comparison, we train the Hough forests [Razavi et al. 2010] for a category only once and without considering the view annotations or learned groupings. For the optimization with ISA, we have used 500 particles and the number of epoch and iterations were both set to 40.

Figure 5.3 compares the performance of the learning with our baselines on the Leuven cars [Leibe et al. 2007]. Disjoint groupings of the training data based on view annotations, improves the result by 14% compared to the marginalization. By learning the latent groups the performance improves by 21% and 6% w.r.t the marginalization and manual view annotations respectively. Learning the generalized latent matrix improves the result further by about 9%. By allowing more latent assignments, one can learn finer groupings of the data and increase the performance to get 34% increase compared to the marginalization baseline.

The detection performance with the two clustering and view annotations on three distinct categories “Aeroplane”, “Bicycle” and “Sheep” of VOC’07 dataset are summarized in Fig. 5.2. As can be seen, by grouping the training images the detection performance may improve. However, this improvement is very much dependent on the grouping criteria, the category and the number of training data per group. For example in detecting airplanes, in Fig. 5.2, although using two clusters leads to a significant improvement over marginalization, clustering the data into more groups impairs the performance significantly. As another example, in detecting sheeps, the marginalization is clearly outperforming clustering with two cluster. The clustering or the view annotations do not lead to optimal groupings and even finding the well performing ones requires plenty of trial and error.

In contrast to clustering, one can group training data optimally by treating the groups as latent variable and learning them discriminatively.
5.5. Experiments

Figure 5.4: This figure shows the result of learning the latent matrix on three categories. By learning the latent matrix we can consistently outperform the clustering (“AR clustering”) and the Hough transform baseline (“Marginal”). (a) When learning 2 latent groups, there is not much benefit in assigning training examples to multiple groups. (b) However, doing so already gives a benefit for learning three latent groups as it models the uncertainty in the assignments. (c-d) Results for two other categories. (e) Shows the comparison of the training and testing performance as a function of the number of epochs. Since the same training data is used for creating the ISM codebook and learning the latent matrix, the overall performance of the training is much better. Yet, the two curves correlate well and the training shows little overfitting.
Table 5.1: Detection results on the PASCAL VOC 2007 dataset [Everingham et al. 2010]. The first block compares the performance of the Hough transform (HT Marginal), Hough transform with aspect ratio clustering (HT + AR), Hough transform with view annotations (HT + View) and our proposed latent Hough transform (LHT Ours). As can be seen the clustering improves the results for 14 categories over the marginalization but reduces it for the other 6. Yet, by learning latent groups we outperform all three baselines on most categories and perform similar or slightly worse (red) on others. The comparison to the state-of-the-art approaches is shown in the second block. We outperform the best previously published voting-based approach (ISK [Zhang and Chen 2010]) in mAP. Our performance is competitive on most categories with the best performing sliding window approaches [Vedaldi et al. 2009, Song et al. 2011] and the latent part model of [Felzenszwalb et al. 2009] and is state-of-the-art on two categories (green).
Figure 5.5: Some qualitative results on the test set of PASCAL VOC 2007 database. Ground-truth bounding boxes are in blue, correctly detected boxes in green and false positives in red.
5.6 Discussions

In this chapter, we have introduced the latent Hough transform to enforce consistency among votes that support an object hypothesis. To this end, we have augmented the Hough space with latent variables and discriminatively learned optimal latent assignments of the training data for object detection. To make effective use of the training data, we have generalized the model by allowing the training images to have multiple assignments. Although we have focused on learning latent assignments of the training data, the model can be used in a more general context, e.g., learning a multi-class latent Hough transform model or learning latent transformations of the votes for a better detection accuracy.

Learning a latent Hough transform model is equivalent to learning a latent representation of the *structure* of object instances, in other words how the overall appearance of objects varies across instances. Note that this latent representation may not encode all appearance variations but the space of valid combinations of training patches. In this respect, the visual vocabulary and this latent representation provide complementary information about appearance and structure of a category. Although the appearance of object instances can vary greatly due to lighting, coloring, etc. the structure of the objects tends to vary less and thus can be shared among many appearances. The non-parametric description of appearances with patches and the parametric representation of the structure with latent variables can be extended to also include scene context, such as geometry, type, etc.
6

Beyond Hough Transform

In the previous chapter, we have shown how the consistencies among object parts can be enforced by voting in a latent space. In principle, any variation in the positive training data can be modeled by this formulation. Yet, the latent Hough transform model is only efficient for modeling global variations/transformations due to viewpoint, coloring, lighting etc. Many intra-class variations in an object category such as deformations or articulations are inherently local and modeling the joint variation leads to a combinatorially large space which is neither efficient to learn nor easy for inference at test time.

Previous works had addressed this issue by taking a local approach and modeling pieces/parts of an object category instead. In particular, two distinct approaches have been proposed in the literature.

The first approach is the pictorial structures [Fischler and Elschlager 1973] and it variants [Yuille 1991]. Each object category in these works is divided into a fixed number of (semantic) parts and a classifier is trained for modeling the appearance of each part instead of the whole object. In addition a Markov Random Field is imposed on top of the parts to model the spatial relations among parts. The hope here is that the variation in appearance of each part is limited, e.g. to local rotations and scaling, and thus it is easier to learn an appearance model for each part than the whole object. In addition, since the total number of parts is fixed and is much smaller than the number of patches/pixels, an explicit prior model for deformations can be learned from training data and efficiently used for inference.

However, the explicit model has the disadvantage that the parts and their configurations are class-specific and it is difficult to scale these
approaches to a large number of classes. In addition, as a parametric approach the unusual examples are not properly modeled in these models, an effect of which can be seen in the precision/recall curves of these methods (e.g. see [Felzenszwalb et al. 2009]) with a sharp decrease of precision at a certain recall. Since the number of parts in these models is fixed, there is also a question of what the system should do in case of occlusions. Some recent works on image grammars [Girshick et al. 2011] try to address this issue by introducing an occluder to the model although automatic modeling of several occlusions and learning such models remains a challenge [Enzweiler et al. 2010, Tang et al. 2012].

The second direction of research has been focused on bottom-up grouping of image patches using either segmentation [Fulkerson et al. 2009] or edges [Karlinsky et al. 2010, Yarlagadda et al. 2010]. The classification or voting in these works is performed for a group of patches instead of individual patches. The reasoning behind the grouping is that each group will have much more information than the individual patch and thus can cast a stronger vote for the position or label of the object category [Ommer and Buhmann 2007]. Further, since the number these groups is substantially less than the number of patches, a more sophisticated deformation model (like the constellation instead of star-shaped) can be used which is an advantage.

Yet, the bottom-up grouping of the patches/pixels is a non-trivial task. It is also not clear at all what criteria to use for the grouping. In a way, this grouping stage is also harder than the original task since the spatial extent of the edges/segment must be determined accurately first. The hard grouping of the patches is very sensitive to the mistakes made by the early vision pre-processing as well. The number of patches and the spatial extent of each group can also greatly vary depending on the length of the edge or size of the superpixel segment. This limits the appearance modeling in these works to a simple bag of words model and brings about a variety of normalization issues.

In this chapter, we aim at overcoming these limitations by introducing an extension to the Latent Hough Transform model that includes local latent variables. This formulation overcomes the limitations of the previously mentioned approaches in that it neither requires the number of parts to be fixed nor does bottom-up grouping of patches.
The rest of this chapter is organized as follows. The new formulation with local latent variables is described in section two. Section three discusses a special case of the local latent formulation where the latent variable is the self-similarity of a patch. The result of experiments with the sparsity potentials is presented in the next section before the chapter is concluded. This work is still at its early stages and the experiment are limited to very simple settings.

6.1 Detection with Local Latent Variables

If we recall from Chapter 5, in the Latent Hough Transform formulation each feature $f_i$ casts votes $V(h, z|t, f_i)$ for the hypothesis $h \in \mathcal{H}$ and the latent values $z \in \mathcal{Z}$ given the training image $t \in \mathcal{T}$. In this formulation these votes only depend on the appearance of the feature $f_i$. Although voting in the latent space ensures the global higher-order consistency of the patches in the assignment of the latent variables, as mentioned earlier, many changes in appearance of the objects, like color, texture, or orientation are inherently local. Modeling these local changes with a global variable is very expensive in data.

It is to this motivation that we propose to augment the LHT with local latent variables. Unlike the global latent space $\mathcal{Z}$ that is the same for all features $f_i \in \mathcal{F}$, the local latent space $\mathcal{Z}^i$ is specific to each feature $f_i$ and is used to enforce local consistency in the appearance of the patch and eventually condition the votes of every feature for the global variable.

When using local latent spaces, the score of the hypothesis is obtained by

$$S(h) = \max_{z \in \mathcal{Z}} \sum_i \sum_t V(h, z|t, I_i, \hat{z}^i)$$

(6.1)

where $\hat{z}^i$ denotes the most likely local latent variable for the feature $f_i$. Using the appearance of the patches in the neighborhood $\mathcal{N}^i$ of $f_i$, $\hat{z}^i$ is obtained by

$$\hat{z}^i = \arg \max_{z^i \in \mathcal{Z}^i} p(f_i|\mathcal{N}^i, z^i).$$

(6.2)

Note that the fact that a latent space is assigned to every feature does not necessarily mean that $\mathcal{Z}^i$ is different for every feature, although in...
6. Beyond Hough Transform

(a) In a Latent Hough Transform (LHT) model, all image patches vote for the hypotheses $h \in \mathcal{H}$ and the latent values $z \in \mathcal{Z}$ which enforces global consistency among votes. (b) In this chapter we propose an extension of this model that includes local latent variables. In this extended model, a latent space is learned for each voting patch $I_i$ using the appearance of the other patches in its local neighborhood (orange arrows). The latent assignment of a patch to the local latent variable, $z^i \in \mathcal{Z}^i$, is then used to condition the votes that $I_i$ casts to the global latent space, $V(z, h|I_i, z^i)$ (red arrows). As can been seen, in this formulations the groups of patches are not mutually exclusive and a patch can vote in the local latent space of more than one patch (e.g. both to $\mathcal{Z}_i$ and $\mathcal{Z}_j$). In addition, the number of these latent spaces is not fixed in contrast to the part-based models since a latent space is assigned to every single patch.

**Figure 6.1:** (a) In a Latent Hough Transform (LHT) model, all image patches vote for the hypotheses $h \in \mathcal{H}$ and the latent values $z \in \mathcal{Z}$ which enforces global consistency among votes. (b) In this chapter we propose an extension of this model that includes local latent variables.
6.2. Local Self-Similarity Potentials

principle it can be. Figure 6.1 gives a schematic comparison of the voting in Latent Hough Transform with or without local latent variables.

**Marginalization or maximization over local latent variables:** In the formulation of the Latent Hough Transform in Eq. (6.3), the score of the hypothesis is obtained by taking the maximum over the latent variables. As we have shown in Chapter 4 for joint voting, it is also possible to marginalize over the latent variables leading to

\[
S(h) = \max_{z \in Z} \sum_i \sum_t \sum_{z^i \in Z^i} V(h, z|t, I_i, z^i)
\]  

(6.3)

The question of whether marginalization or maximization should be used remains largely unclear for the general context. Yet, one of the contributions of the LHT formulation is that it allows for learning the latent variables and sharing training examples among them that makes the maximization robust. Our experiments also confirm this by showing how such an approach can always outperform marginalization.

Similar to the Latent Hough Transform, the question that comes to one’s mind is how should one select the latent space. A comparison of Eqs. (6.3 and (2.10) and (2.11) reveals that how the visual vocabulary used for detection can be interpreted as a local latent variable model. And taking the maximum likely word or marginalizing over all of them is related to the discussions above.

6.2 Local Self-Similarity Potentials

In this section, we show how the sparsity of the appearance of a patch in its neighborhood can be used as a local latent variable. We make the observation that the patterns in an image can range from simple textures to complex shapes. Likewise a local patch in an image can be an element of a textured region or a complex shape region. In the case that the local patch is a part of a textured region, by definition, we expect to see many similar patches in its neighborhood. In contrast, if this local patch is part of a complex structure in an image, it is unlikely for its neighboring patches to have similar appearances.

Based on the self-similarity of a patch with the patches in its local neighborhood, we define the sparsity of a patch as a measure of the uniqueness
The patches in an image exhibit different sparsity values. (a) Consider the two patches on the foot of the rider and the leaves of a tree. The images in (b) and (c) show the normalized cross correlation of these patches with the image. While the self-similarity of a non-texture patch like the foot to its neighboring patch is low, the patch on the tree is less sparse and much more similar to the patches in its neighborhood. Based on this observation, in this paper, we introduce a measure which captures the sparseness of a patch within its neighborhood and incorporate it as a “sparsity potential” for object detection.

of its appearance in its local neighborhood. We argue that this measure can be used to compactly summarize the information in an immediate context of a patch and thus increase its discriminative power. In particular, we make the following two observations

- the similarity in appearance of patches in a neighborhood of a central patch exhibit different sparsity values when the central patch appears on an object as opposed to a background region.

- the codebook entries associated with texture or simple edge patterns are consistently less sparse in their neighborhood as opposed to entries which are associated to more complex patterns (see Fig. 6.2).

Based on these observations, we utilize the sparsity of the local self-similarity as a local latent variable for each patch to increase discriminativity of local patches. Specifically, for every codebook entry a classifier is trained on the sparsity values to classify a patch. Calculating the self-similarities with Normalized Cross Correlation [Shechtman and Irani 2007] or image specific codebooks [Deselaers and Ferrari 2010] is computationally very expensive. To overcome this limitation, the lo-
6.2. Local Self-Similarity Potentials

cal self-similarities are efficiently calculated using the ISM codebook by assuming the patches with similar assignments to be self-similar.

We have evaluated our method on the challenging PASCAL VOC 2007 dataset [Everingham et al.] and show that the proposed sparsity measure substantially improves the results although coming at very little computational overhead. The proposed potential is also quite general and can be easily integrated in other Hough-transform based detectors or used in other domains.

6.2.1 Related Works to the Self-Similarity

In the recent years, several methods have been proposed for measuring saliency of image [Goferman et al. 2010, Valenti et al. 2009, Hou and Zhang 2007] to focus the attention on regions with distinctive patterns. Similarly a number of approaches have been proposed to detect salient key points in an image. These works are mainly used to extract sparse image features that are invariant under certain geometric transformations, e.g. scale [Kadir and Brady 2001, Lowe 2004] or affine [Mikolajczyk and Schmid 2004] invariance. Similar to those approaches, in this work, we are also aiming at measuring the sparsity of regions in an image to separate foreground from background. However, we differ from these approaches since firstly our sparsity measure is class-specific instead of a generic measure as in those works. Secondly, in our approach we use dense features and no non-sparse image region is discarded a-priori. Although detecting objects with sparse features can be quite fast, it has been shown previously [Fei-Fei and Perona 2005, Boiman et al. 2008] that superior performance can be achieved by dense sampling of features.

Learning the codebook of local appearances has been the subject of many investigations as well [Leibe et al. 2008, Boiman et al. 2008, Gall et al. 2011]. Although the codebook is obtained generatively by clustering local patch appearances in [Leibe et al. 2008], the class label and spatial distribution of the patches are used in [Gall et al. 2011] to discriminatively learn the codebook in a random forest framework. While these approaches cluster training patches, Boiman et al. [Boiman et al. 2008] take a non-parametric approach and directly retrieve the nearest training
 patches without performing any quantization. Learning the spatial distributions of a codebook entry is done generatively in [Leibe et al. 2008, Gall et al. 2011] whereas in [Maji and Malik 2009] and [Zhang and Chen 2010], a max-margin approach is taken to learn weights for codebook entries and training images respectively.

Our self-similarity based sparsity potential is also related to the self-similarity features. [Shechtman and Irani 2007] proposed the local self-similarity (LSS) feature as an appearance independent feature which encodes the spatial location of self-similar patches in a neighborhood of a patch. The LSS has subsequently been integrated in several image classification benchmarks and have shown to consistently improve classification accuracy (e.g. see [Gehler and Nowozin 2009]). This feature has been extended by [Deselaers and Ferrari 2010] as a global self-similarity measure. Our proposed sparsity potential can also be interpreted as a self-similarity feature. However, in our approach, the self-similarity is used to measure the sparsity of a patch in its neighborhood. To this end, this measure is used conditioned on the appearance of the central patch to classify a patch as foreground or background.

Several approaches have previously used the information around a single patch for stronger classification. Similar to our work, in [Shotton et al. 2008a] all patches are assigned to a codebook and a classifier is trained on the cluster assignment of neighboring patches to improve the classification performance. Similarly, in [Fulkerson et al. 2009] a bag of words model [Sivic et al. 2005, Csurka et al. 2004] is learned on neighboring superpixels for classification of a superpixel. In [Yarlagadda et al. 2010], according to certain proximity rules, the close patches have also been grouped together to cast more accurate votes for object detection. Yet, to the best of our knowledge, we are the first to propose the self-similarity sparsity of a neighborhood for object detection.

6.2.2 Sparsity Potentials

In this section, we show how the sparsity can be used as a local latent variable. To this end, we use a binary latent variable which assumes 1 if the patch is sparse in its neighborhood and otherwise 0. We base our sparsity or distinctiveness measure on self-similarity [Shechtman and
Let us assume that we have a metric that measures the similarity of a patch \( f_i \) with all patches in its neighborhood, \( \{ f_n | n \in \mathcal{N}^i \} \), e.g., NCC as in [Shechtman and Irani 2007]. Further, we assume that the returned similarity is normalized to be in \([0, 1]\) with 1 representing the most similar and 0 the most dissimilar patch. In this case, one is getting a real valued self-similarity vector \( \text{ss}_i = (ss_1, \ldots, ss_{|\mathcal{N}^i|}) \) where each element \( ss_n \) records the normalized similarity of \( I_n \) to \( I_i \).

The sparsity of the self-similarity vector \( \text{ss}_i \) can be measured in many different ways, e.g., by using entropy or various vector norms. In this work, we use the L1-norm,

\[
\| \text{ss}_i \|_1 = \sum_{n \in \mathcal{N}^i} |ss_n|
\]  

(6.4)

which is both simple and fast to calculate.

### 6.2.3 Calculating the Self-Similarity Efficiently

Measuring the self-similarity by cross correlation of patches in the feature space, as done in [Shechtman and Irani 2007], can be quite time consuming and is not appropriate for object detection. To efficiently calculate the self-similarities, Deselaers et al. [Deselaers and Ferrari 2010] proposes to cluster the patches in an image into an image specific codebook and assume that the patches assigned to the same codebook entry are self-similar. Although, this can be done faster, performing this on the image pyramid densely as required in our setup is quite challenging. However, since in detection with an ISM, the image patches are already assigned to the entries of a large codebook, these assignments can be directly used to calculate the self-similarities.

For each patch in the scale pyramid of a test image, a self-similarity score in Equation (6.4) needs to be measured. To this end, the features at one level are passed to the Hough Forest codebook [Gall et al. 2011], their leaf assignments are recorded and for each tree an assignment image is formed. Each pixel \( i \) in the assignment image records its matching leaf index, i.e., the codebook entry \( \omega_j \). For each image feature \( f_i \) with a leaf assignment \( \omega_j \), all the neighboring features that are also assigned
Figure 6.3: This figure evaluates the effect of the neighborhood size used for calculating the sparsity on the accuracy of the detector. (a) The comparison of the detection performance of our baseline [Razavi et al. 2010] with our proposed sparsity measure with various window sizes on the test set of ‘aeroplane’ category of the PASCAL VOC 2007 dataset. As can be seen, the proposed sparsity potential always improves the accuracy. The performance tends to increase with the window size until it saturates at around 71 pixels, almost doubling the average precision (AP) compared to the baseline. (b) The sparsity potential is calculated on a square neighborhood of every 16 × 16 patch.

to \(\omega_j\) are considered to be self-similar and their similarity is set to 1. The similarity of all other patches in the neighborhood is set to zero. Therefore, (6.4) can be written as

\[
||s_{si}||_1 = \sum_{k \in N_i} \delta_{\omega_j, \omega_k}
\]

where \(\delta_{\omega_j, \omega_k}\) denotes the Kronecker delta. Given the assignment image, all sparsity measures can be efficiently calculated by decomposing the assignment matrix based on unique leaf assignments.

6.2.4 Learning the Sparsity Classifiers

For training the sparsity classifiers, first a set of features on the validation set, both on objects as well as background, are extracted and are assigned
to one or more codebook entries $\omega_j$. Given a neighborhood function, the sparsity measure of every feature is calculated. Next, for each $\omega_j$ and class label $c$, these sparsity measures for different class labels are collected and used to learn a simple threshold $\theta_{c,\omega_j}$. These thresholds are then used to calculate the assignment of $z^i$ as

$$
z^i = \begin{cases} 
1 & \text{if } ||\mathbf{s}_i||_1 \leq \theta_{c,\omega_j}, \quad z^i = 1 \\
0 & \text{otherwise.} 
\end{cases}
$$

(6.6)

Using the sparsity measure as a single dimensional feature, the thresholds $\theta_{c,\omega_j}$ are learned such as to separate the positive and negatives with the best classification accuracy with zero false negatives on the training data.

### 6.3 Experiments

In the previous section we have proposed the sparsity potentials described how to detect objects with them. In this section, we evaluate the benefits of using these potentials for object detection. As a baseline for our comparisons, we use the class-specific Hough Forests [Gall et al. 2011] and all evaluations are done according to the rules of the ‘competition 3’ of PASCAL VOC 2007 detection challenge [Everingham et al.].

The trainings of the forests are done using the ‘trainval’ set of images. Prior to the training, all object bounding boxes together with a 10% around them are cropped and rescaled to the smallest possible box with a maximum width or height of at most 100 pixels and a least minimum width or height of 50 pixels. The boxes annotated with ‘difficult’ tag are removed from the training set and for every category the images that do not contain any object of that category are used as the negative set. 15 trees are trained for each class. For training each tree, 200 positive objects and 200 background images are chosen randomly and from each of which 250 $16 \times 16$ patches are extracted and represented with the 32 channels HoG-like features as in [Gall et al. 2011]. The training of each tree is continued until either the maximum depth of 20 is reached or less than 10 patches are left in a leaf.
The multi-scale detection on an image in the 'test' set is done by building an image pyramid of 18 scales; starting from an upscaled image of 1.8 times the original image size and a scaling factor of $\frac{1}{\sqrt{2}}$. For every detection image, 40 candidate objects are detected and their bounding boxes are estimated from backprojection [Razavi et al. 2010]. Similar to [Barinova et al. 2010], the non-maxima suppression is done by considering all features contributing more than 0.0005 to a detection as its support and removing the votes of all patches at its position from the scale pyramid.

For measuring the sparsity of a patch, the patches in a square neighborhood around it are considered 6.3(b). The parameters of the self-similarity classifiers are trained by first running the detector on cropped positive objects and all negative images in the validation set of a category. Then two histograms, one for positives and one for negatives, is created which records the sparsity values for every codebook entry.
6.3. Experiments

Table 6.1: The comparison of our detection results on the PASCAL VOC 2007 dataset [Everingham et al.]. The first block compares the performance of our proposed approach with sparsity potentials (HF + sparsity) to the Hough Forest baseline (HF baseline [Gall et al. 2011]). As can be seen, by using the sparsity potentials the performance has been substantially improved for most categories. The comparison to the state-of-the-art approaches is shown in the second block. We outperform the best previously published Hough transform based approach (ISK [Zhang and Chen 2010]) in mAP. The other two rows give a comparison of our approach compared to the latent part model (LSVM [Felzenszwalb et al. 2009]) and the best results of the PASCAL VOC Challenge (VOC best [Everingham et al.]).

Finally, a classifier is trained to separate the positives from negatives based on this value.

Figure 6.4 evaluates the performance, in Precision/Recall, of using our proposed sparsity potentials for detecting objects on the PASCAL VOC 2007 dataset. The full comparison of the proposed sparsity potential with our baseline and other state-of-the-arts methods for all categories is given in the Table 5.1. As can be seen, the proposed sparsity potentials significantly improve the detection performance on this challenging dataset. The Fig. 6.3, shows the effect of this neighborhood size on the detection performance. For all of these experiments, the sparsity is calculated on a square neighborhood size of $71 \times 71$ pixels.
6.4 Conclusions

In this chapter, we have generalized the Latent Hough Transform by introducing local latent variables for individual features. This approach has several advantages over the state-of-the-art local approaches where either the number of parts needs to be fixed in an explicit model or the patches must be grouped in a bottom-up procedure.

As a proof of concept for the idea of local latent variables, we have introduced the sparsity potentials for object detection with the Hough transform. We have shown that the distinctiveness of the appearance of a patch in its neighborhood can be a strong cue for object detection. Further, we have proposed to measure this distinctiveness by the L1-norm of the self-similarity vector of a patch to its local neighborhood. Based on this measure and using the validation set, we have discriminatively trained a classifier for each codebook appearance and used it to separate foreground from background patches at test time. The proposed detector with sparsity potentials substantially outperforms the baseline detector and leads to a comparable accuracy to the state-of-the-art detectors on many categories of the PASCAL VOC 2007 dataset.

In the future, it would be interesting to use this sparsity potentials in a multi-class setup to also discriminate classes from one another. Since the self-similar patches tend to belong to the same label, it would be also interesting to incorporate it as a higher order potential for image segmentation. Although we have used the simple L1-norm for sparsity, other popular measures (e.g. mutual information) can be used as well. Yet, efficient calculation of these measures remains a challenge.
Conclusions

This chapter concludes this thesis by an overview of our main contributions, discussion of the advantages and limitations of the proposed approach, and finally presenting some ideas for the future work.

7.1 Contributions

- **Scalable multi-class detection:** In this thesis, we give an in-depth complexity analysis of object detection using a shared vocabulary. Based on this analysis, we derive the necessary conditions for having a sub-linear detection complexity in the number of categories. In particular, we show that the scalability can only be achieved when the individual codewords are able to discriminate between classes. We empirically demonstrate this by training a multi-class detector with a very sub-linear computational complexity in the number of classes and show that training such a detector does not come at the cost of reduced accuracy.

- **Data-driven class hierarchies:** By using a shared vocabulary for multi-class detection, we implicitly let the appearance of local patches to be shared across different categories. The sharing structure reveals very interesting relations between the appearance of the categories at the patch level. We use this structure to create class taxonomies and use them to speed-up the detection time.

- **Patch-based description of an instance:** In the proposed multi-class detector and object hypothesis is detected by accumulating the votes of a set of consistent patches. In this thesis, we
show how these patches and their configurations can be used to describe an object instance. Based on this description, we further define a distance between two detections and use it to retrieve auxiliary information about a detected object and track an object instance in a video.

- **Latent Hough transform:** In order to enforce higher order consistency among patches we propose to augment the Hough transform with local and global latent variables. Our experimental results suggest that, given enough training data, the detection performance can substantially increase by using these latent variables. Yet, such improvements depend very much on the latent variable used and their quantizations. In this thesis, we give an optimization framework for discriminative learning of a latent space such as to directly improve detection performance. Moreover, we show how this formulation can be extended to local latent variables to further improve performance.

- **Generalized Latent Assignments:** In training a latent variable model, the values of the latent variables for training data needs to be estimated. Traditional approaches, use a hard assignment of the data points to latent variables. In this thesis, we generalize this notion and propose a real-valued soft memberships of data points to latent variables and represent these assignment using a latent matrix. This representation allows for modeling the uncertainty involved in these assignments and makes the model robust with respect to the number of latent variables.

7.2 Discussions

In this thesis, we have investigated a non-parametric patch-based approach for the task of multi-class visual object detection. In this approach, an object is represented as a mixture of relevant training patches. The patch-based representation of images is very flexible and powerful. Unlike the competing part-based representations, the patch-based representation leads to non-parametric models in which the relation between patches can be modelled implicitly. Similar to other non-parametric
models, there is no need to make strong assumptions about the distribution of the data which is an advantage.

As mentioned earlier, in many applications the object detection is only an input to the higher level tasks. The information stored in the training patches can thus be transferred to new instances or used to extract auxiliary information. The configuration of the consistent training patches can also provide a rich description of the objects for higher-level tasks such as action recognition or pose estimation.

Moreover, using training patches as the building block of recognition allows for the modeling of background in addition to the objects which can eventually lead to the recognition and interpretation of the whole scene and the interaction among its elements.

Common object detector, including the one presented in this thesis, assume a non-structured background model which is a major limitation. Modeling the background structure, whether geometric or semantic, can play a very important role in object detection. Firstly, objects in images do not appear in isolation and the background can be used to either guide the object detector [Heitz and Koller 2008] or serve for contextual priming of these detectors [Torralba and Sinha 2001]. Secondly, the background is also very structured and modeling it can help in disambiguating the final output of the detector.

Another important limitation of the patch-based approaches is their requirement for storing all training patches in the memory. Several previous approaches have proposed to prune redundant patches or use a parametric model instead. Although these works can significantly reduce the memory requirements, this gains usually comes at the cost of losing flexibility in modeling unusual examples.

7.3 Outlook

The presented work in this thesis can be extended in a number of ways:

**3D geometry:** When reasoning about the consistency of training patches for explaining a test image, it will be also very helpful to reason about the geometry of the combinations and the physical constraints they impose. This direction goes more towards the works in [Gupta et al. 2010]
and [Hoiem et al. 2006]. However, the main difference will be that the geometric interaction of patches needs to be handled locally and not globally, and by incorporating them as local latent variables in the model.

**Cognitive Feedback Loops:** The proposed detection approach is also limited in that it is a feed-forward method and does not make use of cognitive feedback loops. The cognitive feedback loops can be particularly useful for combining multiple cues, however efficient.

**Stronger Object Models:** In this thesis, we have shown how the latent variables can be used to enforce higher order consistency of patches. To this end, we have introduced the local and global latent variables. Yet, our experiments are limited in that we have only used three layers of latent variables and trained each layer independently. In the future, it will be interesting to jointly learn many more layers to arrive at a stronger object model.

**Multi-Scale Models:** The current model is a single-resolution and scale-independent model. For performing detection at multiple scales, the images are rescaled at multiple levels of the scale pyramid and the detection is performed independently at each scale. In the future, it would be beneficial to learn a multi-resolution model similar to [Benenson et al. 2012]. On the one hand, this can help to speed up the detection process and the features need to be extracted only at a single scale. On the other hand, as shown in [Park et al. 2010] the objects at multiple resolutions can have different appearances due to the perspective changes and a multi-scale approach can lead to better performances.
Own Publications


List of Figures

1.1 How many animals are in this photograph? Where are they located? From which species? Can you describe how they look like? As humans, we are able to instantly answer many such questions about an image. Despite recent progress, performing these tasks using computer vision remains largely an unsolved problem. In this thesis, we refer to these tasks as visual recognition and investigate a non-parametric approach for solving it. 

2.1 This figure shows three popular methods for parametrizing the location of an object in an image. (b) An object can be parametrized by the position and scale of a reference point, e.g., center of mass. (c) A bounding box, (d) pixel level segmentation. Although a hypothesis is more accurately described by using more parameters, providing ground truth data becomes more difficult and expensive. A good trade-off is usually found by using bounding boxes [Deselaers and Ferrari 2011].

2.2 This figure shows the detector introduced [Dalal and Triggs 2005]. This detector is an example of detection on template matching. In the case of this detector, a template is learned based on Histogram of Oriented Gradients (HOG) features using a linear SVM. (c-d) Positive and negative weights for different features in the template. The picture is reprinted from [Dalal and Triggs 2005] with permission.
2.3 Features of different object classes can share appearance although they do not necessarily also share their location. For instance, the legs of a person and a horse share both appearance (bounding boxes) and location (arrows) whereas the wheels of a bus and car are similar in appearance but not in location (red/blue arrows). Although the location is used as the main source of discrimination in template matching methods, the appearance is used as the only cue in the bag-of-words methods by discarding the location for detection.

2.4 Complexity in appearance of a local patch, and thus its discriminative power, can be altered by varying its relative size/resolution to the object. Using small regions of an image only contain basic edge information and are not alone informative for either a possible object’s position (localization) or its class label (classification). Yet, by increasing the size of a feature its discriminative power rapidly increases leading to more precise predictions even specific to an instance or situation.

2.5 Two examples of pictorial structure models. (a) The face model of [Fischler and Elschlager 1973] with loops. (b) The tree structured pictorial structure of [Felzenszwalb and Huttenlocher 2005]. The figures are re-printed with permission.

2.6 Overview of Spatial Pyramid Matching as an extension of bag-of-words models. The first level of the spatial pyramid is a bag-of-words with no location information. By dividing the hypothesis bounding box into spatial regions in subsequent levels the structure of the object is captured in more and more rigid models. The figure is reprinted from [Lazebnik et al. 2006].

3.1 Features of different object classes can share appearance although they do not necessarily also share their location. For instance, the legs of a person and a horse share both appearance (bounding boxes) and location (arrows) whereas the wheels of a bus and car are similar in appearance but not in location (red/blue arrows).
3.2 Complexity in appearance of a local patch, and thus its discriminative power, can be altered by varying its relative size/ resolution to the object. Using small regions of an image only contain basic edge information and are not alone informative for either a possible object’s position (localization) or its class label (classification). Yet, by increasing the size of a feature its discriminative power rapidly increases leading to more precise predictions even specific to an instance or situation. \ \ \ \ \ \ \ \ \ \ \ \ \ 33

3.3 Prior to training a visual vocabulary with the multi-class Hough Forests, a set of background and foreground training images are collected. The foreground images are also assigned a label determining their class/viewpoint (coded by color). For training every tree, a set of features are extracted from the training images $t \in T$ and are recursively split such as to optimize the discriminative power of the resulting nodes. The leaves of the forest are treated as codewords $\omega_j$ and store the occurrences of the training data and their offset $l^t_i$, class label $c^t_i$ and auxiliary variables $\theta^t_i$. The visualized occurrences for every leaf node are the actual patches in the trained vocabulary. \ \ \ \ \ \ \ \ \ \ \ \ \ 43

3.4 Patches clustered in two sample leaves of our multi-class detector trained for VOC’06 dataset. Although the appearance of our middle complexity patches is shared among some classes, it is yet discriminative to other classes. For instance, when a patch is assigned to leaf (a), it is difficult to determine if it belongs to a dog or a person but we can say that it is not coming from a car or a motorbike. This property of middle-complexity features enable building of part-based category hierarchies introduced in Sec. 3.4.3. \ \ \ \ \ \ \ \ \ \ \ \ \ 44
3.5 In this figure, the effect of training with various optimization criteria is illustrated. (a) When training according to (3.8) and (3.9). (b) By relaxing the location with (3.10), more distinction is appearing between dissimilar classes but the separation of background class is problematic. (c) If we only separate the foreground and background using (3.11), the background gets well separated but the classes remain mixed. (d) This is the sharing matrix obtained from (3.14) for which the classes and background are both well separated. As can be seen, the classes are separated similar to (b) and the foreground and background similar to (c). (best viewed in color) ........................................ 46

3.6 Sharing matrices for VOC’06 dataset using our proposed multi-class criteria in Eqs. (3.10) and (3.14) for training. (a) Sharing matrix of feature appearances using (3.15) and its corresponding taxonomy which is automatically obtained by clustering the sharing matrix. When calculating sharing of both appearance and location with (3.16), the sharing matrix in (b) is obtained. When discriminativity in appearance is combined with location, our multi-class training leads to a very good separation of classes. (c) Speed-up achieved using the taxonomy in (a). The more discriminative features lead to more speed-up. (best viewed in color) ........................................ 49

3.7 This figure gives an overview of the multi-class detection with a verification classifier. ................................. 51

3.8 The sharing matrix of appearance and its automatically built taxonomy for the VOC’07 dataset. (best viewed in color) ......................................................... 52

3.9 Our scalable method shows a sub-linear growth in the size of codebook (a) and in the number of votes (b) as opposed to linear in one-vs-the-rest. The taxonomy further reduces the number of votes. (20* uses VOC’07) ............................. 54
4.1 Object detection using Implicit Shape Models: Features are matched against the codebook $\mathcal{C}$ casting votes to the voting space $\mathcal{V}$. The local maxima of the voting space is localized and the votes contributing to it are identified (inside the blue circle). The contributing votes are backprojected to the image domain creating the backprojection mask. At right, visualization of the backprojection mask blended with the image. Note that the mask does not include the area occluded by a pedestrian. The image is taken from UIUC cars dataset [Agarwal and Awan 2004]. 60

4.2 In this chapter we propose a two stage approach for estimating auxiliary variable like view point of a detection. (a) The object instances in an image are detected by running the detector on the image. (b) Using their support, the most similar examples of a set of view annotated objects are retrieved. (c) The view point is estimated by transferring the view of the nearest examples and based on a single image and without using any motion model. 62

4.3 In order to track an instance of a class, like a certain person from the class pedestrians, we adapt on-line a class-specific codebook to the instance. This has been done by creating an instance model using its supporting votes. (a) Blue box indicates the instance of interest. (b) Voting image obtained by an off-line trained codebook for pedestrians. (c) Voting image obtained by the proposed instance-specific codebook. Despite the presence of other pedestrians and the confusing background, the peak at the center of the instance is clearly visible and votes for background objects are reduced. 63

4.4 Some example cars from the UIUC dataset together with their backprojection masks. Although only bounding box annotations are used for training, the backprojection can lead to a relatively accurate segmentation mask; already telling us about the occlusions. 69

4.5 The backprojection of the votes to the image domain according to the training image id (left). The visualization of the backprojected masks $\mathcal{M}_t$ for the first four most contributing training examples. 70
4.6 The probability $p(v_{j,occ}|\tau)$ for 2500 occurrences in the vocabulary. The probability is on-line estimated by Equation (4.13) and gives an instance-specific signature that can be used to improve tracking, see Fig. 4.3. While occurrences with probability $> 0.5$ are specific to the instance $\tau$, occurrences with probabilities $< 0.5$ vote for non-instance hypotheses. The estimates shown are from two sequences after 100 frames. The occurrences with probability equal to 0.5 are mainly patches that have not been activated during tracking.  

4.7 On-line adaption of the codebook. (a) After updating the particles, the multi-modal posterior distribution is approximated. The weights of the particles are indicated by color (yellow: high, red: low). The target is marked by a blue dot. (b) Based on the posterior, the voting space is clustered (blue: foreground, red: background, green: uncertain). Note that the intensity of the background (red) has been increased for a better visibility. (c) Votes that contributed to the detected local maxima are used to update the instance-specific statistics. Note that there is no local maximum for the most left person of the foreground (blue).  

4.8 (a) After a hypothesis is detected, its support is backprojected to the image domain to estimate its bounding box. (b) Estimating the bounding boxes from backprojection (Multi-Aspect Ratio) leads to more accurate estimation than using average training aspect ratio (Average Aspect Ratio).  

4.10 (a) Joint voting achieves better results than separate voting because of insufficient training data per view and finely quantized views although it does not use view annotations. Estimating the bounding box from backprojection even leads to a small improvement. (b) Sharing of features across views of motorbikes. (c) Performance comparison to state-of-the-art multi-view detectors for the motorbikes dataset. 79

4.11 The effect of training size on the performance of joint voting, separate voting, and a battery of independent one-vs-the-background classifiers: With the abundance of training data per view, the separate-voting works best. The advantage of sharing is significant with lower number of training examples, especially compared to the separate voting with an identical codebook although no view annotations are used. Joint and separate voting outperform the independent detector in efficiency and/or accuracy. 81

4.12 Some typical detections of our multi-view detectors on Leuvencar test sequence. The ground truth annotations are drawn in blue and false detections and correct detections are in red and green respectively. 82

4.13 Some example detections of our multi-view detector on motorbikes dataset. Similar to 4.12, the ground truth annotations are drawn in blue and false detections and correct detections are in red and green respectively. 83

4.14 Sharing in appearance and location among views for the motorbikes and cars as calculated in 3.16. As can be seen, the position and appearance of patches in both categories is shared across views. Due to the rapid change in appearance in the frontal and back views, these views share less with the side views. 84
4.15 (a) Viewpoint retrieval with the nearest neighbor using (4.9) (b) Moderate improvement by retrieving views by majority voting of 35 nearest neighbors. As can be seen, the nearest neighbor works already very good and the confusion appears to be limited to nearby views. In particular, comparing the sharing pattern in Fig. 4.14(a) and view confusions is interesting; e.g. front and back views share many features but they have been separated well for view retrieval. This shows the presence of additional information in the support that is exploited.

4.16 (c) View-point classification for detected cars in the VOC’06 dataset. As can be seen, the confusion appears to be limited only to similar views. (d) Comparison to state-of-the-art [Su et al. 2009, Sun et al. 2009] for view classification on VOC’06 (for a fair comparison detections up to 43% recall are considered; average accuracy 82%). Note that our nearest neighboring approach leads to superior performance and more balanced estimation. Comparing the sharing pattern and view confusions is also interesting; e.g. front and back views share many features but their view have been separated well. This shows the presence of additional information in the support.

4.17 View point estimation using nearest neighbor retrieval for two frames of the Leuven sequence. The nearest neighbors are shown in the right column.

4.18 (a-b) The view retrieval performance using (4.9) together with the proportion of the cars detected depending on the amount of occluded regions for a subset of the Leuven-car sequence. (the last two sets, 50% and > 50%, have very few instances). The recall and view retrieval performances were calculated independently for each occlusion set. Interestingly, although the detection performance deteriorates from large occlusions (a), the viewpoint retrieval performance is affected very little (b) which shows robustness of this similarity measure to occlusions.
4.19 Distance between the ankles estimated by the median of the $k$ nearest training images using (4.6) compared to mean and median as baselines. The estimation is robust even at high recall rates. .......................... 88

4.20 Two test detections from TUD-pedestrians dataset and their top ten nearest training examples (top row; nearest examples ordered from left to right) and backprojections of detection support to them (bottom row) using (4.6). The blue box shows the estimated bounding box from the backprojection mask (blended). Note the similarity of the poses between the test instances and retrieved nearest training images. .......................... 89

4.21 Tracking accuracy for the i-Lids sequences [Branch] over time: (a) easy; (b) medium; (c) hard. Mean and standard deviation are given in Table 4.1. On the easy sequence, the class-specific codebook and the on-line adaption perform well. In sequences medium and hard, the scene is more crowded, see Fig. 4.23 for an example image. This is a situation where on-line adaption outperforms the class-specific codebook. Taking the confidence into account for clustering improves slightly the accuracy compared to the update-always-strategy. .......................... 90

4.22 Tracking accuracy for the sequence S2.L1 (view 1) of PETS09 [Ferryman et al.]. Neither the off-line codebook nor the update-always-strategy are able to track the person since they are misled by other pedestrians in the scene (a). The proposed approach tracks the target (blue box) despite of occlusions and similarity with other instances. Two frames are shown in (b) and (c). Note that the accuracy is zero for a few frames where the strongest mode of the multi-modal posterior distribution does not overlap with the target, see Fig. 4.7 (a). .......................... 91
4.23 From top to bottom. Standard test sequences: i-Lids hard [Branch] (surveillance, crowded scene); David Indoor [Ross et al. 2008] (changing lighting conditions); Girl [Birchfield 1998] (fast movements, 360 degree head rotation); Occluded Face [Adam et al. 2006] (occlusions); Occluded Face2 [Babenko et al. 2009] (strong occlusions). YouTube test sequences: Klaus Kinski 1971 (camera zoom, 14166 frames); David Attenborough Night (very dark); Desert (strong shadow); Walrus (fast movement); Top Gear (multiple faces); Top Gear Ariel Atom (face deformations); Monty Python Military (multiple faces); Florian Silbereisen (fast movements, low resolution). The last two are from public street parades (occlusions, multiple faces, low resolution, difficult lighting conditions).

5.1 (a) Visualization of the $\chi^2$ similarity metric using Isomap. The two ellipses show the clustering of the training images of the ‘Aeroplane’ category of PASCAL VOC 2007. As can be seen the clustering splits the data into a very meaningful groups. (b) The importance of the first twenty intrinsic dimensions. The visualization is accurate since the first two dimensions cover most of the variation in the data.

5.2 This figure illustrates the result of using view annotations and unsupervised clustering for grouping training data of “Aeroplane”, “Bicycle” and “Sheep” categories of PASCAL VOC 2007. Groupings based on aspect ratio are shown in the first row, similarity clustering in the second, and the manual view annotations in the third row. Although clustering increases the performance for aeroplanes, it is reducing it for the “Sheep”. Also the AR clustering is performing better than similarity clustering for aeroplanes and bicycles, yet clustering similarities leads to better results for the sheep. By using four clusters, the results are deteriorating in all three categories which is due to the insufficiency of the number of training data per cluster.
5.3 (a) The performance comparison of our latent Hough transform with the baselines on the Leuven cars dataset. As can be seen in the red curve, AP is clearly increased by manually splitting the data to 14 views. By learning 14 latent groups, the performance is significantly improved over both baselines, the magenta curve. Learning the generalized latent matrix (green) and increasing the number of groups (cyan) improves the results further. (b) Examples of the 14 views manually annotated in the training data.

5.4 This figure shows the result of learning the latent matrix on three categories. By learning the latent matrix we can consistently outperform the clustering (“AR clustering”) and the Hough transform baseline (“Marginal”). (a) When learning 2 latent groups, there is not much benefit in assigning training examples to multiple groups. (b) However, doing so already gives a benefit for learning three latent groups as it models the uncertainty in the assignments. (c-d) Results for two other categories. (e) Shows the comparison of the training and testing performance as a function of the number of epochs. Since the same training data is used for creating the ISM codebook and learning the latent matrix, the overall performance of the training is much better. Yet, the two curves correlate well and the training shows little overfitting.

5.5 Some qualitative results on the test set of PASCAL VOC 2007 database. Ground-truth bounding boxes are in blue, correctly detected boxes in green and false positives in red.
6.1 (a) In a Latent Hough Transform (LHT) model, all image patches vote for the hypotheses \( h \in H \) and the latent values \( z \in \mathcal{Z} \) which enforces global consistency among votes. (b) In this chapter we propose an extension of this model that includes local latent variables. In this extended model, a latent space is learned for each voting patch \( I_i \) using the appearance of the other patches in its local neighborhood (orange arrows). The latent assignment of a patch to the local latent variable, \( z^i \in \mathcal{Z}^i \), is then used to condition the votes that \( I_i \) casts to the global latent space, \( V(z, h|I_i, z^i) \) (red arrows). As can been seen, in this formulations the groups of patches are not mutually exclusive and a patch can vote in the local latent space of more than one patch (e.g. both to \( \mathcal{Z}_i \) and \( \mathcal{Z}_j \)). In addition, the number of these latent spaces is not fixed in contrast to the part-based models since a latent space is assigned to every single patch. . . . . . . . . . . 116

6.2 The patches in an image exhibit different sparsity values. (a) Consider the two patches on the foot of the rider and the leaves of a tree. The images in (b) and (c) show the normalized cross correlation of these patches with the image. While the self-similarity of a non-texture patch like the foot to its neighboring patch is low, the patch on the tree is less sparse and much more similar to the patches in its neighborhood. Base on this observation, in this paper, we introduce a measure which captures the \textit{sparseness} of a patch within its neighborhood and incorporate it as a “sparsity potential” for object detection. . . . . . . . . . . 118
6.3 This figure evaluates the effect of the neighborhood size used for calculating the sparsity on the accuracy of the detector. (a) The comparison of the detection performance of our baseline [Razavi et al. 2010] with our proposed sparsity measure with various window sizes on the test set of ‘aeroplane’ category of the PASCAL VOC 2007 dataset. As can be seen, the proposed sparsity potential always improves the accuracy. The performance tends to increase with the window size until it saturates at around 71 pixels, almost doubling the average precision (AP) compared to the baseline. (b) The sparsity potential is calculated on a square neighborhood of every $16 \times 16$ patch.

6.4 The precision recall curves for some categories of the PASCAL VOC 2007 [Everingham et al.]. As can be seen, the proposed sparsity potentials substantially improve the detection performance. The average precision (AP) is calculated by the integral under the curve.
3.1 Complexity of different methods for object detection in the number of classes. For [Fidler and Leonardis 2007, Fidler et al. 2010], the constellation model of contour features is used. Although both our method and [Fidler et al. 2010] are sub-linear, [Fidler et al. 2010] uses coarse-to-fine representation of contour features at fixed locations which leads to accuracy loss.

3.2 Performance comparison of our multi-class method (MC) with the baselines in average-precision for VOC’06 dataset. The first block shows the detection without verification and without non-maxima suppression. MC outperforms one-vs-the-rest. The taxonomy not only reduces the amount of voting (Fig. 3.9), it also gives a slight improvement. In the second block, verification is performed with [Felzenszwalb et al. 2009]. By using a two-stage method, we are not loosing accuracy compared to [Felzenszwalb et al. 2009]. The number of performed verifications is given in Table 3.3.

3.3 Our multi-class detection approach reduces the number of windows for verification per image by three orders of magnitude. Unlike sliding window approaches, our method assigns a class to each returned window. This eases the verification as one classifier should be evaluated per window, and this without compromising accuracy; see Tables 3.2 and 3.4.
3.4 Performance evaluation on VOC’07 dataset in average precision (AP). Our scalable method with verification (MC+verif.) achieves very similar results to the baseline [Felzenszwalb et al. 2009] although using LSVM on significantly lower number of subwindows. ............................................. 57

4.1 Mean and standard deviation of the tracking accuracy. . . 93

4.2 Since votes with zero probability are not cast, voting with on-line adaption is 2.8 times faster than voting with the class-specific codebook. ................................................................. 93

5.1 Detection results on the PASCAL VOC 2007 dataset [Everingham et al. 2010]. The first block compares the performance of the Hough transform (HT Marginal), Hough transform with aspect ratio clustering (HT + AR), Hough transform with view annotations (HT+View) and our proposed latent Hough transform (LHT Ours). As can be seen the clustering improves the results for 14 categories over the marginalization but reduces it for the other 6. Yet, by learning latent groups we outperform all three baselines on most categories and perform similar or slightly worse (red) on others. The comparison to the state-of-the-art approaches is shown in the second block. We outperform the best previously published voting-based approach (ISK [Zhang and Chen 2010]) in mAP. Our performance is competitive on most categories with the best performing sliding window approaches [Vedaldi et al. 2009, Song et al. 2011] and the latent part model of [Felzenszwalb et al. 2009] and is state-of-the-art on two categories (green). ................................................................. 110
6.1 The comparison of our detection results on the PASCAL VOC 2007 dataset [Everingham et al.]. The first block compares the performance of our proposed approach with sparsity potentials (HF + sparsity) to the Hough Forest baseline (HF baseline [Gall et al. 2011]). As can be seen, by using the sparsity potentials the performance has been substantially improved for most categories. The comparison to the state-of-the-art approaches is shown in the second block. We outperform the best previously published Hough transform based approach (ISK [Zhang and Chen 2010]) in mAP. The other two rows give a comparison of our approach compared to the latent part model (LSVM [Felzenszwalb et al. 2009]) and the best results of the PASCAL VOC Challenge (VOC best [Everingham et al.]).
References


[Alexe et al. 2010] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In CVPR, 2010. 2.2, 3.1, 3.5.3


[Dalal and Triggs 2005] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005. 2.2.1, 2.2.1, 2.2, 2.2.2, 2.2.3, 2.2.3, 7.3


Gall and Lempitsky 2009] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009. 2.4, 3, 3.1, 3.3, 3.4, 3.4.1, 3.5.3, 4.0.2, 4.1.1, 4.2.1, 4.4.4


[Hough 1962] P. V. Hough. Method and means for recognizing complex patterns, 1962. 2.4.2


scene categories. In *CVPR*, 2006. 2.2.2, 2.4, 2.6, 2.4.1, 2.4.1, 2.4.2, 4.3.1, 7.3


[Leibe et al. 2008] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008. 1, 2.2, 2.2.2, 2.4, 2.4.2, 2.4.2, 2.4.2, 3, 3.1, 3.2, 1, 3.3, 3.6, 4, 4.1.1, 4.2.1, 4.3.1, 5, 5.1, 5.2, 5.2, 5.5, 6.2.1


[Maji and Malik 2009] S. Maji and J. Malik. Object detection using a max-margin hough transform. In CVPR, 2009. 2.4, 2.4.1, 3.1, 3.4.5, 4.1.1, 5.1, 6.2.1

[Maji et al. 2008] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. CVPR, 2008. 2.4.1


[Viola and Jones 2004] P. Viola and M. Jones. Robust real-time face detection. IJCV, 57(2):137–154, 2004. 2.2.1, 2.3.1


[Woodford et al. 2011] O. Woodford, M. Pham, A. Maki, F. Perbet, and B. Stenger. Demisting the hough transform. In BMVC, 2011. 2.2.3, 2.4.2, 5.1


[Zhang and Chen 2010] Y. Zhang and T. Chen. Implicit shape kernel for discriminative learning of the hough transform detector. In *BMVC*, 2010. 2.4, 3.1, 5.1, 5.3.2, 5.1, 6.2.1, 6.1, 7.3


**Curriculum Vitae**

<table>
<thead>
<tr>
<th>Name</th>
<th>Nima Razavi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date of Birth</td>
<td>September 2\textsuperscript{nd}, 1984</td>
</tr>
<tr>
<td>Place of Birth</td>
<td>Shemiran, Iran</td>
</tr>
<tr>
<td>Citizenship</td>
<td>Iran</td>
</tr>
</tbody>
</table>

**Education**

<table>
<thead>
<tr>
<th>Year</th>
<th>Degree</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008 – 2012</td>
<td>Doctor of Science ETH</td>
<td>ETH Zurich, Computer Vision Laboratory</td>
</tr>
<tr>
<td>2006 – 2008</td>
<td>Master of Science ETH in Computer Science</td>
<td>ETH Zurich, Department of Computer Science</td>
</tr>
<tr>
<td>2002 – 2006</td>
<td>Bachelor of Science in Computer Engineering-Hardware</td>
<td>Sharif University of Technology</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Department of Computer Engineering</td>
</tr>
</tbody>
</table>

**Occupations**

<table>
<thead>
<tr>
<th>Year</th>
<th>Role</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012 – 2012</td>
<td>Research Intern</td>
<td>Microsoft Research, Cambridge, UK</td>
</tr>
<tr>
<td>2008 – 2012</td>
<td>Research Assistant</td>
<td>Computer Vision Laboratory, ETH Zurich</td>
</tr>
<tr>
<td>2008 – 2008</td>
<td>Student Intern</td>
<td>De La Rue Int. Ltd. (Now Talaris Co.), Berne Branch</td>
</tr>
<tr>
<td>2007 – 2008</td>
<td>Research Assistant</td>
<td>Institute for Computational Science, ETH Zurich</td>
</tr>
<tr>
<td>2003 – 2006</td>
<td>Student Intern</td>
<td>Institute for Research in Fundamental Sciences (IPM)</td>
</tr>
</tbody>
</table>