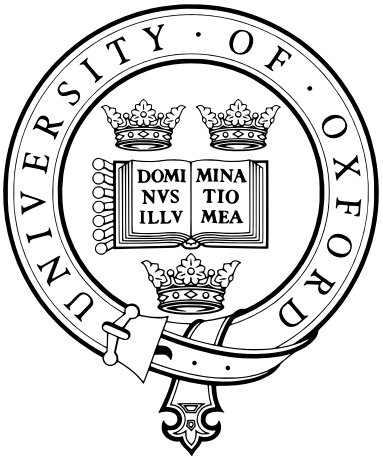
Semantic Image Segmentation and Web-Supervised Visual Learning

D.Phil Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford



Supervisors:
Professor Andrew Zisserman
Dr. Antonio Criminisi

Florian Schroff
St. Anne's College

Trinity, 2009

Florian Schroff
St. Anne's College

Doctor of Philosophy
Trinity Term 2009

Semantic Image Segmentation and Web-Supervised Visual Learning

Abstract

Given an image, the goal of this work is to recognise objects of certain categories despite intra-class appearance variations and small inter-class differences. The appearance of objects in photographs is influenced by lighting, scale, different poses, viewpoints, articulation of objects, clutter and occlusion. Two different aspects of object recognition are investigated in this thesis. The first part develops models for semantic object segmentation of natural images and relies on groundtruth labelling for training. The second part uses the implicit supervision that is available on web-pages to learn visual object-class models automatically. It can then provide training data for object detection or segmentation algorithms.

The goal in the first part is to label connected regions in an image as belonging to specific object classes, such as grass or cow. We introduce a compact model, where each class is modelled by *a single* histogram of visual-words, this is in contrast to common nearest-neighbour approaches which model *each* class by storing exemplar histograms. After introducing segmentation algorithms based on these histogram models we extend the Random Forest classifier and evaluate its feature selection properties for the semantic object segmentation task.

Most object recognition methods rely on labelled training images. For each object category to be recognised, the system is trained on a set of images containing instances of these categories. The second part of this thesis focuses on the automatic creation of sets of images that contain a certain object class. The idea is to download an initial set of images from the Internet based on a search query (*e.g.* penguin). Given the images a text based ranking that exploits the information on the web-pages is performed. This ranking is then used to automatically learn visual models for object categories. We compare the performance of our system to previous work and show that it performs equally well without the need of explicit manual supervision.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Florian Schroff, St. Anne's College

Acknowledgements

I would like to thank my supervisor Professor Andrew Zisserman and my co-supervisor Dr. Antonio Criminisi. Without their support, guidance and encouragement this work would not have been possible. I would also like to thank Microsoft Research for supporting me with the European PhD scholarship program, which allowed me to work on my research and to present my work at international conferences.

I thank Professor Hans-Helmut Nagel for sparking my interest in computer vision during my diploma thesis and for supporting me to achieve my goals.

Specifically, I would like to mention my long-term lab colleagues Maria-Elena Nilsback, Mukta Prasad, James Philbin and Patrick Bühler, for being helpful colleagues, friends and entertaining travel buddies. But I am thanking everyone in the Vision Lab for creating a friendly working environment and for the fruitful discussions and relaxing tea breaks. I would also like to thank my college mates and all the new friends I have made in Oxford for interesting discussions and distraction from the sometimes busy work schedule. I am grateful to Maya Ganapathy, Amreeta Mathai, Matthew Blaschko, András Salamon and Michele Taroni for proofreading parts of this thesis and/or inspiring me in other ways.

Finally, I would like to thank my parents and brothers for their support, understanding and for always being there.

Contents

Chapter 1

Introduction

Nowadays there are numerous digital images that are readily available. The number of images on mobile phones, personal image collections on the computer or online photo albums such as Google's Picasa greatly increased over the past years due to the abundance of digital consumer cameras. In addition to the increasing number of images on web-pages there is a dramatic increase of image sharing sites such as Flickr. In order to take advantage of all these images, advanced visual content recognition techniques become more and more important. Systems that are able to provide high-level information about the objects contained in the images can provide immediate advances. Starting with the wish for a more reliable web image search to tools for handling personal image collections, there is an abundance of possible applications. One might want to automatically build a list of all objects and people that are shown in personal collections. The possibility to search for all images of one person, given a query image or name, as well as grouping images by location are examples of useful applications. Computer vision methods and object recognition specifically, can help to cope with the increasing size of personal image collections. In addition, there is a wide range of industrial applications such as driver assistant systems or quality control.

One important goal in image analysis is to label the objects that occur in natural images. Research in this area started over 40 years ago. Roberts (1965) describes how simple objects in an image can be transformed into a 3D line representation. Since then visual image analysis as well as object recognition were subject to research by many people. Taking this amount of past research and the current state of the field into account, it is unlikely that performance comparable to humans can be achieved in the near future. Humans are estimated to distinguish

(a) (b)

Figure 1.1: Context plays an important role for humans in object detection. The image (a) taken from Oliva and Torralba (2007) shows a car on the street and the same shape rotated by 90°, which then appears to be a pedestrian. (b) gives an example where "context" or expected observations contradict the image objects.

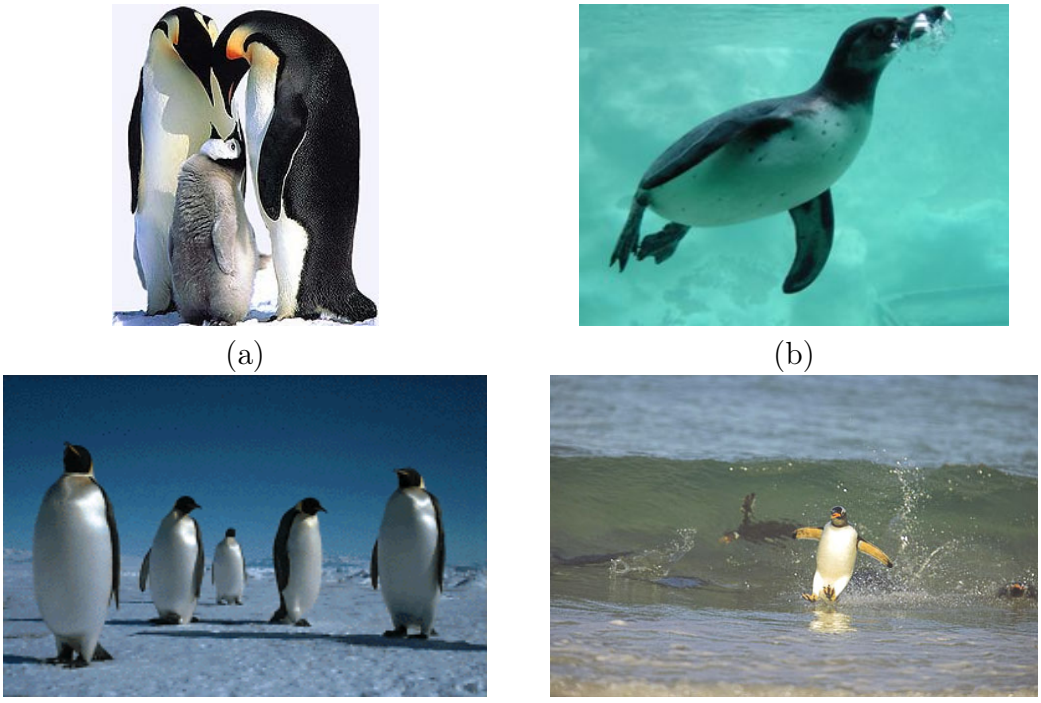
about 30,000 classes [Norman (2002)]. Although current systems seem to be far away from matching human recognition abilities, computers are already able to exceed human performance when it comes to large scale tasks, *e.g.* searching millions of images on the Internet. Many of these tasks have in common that the cost of error is not very high. Applications in robotics or driver assistant systems on the other hand often require a very high reliability as errors can lead to fatal accidents.

This thesis focuses on the task of finding images that contain certain object-classes and on automatically segmenting those objects in images. The next section gives an overview of the challenges that are encountered. It is followed by a more specific problem statement and an outline of the thesis.

1.1 Challenges

In contrast to what one might infer from their own ability to solve the object-detection task in fractions of seconds and with a very small error rate, there exists a wide range of difficulties that need to be overcome by an automatic system, and that are handled very well by humans.

There is a range of possible variations of object appearances especially in natural images



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

Figure 1.2: Appearance variations of penguins. (a) different *size* of baby penguins and grown-ups. (b) different *aspect*, *lighting* condition and *pose*. (c) different *scale* of same sized penguins due to projective transformation. (d) background clutter and unusual *aspects*.

and humans often use the *context* of the image to decide about the specifics of the object-classes contained in an image. Figure 1.1(a) illustrates how important context can be to distinguish similar looking shapes. In Figure 1.2(d) only the presence of the clearly recognisable penguin allows to infer that the other "dark" objects under water might be penguins as well. Generally context can be helpful to resolve ambiguous cases as for example in Figure 1.3(b) and Figure 1.5(a) where the presence of clearly recognisable objects resolves possible confusion with other object-classes, sheep and birds in this case. If objects can be recognised without ambiguity it is less important to take context into account, in fact, disregarding context allows recognition despite apparent contradictions with expected observations as in Figure 1.1(b). This section gives an idea of the main challenges that arise during the object recognition task and can lead to such ambiguities. The appearance variations of objects can be divided into two main classes:

- **Object variations** stemming from the object itself such as *articulation*, *pose* and *size*, which affect the *intra-class variations* directly, but can also influence the *inter-class differences*.

- **Image variations** such as *lighting* conditions, *scale*, *viewpoint*, *background clutter*.

(a) (b)

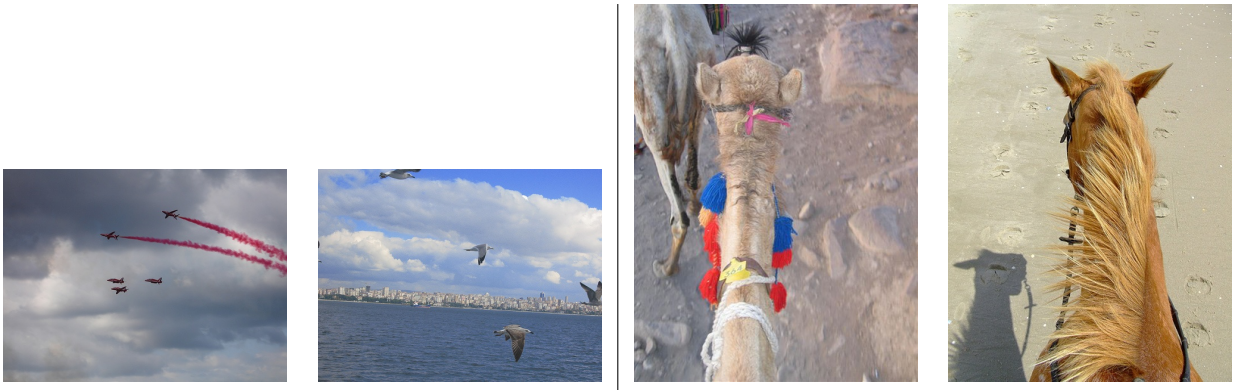
Figure 1.3: Intra-Class variations. Each image shows instances of the same object-class, but with very different appearances.

Figure 1.4: Object-class by functionality. Some object-classes are defined by functionality rather than by visual similarity.

Both of these classes will be discussed in detail in the following.

Object Variations

- *Intra-class variation* describes the variation that occurs between objects belonging to the same class (*e.g.* different breeds of dogs in Figure 1.3). There are many categories which have large intra-class variations and therefore complicate the automatic categorisation of those objects into the same class. One of the problems is that what we commonly describe as object-categories is often based on functionality rather than appearance (*e.g.* chairs, see Figure 1.4). In order to being able to distinguish objects based on functionality a vast body of knowledge about the world needs to be taken into account in addition to the image context and the visual appearance of the object itself.



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

Figure 1.5: Inter-Class variations. Some of the objects in (a) airplanes, and (b) birds, have a similar appearance and only the context enables us to be certain about which object-class they belong to. (c,d) are two examples of similar appearances of a camel and a horse with only very few distinguishing features.

- *Inter-class variation* refers to variation that occurs between objects of different classes. Due to the fact that there are many object-classes with relatively little appearance variation between them, it is challenging not to confuse those classes (for example sheep and white calves in Figure 1.3(b)). Figure 1.5 shows further examples of little inter-class variation and the confusion is partly resolved by taking the context into account.

Conditions that affect inter- and intra-class variation

- *Articulation* describes the variation of appearance caused by different positions of parts of the object relative to each other. It mostly refers to living objects (*e.g.* humans: different arm positions, sitting, running, standing, kneeing), but also applies to other object classes (*e.g.* diggers, trucks, closed or open laptops). For parts-based models this requires to explicitly take deformation into account, other object-class models which do not model the positioning of parts relative to each other implicitly allow for more shape variation at the expense of less discrimination between object-classes.

- Objects occur in different *poses* and can have completely different appearances as a result. As an example, standing or swimming penguins in Figure 1.2(b,d).

- The *size* of objects can significantly influence the similarity to other object-classes and increase the variance within one object-class. There are, for example, penguins in many different sizes, which together with other variations can cause confusion with other bird species (Figure 1.2(a)).

Image Variations

- *Lighting conditions* have a major influence on the appearance of objects. Due to different lighting and the occurrence of shadows objects can appear completely different, or are generally difficult to recognise in the first place, even for humans. Figure 1.3 on page 4 shows how strong shadows can add to background-clutter and even if it does not directly affect the objects' appearances it will still add strong gradient edges to the image, which leads to confusion for many computer vision methods.

- *Scale* variations can directly be caused by the imaging device or can be due to perspective transformation (Figure 1.2(c)). The example given in Figure 1.5(a,b) also illustrates the confusion that can occur due to these scale issues and the related differences in level of detail.

- The *aspect* (viewpoint) can significantly change the appearance of an object. Aspect refers to position of the camera relative to the object. For example, the frontal view of a penguin versus a penguin seen from below or above under water (Figure 1.2(b) and (d)).

- *Clutter* can result in confusion between foreground objects and background. Object features are more likely to occur in the background thereby producing false matches and wrong evidence about the presence of objects.

- *Occlusion* is mostly caused by other objects or truncation by the image border. Self occlusion describes the situation where parts of the object occlude other parts of the same object. All these cases complicate the recognition task significantly as the visual model either needs to explicitly model the possibility of missing parts or needs to be sufficiently robust to it. Figure 1.3(b) gives an example where objects of the same type occlude each other. Figure 1.6(b) shows various occlusions between instances of different object-classes (motorbike, bus, car). These cases are challenging even for humans.

1.2 Problem Statement

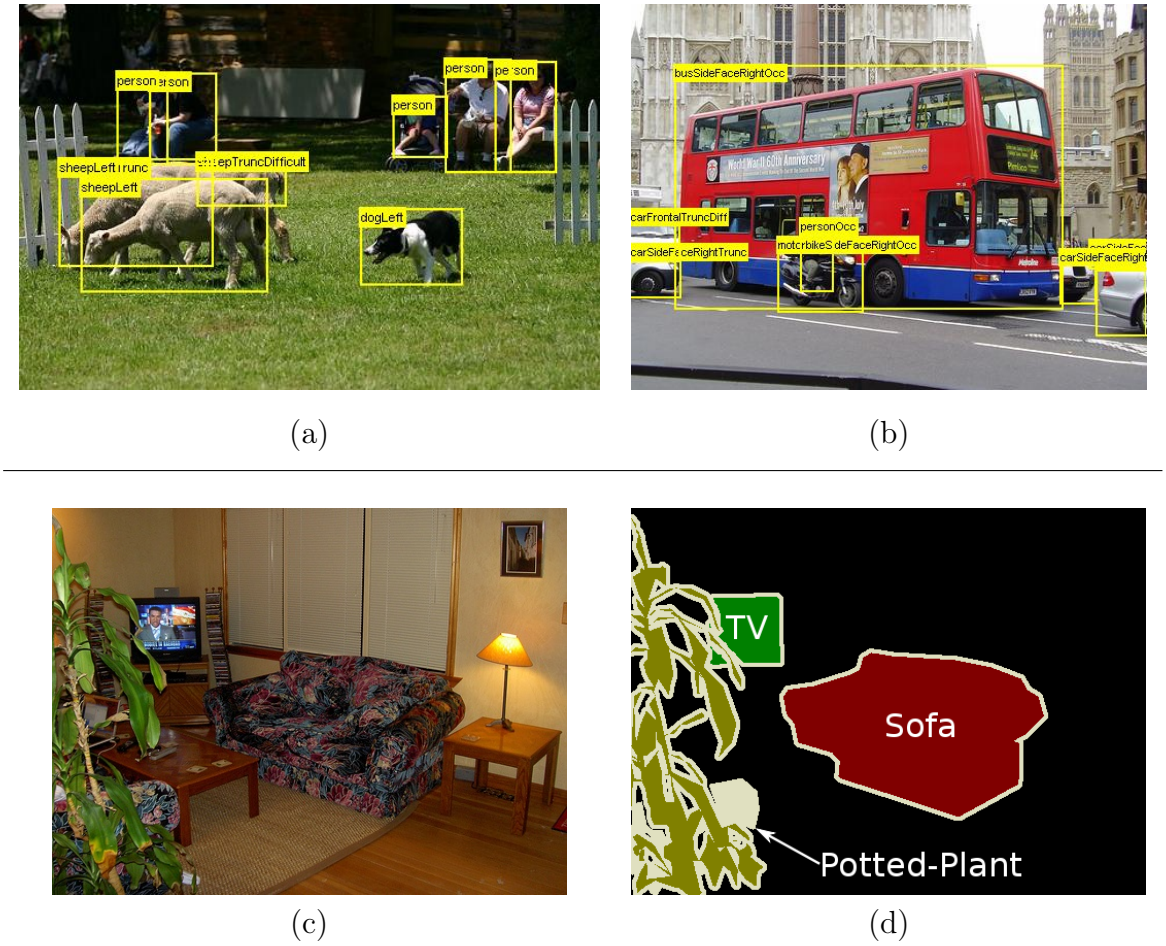
Object recognition distinguishes between the detection of specific object instances in different views and images and the detection of object-classes. Object instance detection has already

advanced very far and current systems can solve tasks such as searching millions of images on Flickr [Philbin *et al.* (2008), Jegou *et al.* (2008)] or searching specific objects in videos [Sivic and Zisserman (2006)]. Object-class recognition, on the other hand, needs to cope with additional challenges as pointed out in Section 1.1, namely intra- and inter-class variations. Current state of the art object recognition systems tackle problems with 20 object-classes [Everingham *et al.* (2008)] in difficult real world images and 256 object-classes [Griffin *et al.* (2007)] on simpler images, and are far from human abilities in terms of number of object-classes and recognition accuracy.

An ultimate goal might be a pixel or sub-pixel labelling of an image – an image parser that would not only assign the image pixels to specific objects, but provide a complete description of the image content including a description of object parts, as well as their relations and relative positions [Storkey and Williams (2002), Jin and Geman (2006), Zhu *et al.* (2007)]. Ideally one would have a smooth transition from the identification of specific object instances in different images to recognising object-classes. In this thesis we concentrate on the task of finding images containing specific object-classes or assessing the presence of objects. The definition of object-class is somewhat vague in the community and it is undoubtedly the case that in the long term one would prefer to use a hierarchical representation. For example, a dalmatian would also be labelled as a dog and an animal. In addition the object's parts such as legs, ears, or head should be labelled as well. Other work such as Liu *et al.* (2007), for example, focuses on finding *generic* salient objects in images instead of aiming to recognise objects from a pre-specified list of classes.

The most basic task in this direction would be to decide if a specified object is present in an image or not. This is often referred to as image classification. The localisation of objects in images is commonly called object detection. This has to be distinguished from object segmentation which does not only detect the position of the object but also returns its extension. *I.e.* the image is segmented into regions with each region corresponding to a specific object or object-class. Figure 1.6 gives examples for both these cases. Object recognition can be divided into these more specific sub-tasks, which are commonly defined as follows [Everingham *et al.* (2008)]:

Image classification. A given image is labelled based on its content. In the simplest case this just denotes if a certain object is present in the image or not (*e.g.* bus in the top right image in



(a) (b)

(c) (d)

Figure 1.6: Detection *vs.* segmentation. The figure illustrates the difference between object detection (a,b) and object segmentation (c,d) on example images from the Visual Object Classes Challenges [Everingham *et al.* (2007, 2008)].

Figure 1.6). This is also called object classification. Alternatively the system could also return a list of the main objects occurring in the image, or perform a *scene classification* of the image, for example outdoor meadow, street scene, or indoor living-room, for the images in Figure 1.6. Related to this is *region classification* where a given region is considered and classified. Often this method together with the restriction that each region contains *one* object-class only is used to simplify the classification task and evaluate algorithms on a simpler task.

Solving this problem can specifically contribute to the improvement of image search engines, such as Google image search for example. In Chapter 6, we perform image or object classification, *i.e.* decide if there are objects belonging to a specified object-class present in the image or not. We use this to re-rank images downloaded from the Internet, or returned by Google image search. Other applications include video search, *e.g.* searching the thousands of hours on YouTube or news archives for the occurrences of specific objects, in order to trim down the incredible amount of data and simplify video browsing for humans.

Object detection. Given an input image the goal is an output similar to the bounding boxes shown in Figure 1.6, *i.e.* the areas where objects belonging to a set of pre-specified object-classes occur should be marked and labelled with the corresponding class. One could also expect the algorithm to give a more specific description such as the orientation (rear, front, left or right) and if the object is truncated or occluded. In the Visual Object Challenge (VOC) datasets, however, this is currently only used for training purposes. Solving this task would also induce a solution to the previous problem, *image classification.* It is, however, much more difficult to build an object-detection system than an image classification system and the run-time complexities are usually increased as well. Consequently, it is usually advantageous to focus on image classification if that is the problem one is interested in.

Solutions to the detection problem have possible applications in robotics, where robots need to detect objects to avoid or interact with them. Related is the application of driver assistance in cars. Those systems should be able to detect cars or pedestrians in order to avoid collisions by issuing warnings to the driver, for example. Face detection is an example that was successfully transferred into many consumer cameras where it supports the auto-focus system to keep all faces in the picture in focus.

Object segmentation. This is a simpler version of the general labelling problem where a parse tree for all structures in the image down to the pixel or sub-pixel level is returned. Object segmentation is more general than object detection and image classification in the sense that they both can be derived from object segmentation results. It needs to be pointed out that in general objects also include categories like grass, sky, or water that can extend "indefinitely". They are sometimes referred to as "stuff" categories. A complete segmentation system would label those "objects" as well. Chapter 4 and Chapter 5 investigate various methods to retrieve the segmentation of an image into constituent object regions for a set of pre-specified object-classes. Figure 1.6 shows an indoor scene and an image segmentation into the object regions corresponding to the object-classes TV-monitor, potted-plant, and sofa.

Some of the possible applications are related to image or video editing, where instead of interactive approaches such as Rother *et al.* (2004) further automation can be imagined. Good automatic segmentations of objects can allow automatic replacement of the background, similar to Yin *et al.* (2007), but more general and based on a specified set of object-classes. Reliable

automatic object segmentation can also help to anonymise publicly available image or video data. The images available in Google's street view could be further anonymised by removing cars or people in addition to blurring faces and number plates as it is done already.

1.3 Thesis Outline

We give an overview of related literature and relevant methods in Chapter 2, where we introduce standard methods for *image description* and provide a comprehensive overview of the machine learning tools used throughout this thesis. Chapter 3 introduces the image datasets and performance measures that are used to evaluate our proposed methods.

The main part addresses two different aspects of object recognition. First, in Chapter 4 and Chapter 5, we investigate various methods to classify pixels based on their local image context, *image* or *object segmentation*. In the current setup, this requires strong supervision during training. Complementary to that, we provide a system to learn visual object-class models for image classification based on no explicit supervision in Chapter 6. We exploit the vast amount of data that is available on the Internet to provide the supervision that is necessary for the training of a visual-model. This can then provide supervision for the segmentation system and reduce or eliminate the amount of manually labelled training data that is necessary for training.

Chapter 7 concludes and summarises our observations. We also show preliminary results on how the harvested images can be used to train our segmentation system. We then presents ideas for future research. The following two paragraphs summarise the two main aspects of this thesis.

Object segmentation. The first part (Chapter 4 and Chapter 5) addresses the problem of pixelwise image segmentation. Parts of this work were published in Schroff *et al.* (2006, 2008). Our approach classifies each pixel in the image by taking its local context into account. In Chapter 4, we introduce a compact model for object-classes that uses local context in the form of a fixed sliding-window. Due to the compact representation the computational requirements are drastically reduced in comparison to standard nearest neighbour methods. We then investigate the use of a purely image content driven bottom-up segmentation (*i.e.* colour, texture, or gradient) in order to define the context of a pixel and show how it improves the segmentation.

In Chapter 5, we employ the Random Forest framework to learn the context of pixels discriminatively and improve significantly over the simpler histogram based models with "fixed" pixel context. In addition to this context the classifier also selects the types of low-level features that are most discriminative.

Harvesting image datasets from the web. For current object recognition research good training and test sets are crucial and often need to contain labelled image data which are very time consuming to construct manually. Image search engines apparently provide an effortless route, but currently are limited by poor precision of the returned images and restrictions on the total number of images provided. For example, with Google Image Search the precision is as low as 32% on one of the classes tested here (shark), averages at 39%, and downloads are restricted to 1000 images[1]. Flickr also provides millions of images together with user defined tags that describe the image contents. Even though a vast number of images are available on the net, it is not straight forward to retrieve a set of images belonging to one specific object category.

Following Berg and Forsyth (2006) we use a web search engine to obtain a large pool of images and the web pages that contain them, for a specified object-class. The low precision of such an initial pool of images does not allow us to learn a visual class model from such images directly. Our system is based on the work presented in Schroff *et al.* (2007). An initial filtering step aims to remove images such as drawings, cartoons and sketches. Then we apply a class independent Bayes classifier that ranks each image based on textual attributes. The final step builds on top of the text ranker and uses its output to learn a visual model for the specified object-class. This model can than be used to rank images without the use of textual meta-data. The applicability of this method is proven in various experiments on a wide range of datasets, and it is shown to perform comparable to or better than state of the art without the need for human interaction.

[1]These numbers are based on our work carried out late 2006 and early 2007.

Chapter 2

Literature & Methods

This chapter provides an overview of previous relevant research on object recognition. It focuses mainly on the topics of object recognition and object segmentation and starts with a short introduction to commonly used feature extraction methods and feature descriptors. Feature description plays a crucial role in object recognition and it is important that the extracted features are able to discriminate between different object-classes for the whole system to perform well. If the feature descriptors for two different object-classes are very similar, the best classifier will not be able to return useful classifications. In Section 2.2 we present a set of widely used classifiers. The remainder of this chapter introduces further literature on object detection and focuses on work in the area of object segmentation.

2.1 Low-Level Features and Object-Class Models

In this section the basic techniques for describing images and the objects therein are laid out. The first part gives an overview of the low-level features used throughout this thesis. These features describe the statistics of the images and objects therein. They make up a crucial part of any object classification system as the object-class models rely on the discrimination of the underlying features. Simply put, properties that cannot be described by the features cannot be used to discriminate between the object-classes. While trying to describe the image content as completely as possible the extracted features aim to be robust to external influences like viewpoint or lighting changes. After fundamental methods for the image description or more generally image region descriptors are presented in Section 2.1.1 we introduce a very basic

object-class model in Section 2.1.2, the common *bag of visual-words* model (BOW) and the related texton histogram model which is extensively used in Chapter 4. Finally we discuss how other research models objects and object-classes.

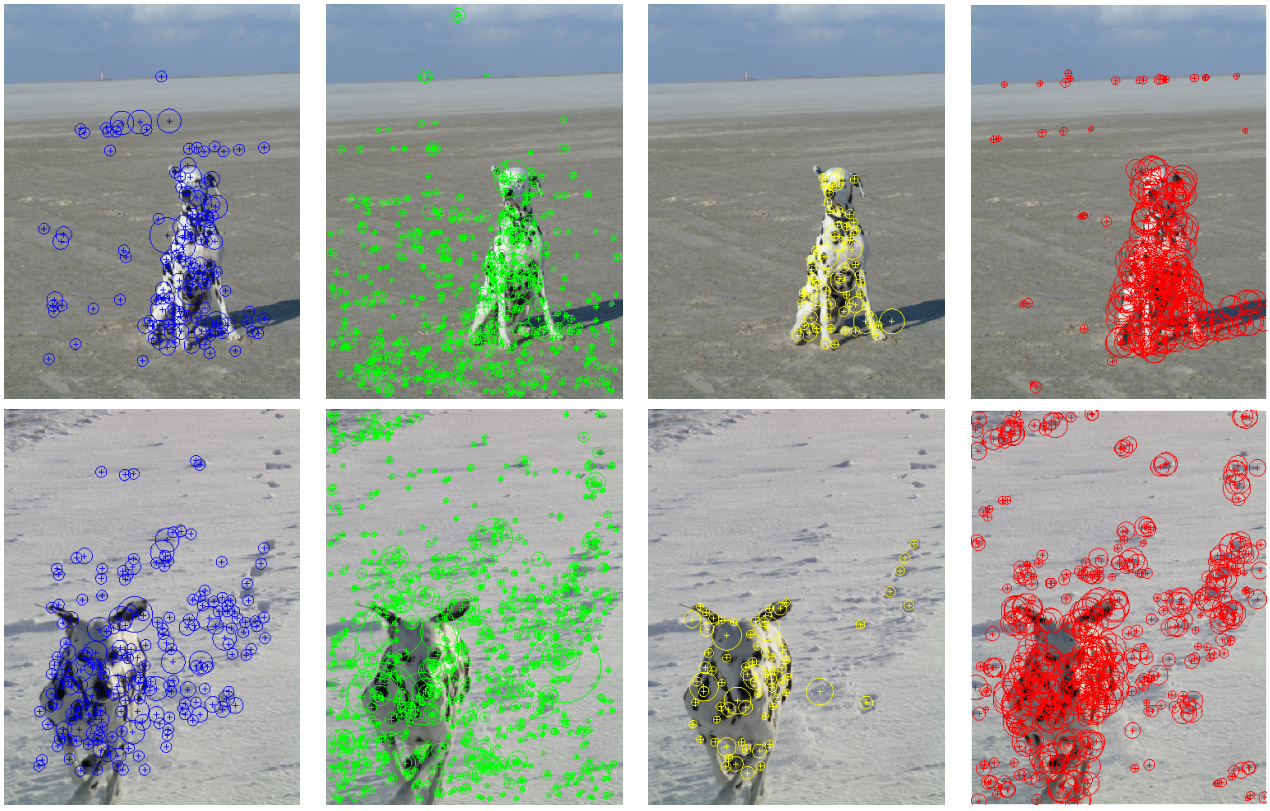
2.1.1 Image and Region Description

Each object or image classification system requires a representation of the input images. In the following three paragraphs we introduce *sparse* and *dense* image representations and feature descriptors. Sparse as well as dense image representations can focus on specific image regions instead of describing the image as a whole. The next paragraph provides a detailed overview of *sparse* representation techniques, which only represent "interesting" areas of an image. This is followed by the presentation of methods that provide a *dense* representation of the image in the sense that each pixel contributes to the feature description of the image.

Sparse image representation

In general sparse image representations are carried out in two steps: (i) finding interest points/regions in the image; (ii) applying a feature descriptor to each region in order to retrieve a vector (called feature descriptor) which describes that specific image region. This provides a *sparse* representation of the image, since only a subset of image regions is represented by these feature descriptors. We now give an overview of a few *interest point detectors* that are used throughout the computer vision community to model images sparsely. Examples of detected interest points are shown in Figure 2.1. Note that the early interest point detectors only detected *points* of interest (*e.g.* Harris corner detector), whereas newer methods tend to determine regions of interest that fulfil certain invariance properties (*e.g.* the salient region detector by Kadir *et al.* (2004)).

Edge and corner detectors. One of the first introduced, and compared to newer achievements very basic but still influential, interest point detectors is the *Harris corner detector* [Harris and Stephens (1988)]. It is based on detecting corners as areas with low self similarity, *i.e.* small shifts of an image patch result in a large sum of squared differences. The *Canny edge detector* [Canny (1986)] was introduced earlier and provides a basis for the combined corner and edge detector in Harris and Stephens (1988). It detects image edges based on the gradient,



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

Figure 2.1: Interest point detectors. Example images with interest points overlaid. The coloured circles indicate the position and scale of the detected interest points. The interest point detectors used are: (a) difference of Gaussians, (b) Multi-scale Harris, (c) Kadir's salience regions detector, (d) Canny edge points (see text). The images are part of the VOC2006 dataset [Everingham *et al.* (2006)].

then uses non-maximum suppression, and hysteresis thresholding with a high and low threshold to trace edges along the image. One technique to extract interest points from the edges is to sample points along those edge elements. This provides us with two sets of sparse interest points, namely points in the form of corners and points sampled along edges. As opposed to the region detectors introduced in the next paragraph, these versions only return points, and the regions (scale of the circles in Figure 2.1) are usually chosen independently from the image content. The scale can be sampled from a predefined range, for example. In this work we use the multi-scale Harris detector as described in Mikolajczyk *et al.* (2005). It uses a Laplacian operator to select the scale. It is also possible to use an elliptic region and detect the shape of it using the second moment matrix of the intensity gradient [Baumberg (2000)].

Affine region detectors. Recent interest point detectors have been extended to detect scale or affine invariant regions, therefore also called regions of interest detectors. Affine region

detectors are built to return image regions that are invariant to viewpoint changes, specifically invariant to affine transforms. Ideally, two regions extracted from the same part of an object imaged from two different viewpoints contain the same image content and thus the corresponding image patches are almost identical, despite scale or affine transformation of the originating images. Lowe (1999) uses an approach based on the difference of Gaussians to find scale and rotation invariant interest points. It draws ideas from the scale invariance analysis carried out by Lindeberg (1998). An overview and evaluation of state of the art affine region detectors is given in Mikolajczyk *et al.* (2005). One of them is the salient region detector introduced in Kadir and Brady (2001) and extended in Kadir *et al.* (2004) which is particularly suitable for object-detection. It detects salient affine invariant regions in images. Examples are provided in Figure 2.1(c) and we use it in the work of Chapter 6. The detector deems a region salient if it exhibits unpredictability in both its attributes and spatial scale. Unpredictability is measured by the information theoretic entropy. The proposed method is invariant to viewpoint change, comparable to other state of the art detectors, *e.g.* the maximally stable extremal regions detector (MSER) by Matas *et al.* (2002) or Mikolajczyk and Schmid (2002). Furthermore, this salient region detector is insensitive to image perturbations and detects features repeatably under intra-class variation. This makes it superior to other (affine) region detectors, as Kadir *et al.* (2004) show in their experiments. One disadvantage is its computational inefficiency compared to other detectors like difference of Gaussians or multi-scale Harris.

Feature descriptors

Now we briefly present a few feature descriptors that are suitable to describe the detected regions of interest. In principle it is possible to use these feature representations in the dense framework as well. However, more evolved versions for dense image representations are introduced later on, some of which are based on the descriptors presented next. The "support" region for these feature descriptors is defined by the output of the interest point/region detectors from the previous section.

Patch based descriptors. This feature descriptor is used in sparse and dense representations. The patches are usually of small size (*e.g.* 3×3 to 20×20 pixels) and represent all colour-channels of an image directly or the intensity image only. Alternatively the response of

a filterbank can be used (see next section). A common technique is to vector-quantise small image patches, *e.g.* 5×5 pixels.

SIFT descriptor. In addition the these simple patch based descriptors, there are a variety of other feature descriptors most of which try to incorporate local spatial information in other ways. Probably the best known descriptor is the SIFT (Scale Invariant Feature Transform) descriptor introduced by Lowe (1999). It records the gradient orientations instead of image intensities and uses a spatial histogram which makes it more robust to small shifts. Specifically, SIFT defines a 128 dimensional feature descriptor consisting of an orientation (gradient, 8 bins) and location (4×4 bins) histogram weighted by gradient magnitude. We use 3×3 spatial bins in Chapter 6. A variety of versions exists, *e.g.* a rotation invariant version called RIFT [Zhang *et al.* (2007)]. In Mikolajczyk and Schmid (2003) an extension (GLOH) of SIFT is introduced. It uses log-polar spatial binning instead of the grid layout. The authors show that it outperforms SIFT slightly, which itself outperforms many other descriptors such as spin images [Lazebnik *et al.* (2003)], PCA-SIFT [Ke and Sukthankar (2004)], and shape context [Belongie and Malik (2002)] which uses an edge location histogram in log-polar coordinates and performs similarly except in textured scenes or when edges are unreliable. The gradient histograms seem to contribute significantly to this performance by representing local shape. One disadvantage of SIFT is its high dimensionality and one way to reduce it is using PCA-SIFT by performing Principal Component Analysis (PCA) on the raw 128 dimensional SIFT vector. This step is also part of GLOH. Originally SIFT refers to an implementation which uses a scale invariant region detector (interest point detector), based on the difference of Gaussians. The descriptor is however used stand alone as well in combination with various interest point detectors. In contrast to the high dimensional SIFT, the best *low* dimensional descriptors evaluated by Mikolajczyk and Schmid (2003) are gradient moments [Van Gool *et al.* (1996)] and steerable filters [Freeman and Adelson (1991)].

Shape descriptor. The previous descriptors capture basic image statistics, whereas others try to model the shape of object parts more explicitly. Forssen and Lowe (2007) introduce a shape descriptor for the MSER mentioned earlier. It applies the SIFT descriptor to a shape mask defined by the maximally stable extremal regions. The authors show that this shape

descriptor is more robust to illumination changes than SIFT computed on the intensity patches directly and that it increases the number of matches across images, especially in natural scenes with many near occlusions (*e.g.* trees, bushes). Other approaches capture shape even more explicitly and are based on template matching using Chamfer distance. Gavrila (1998) presents a hierarchical template matching system that uses distance transforms. Shotton *et al.* (2005), Opelt *et al.* (2006) use a similar approach for object detection. Specifically, they use a fragment dictionary of spatially localised re-occurring shapes (object outlines) in the boosting framework. This approach might be more suitable for objects where shape is the major defining property (*e.g.* mugs, bottles). Patch based descriptors like SIFT are more easily "distracted" by appearance properties (*e.g.* bottle labels) that are not as defining for this class as its shape. Their works show that purely shape based systems can perform equal to appearance based systems and the ideal object recognition system would combine appearance and shape based ideas.

Dense feature representation

Dense features are a widely used alternative to region detectors. Again either sub-regions of an image or the whole image can be described by a dense feature representation. "Dense" here means, that the features are not only extracted at previously detected interest points, but a feature descriptor is computed for each image-pixel or sampled on a dense grid. One advantage of dense representations over sparse ones can be the fact that regions with uniform texture, which usually aren't returned by interest point detectors (see the sand and snow in Figure 2.1 on page 14), will be represented equally well. The preferred method depends on the application and computational constraints. There is no general rule stating clear advantages of sparse versus dense image representations. Jurie and Triggs (2005) compare these two ideas on the object categorisation task and conclude that dense features perform better there. However, due to computational constraints a combination of sparse representations and dense sampling can be useful. Leibe and Schiele (2003) use a sparse representation of interest points as a first step and refine the initial object detection by further sampling of dense features around the initial hypothesis. Thereby, dense sampling in the whole image is avoided and only applied to "interesting" regions.

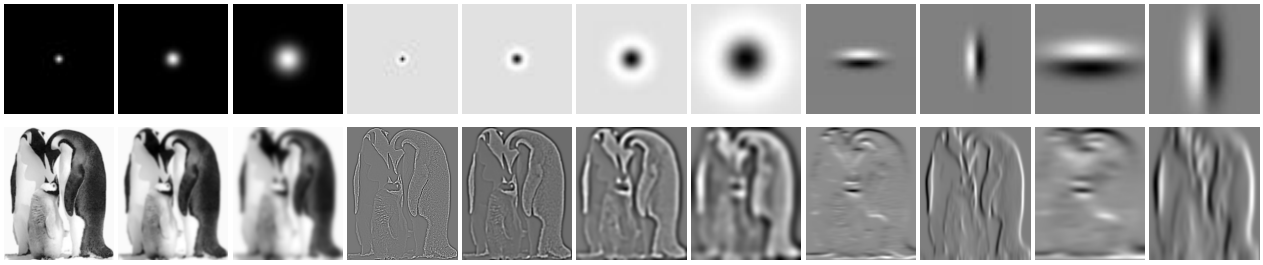


Figure 2.2: Filter bank and filter-responses. From left to right (top row): three Gaussians, four Laplacians of Gaussians, and four derivatives of Gaussians. The filter-responses (bottom row) are computed on the intensity channel L of the *Lab* colour space.

Filter banks. A set of filters is convolved with the image channels to give a set of filter-responses (see Figure 2.2), thus describing each image-pixel with a feature descriptor whose size depends on the number of filters and image-channels used. Leung and Malik (1999, 2001) use a total of 48 filters for recognition of surfaces (materials). They also introduce a feature quantisation step which is explained later on in this section. Varma and Zisserman (2003) evaluate the performance of filter banks and intensity patches for texture classification. The authors compare different filterbanks (MR8, MR4, LM and S) with a solely intensity based approach, and conclude that intensity based approaches can perform as well as filterbanks. MR8 consists of 38 filters that are based on Gaussians, Laplacian of Gaussians, and variations of first and second order derivatives of Gaussians to give six edge and bar filters with different orientations. The oriented filters are "collapsed" into the maximum response over all six orientations, thus resulting in only eight filter responses and a rotation invariant feature descriptor. The MR4 filterbank is related but only uses one instead of three different scales for the orientation filters. LM stands for the filterbank used in Leung and Malik (2001) and S for the filterbank used by Schmid (2001). Serre *et al.* (2005), who motivate their approach by biological correspondences use a set of eight Gabor filter bands, each band contains filters in two sizes. They compute the maximum response of each filter band in a certain image region which leads to a scale and shift invariant response per region. Patches from the resulting feature maps are used as feature descriptor. The classification depends on the minimum distance of test patches to training patches computed over all positions and scales. Practically, this approach provides a dense representation of the image, albeit it doesn't directly follow the common approach where features descriptors are computed in equally spaced window regions. Winn *et al.* (2005) designed the filterbank shown in Figure 2.2 with three Gaussians, four Laplacians of Gaussians, and

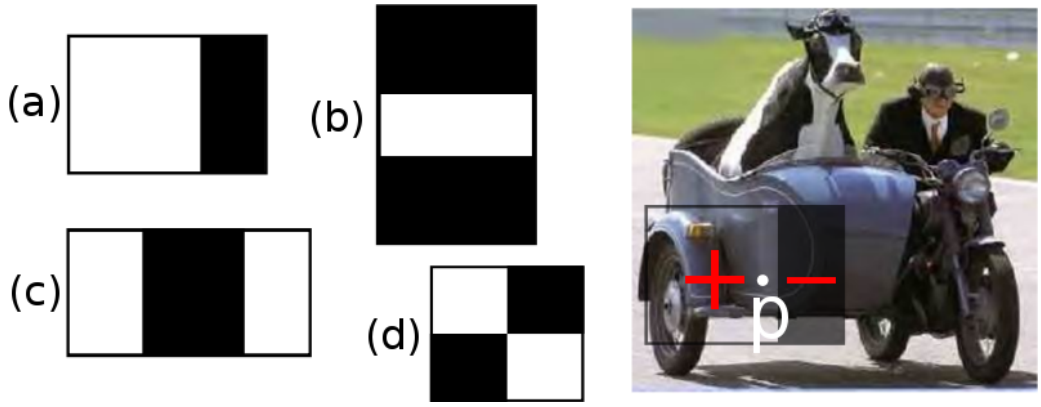
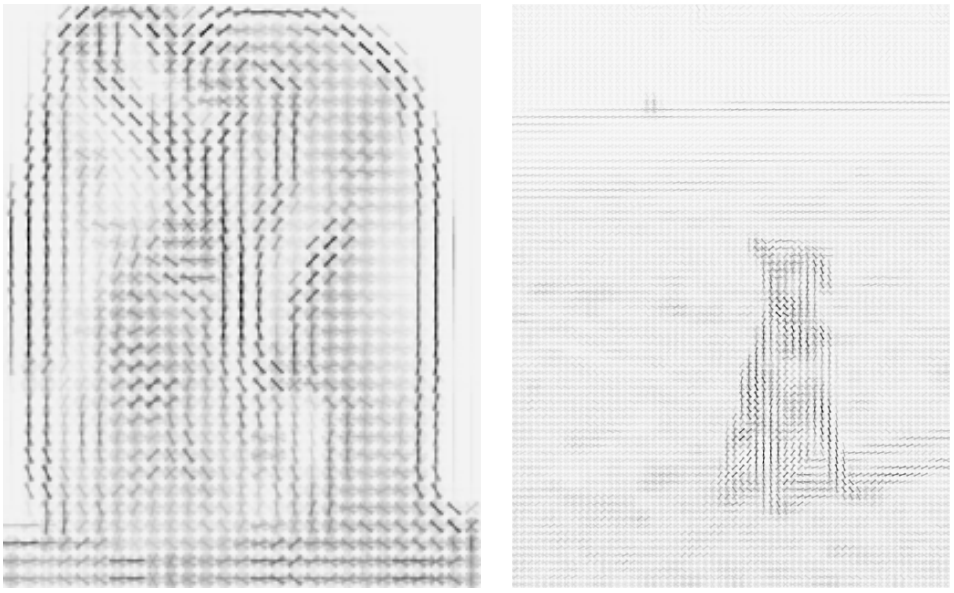


Figure 2.3: Haar-like features. (a) shows a two rectangle, (b,c) three rectangle, and (d) four rectangle Haar-like feature. The sum of the pixels in the black rectangles is subtracted from the sum of pixels in the white rectangles. For example, feature (a) is overlaid to an image on the right hand side. All pixel intensity values in the area marked with "+" are added and all intensity values in the black region, marked with "-", are subtracted from those. This results in the feature response for point p.

four derivatives of Gaussians from left to right. The filters are applied to the L channel of the CIE-LAB colour space, and the Gaussians are additionally applied to the a and b channels, thus resulting in a 17-dimensional feature response.

Haar-like filters. They are simple filters based on Haar wavelet basis functions and compute the difference between rectangular responses as illustrated in Figure 2.3. They were introduced to the computer vision community by Papageorgiou *et al.* (1998) and used in the famous work of Viola and Jones (2001), which introduces their face detector. Recent work uses related features in the Random Forest framework (*e.g.* Shotton *et al.* (2008)) and we present the details of our features that are a more general version of these Haar-like features in Chapter 5.

HOG descriptor. Figure 2.4 shows a visual representation for the HOG descriptor computed on example images. It was first introduced by Dalal and Triggs (2005). The so called *Histograms of Oriented Gradients (HOG)* descriptor, creates a dense image description by using locally contrast normalised 1D-histograms of oriented gradients. These orientation histograms are computed over small non-overlapping cells (*e.g.* 8×8 pixels) covering the whole image, thereby creating a dense description. Each of those cells is normalised with respect to a different block (a set of neighbouring cells) and thus contributes to the HOG descriptor multiple times. The authors tested several modifications from this baseline: *e.g.* radial cells,



(a) (b)

Figure 2.4: Example HOG. Visualisations of the HOG descriptor. The spatial histogram of each cell is visualised by plotting a weighted line in the appropriate direction. The input images correspond to the images used in Figure 2.2 on page 18 (HOG in (a)) and the image in the top row of Figure 2.1 on page 14 with its HOG descriptor shown in (b).

rectangular cells, or different sizes of cells and blocks. Often the HOG descriptor of the whole image is represented as a $1 \times d$ dimensional feature vector that can then be used in the SVM framework, as we do in Chapter 6. This descriptor idea can be seen as a dense version of the SIFT descriptor.

Quantisation of low-level features

The previous two sections introduced a range of widely used low-level feature descriptors. Many of these can be very high dimensional, *e.g.* 5×5 colour patches would define a 75 dimensional feature space, SIFT features a 128 dimensional feature space. Also the range of each dimension can be very large, either real valued for filterbanks, or usually one byte for patch descriptors and SIFT. Vector quantisation of these raw feature descriptors is a common step in the object or texture recognition community. One reason for the quantisation is the large range of values and their sensitivity to small image perturbations. Thus the quantisation introduces robustness and opens up the possibility for a variety of object-class models, some of which are introduced in the next section.

The most widely used method employs *k-means* clustering in the Euclidean vector space. K-means starts with k randomly selected data points, called cluster centres (different data

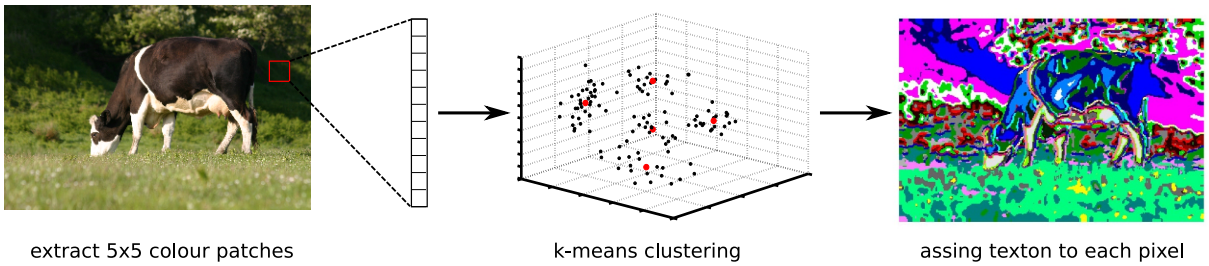


Figure 2.5: Vector quantisation. This figure illustrates the dense feature extraction of 5×5 colour patches followed by k-means clustering. In the quantisation step each pixel is assigned to the cluster centre (red) that is closest to its patch. Each colour corresponds to one of the 50 cluster centres / textons. More example images are shown in Figure 2.6.

driven initialisation techniques are used as well). The first step assigns each of the remaining data points to the closest cluster centre. We use Euclidean distance as the distance measure, but it is also possible to use Mahalanobis distance, for example. The next step recomputes the cluster centres to be the mean of each cluster. These two steps are alternated until convergence. Elkan (2003) propose a sped up version of k-means using the triangle inequality. Other methods such as agglomerative clustering in Leibe and Schiele (2003), which uses normalised grey-scale correlation on 25×25 image patches, or the mean-shift based method described in Jurie and Triggs (2005) are used as well. The latter method assures that dense regions in feature space are not overrepresented by visual-words. New cluster centres are added iteratively and placed based on mean-shift on sub-sampled data points. After a new centre was added all points assigned to it are removed (points are assigned to a centre if they lie within a certain radius r). The removal of points ensures that centres are not repeatedly assigned to dense regions. In standard k-means clustering this cannot be ensured. The radius r, in a sense enforces a minimal distance between cluster centres. Some of these methods are going beyond simple vector quantisation and are crafted towards their application, object recognition. Winn *et al.* (2005) introduce a method to reduce the size of the visual-word vocabulary while preserving its discrimination abilities. The authors learn a multivariate Gaussian distribution with diagonal covariance matrix over textons for each object-class and try to merge textons together in order to obtain a compact yet discriminative "Universal Visual Dictionary". Some of the problems of k-means, *e.g.* overrepresented regions in feature space can be solved this way.

Given the codebook described by the cluster centres computed with the method of choice, each original feature vector can be assigned to its closest cluster centre (*hard* quantisation) or

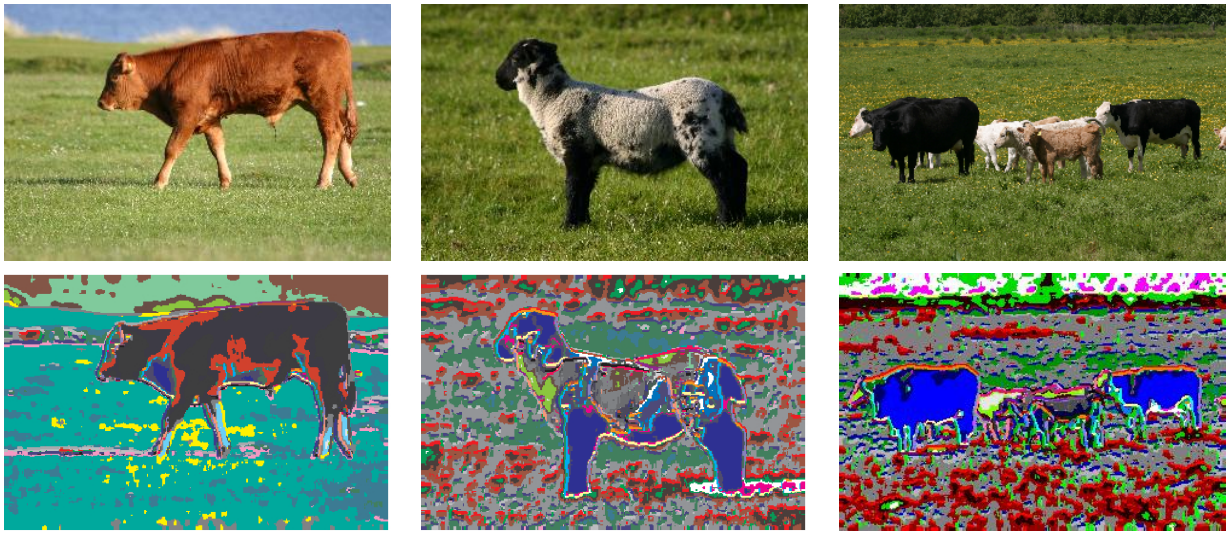


Figure 2.6: Texton maps. Different colours uniquely identify the 50 different textons.

to several of the closest cluster centres in various ways (*soft* quantisation); for example, the assignment can be weighted by the distance to each of the cluster centres. Figure 2.5 illustrates these steps for the case of a hard assignment. Agarwal and Roth (2002) use patches extracted at locations defined by an interest point detector. Each patch is then assigned *one* codebook ID (hard assignment) from a codebook that was computed using agglomerative clustering. Leibe and Schiele (2003) also use a patch based codebook computed by agglomerative clustering, but assign all matching codebook IDs to the image patch (using the same threshold t and normalised grey-scale correlation as distance measure that were used during clustering). This relates to a soft assignment with equal weighting. It is also possible to fit a Gaussian Mixture Model (GMM) [Bishop (2006)] using Expectation Maximisation (EM) to the feature vectors and define the soft assignment as the probability of the feature vector being assigned to each of the Gaussians. *I.e.* the number of Gaussians in the mixture model corresponds to the number of visual-words or textons. Csurka and Perronnin (2008) for example use this technique.

Textons. If the patches are extracted densely (*i.e.* for each image pixel) the assigned codebook IDs are often referred to as textons. The result may be visualised by generating colour-coded texton maps as shown in Figure 2.6. Leung and Malik (1999) first introduced this concept of textons into the computer vision community, and used a distribution over the quantised vectors. Cula and Dana (2001), Varma and Zisserman (2002a) extend this idea and show its value for the material classification task. Varma and Zisserman (2002b) compares this texton based classification to the PDF (probability density function) approach, where the

feature space is modelled by a PDF directly (*e.g.* using a Gaussian mixture model), and found that the performance was comparable on the material classification task. Textons can be based on a filterbank as in Winn *et al.* (2005), Shotton *et al.* (2006) or small image patches, as used throughout this thesis.

The next section gives an overview of object-class models that employ the texton approach described here. We also introduce the concept of visual-words and related models. Visual-words commonly denote sparse quantised feature descriptors, *e.g.* quantised SIFT descriptors extracted at regions of interest.

2.1.2 Visual Object-Class Models

As explained before, object recognition refers to image classification on the one hand and object detection on the other. First the *bag of visual-words* model is introduced as a very common model for image classification. Modifications of this baseline method are used for object detection or image retrieval and we employ the related *texton histogram* models throughout this thesis. After the presentation of these models we give an overview of *geometric* models followed by *explicit* and *implicit* shape models.

Bag of visual-words model

This is a commonly used model in object recognition, object or texture classification, scene classification, image retrieval and related tasks. It directly relates to the bag of words model (BOW) commonly used in text retrieval [Baeza-Yates and Ribeiro-Neto (1999)]. It has been introduced into the computer vision community by Sivic and Zisserman (2003), who apply it to object retrieval in videos. Csurka *et al.* (2004) use this approach for image classification. The BOW model is usually based on interest points and corresponding feature descriptions and uses a clustering-/vector-quantisation method on top of it. Eventually each interest point is represented by an ID indexing into a visual-codebook denoted V. An image is then modelled as a bag of those so called visual-words and can thus be described by a vector $\mathbf{h}$ that stores the distribution of all assigned codebook IDs. Note that this *discards the spatial distribution* of the image features. In contrast, the image descriptions introduced in the previous sections still carry spatial information, especially the dense ones, *e.g.* HOG, are often used directly to

provide a spatial description of the objects. We now give an overview of state of the art work that is based on this idea. In Chapter 6 we employ the BOW model with a support vector machine (SVM) classifier.

State of the art overview. Sivic and Zisserman (2003) use *tf-idf* weighting on the visual-word counts produced by interest point detection and SIFT features in order to retrieve frames in videos containing a query object. *Tf-idf* is a standard text retrieval tool which is used to compute the similarity of text documents.

Csurka *et al.* (2004) compare two common models, namely a Naïve Bayes method and a discriminative support vector machine approach. The SVM classifier uses the aforementioned visual-word distribution $\mathbf{h}$ for each image, together with a one against all approach for multi-class classification. The Bayes model is represented by the class conditional probabilities $P(v_t|C_j)$ for visual-word v_t, $t = 1, \ldots, |V|$, which are estimated from the training images using Laplace smoothing. Then the classification for an image is computed using

$$P(C_j|I) \quad \propto \quad P(C_j)P(I|C_j) = P(C_j)\prod_{t=1}^{|V|} P(v_t|C_j)^{N(t,I)} \tag{2.1}$$

with C_j for the object-classes and I for the test images. $N(t, I)$ is the number of occurrences of visual-word v_t in image I.

The bag of words model was employed by one of the most successful methods in the PASCAL 2006 challenge [Everingham *et al.* (2006)]. One of the best performing approaches uses a combination of the methods introduced in Zhang *et al.* (2007) and Lazebnik *et al.* (2006). The system uses the bag of visual-words model on sparse Harris-Laplace and Laplacian feature detectors or dense features on the one hand, and an extension which uses spatial pyramids to represent spatial dependencies on the other hand. This shows that this model compares to other state of the art object recognition methods despite its apparent simplicity and crude neglect of spatial feature relations.

Texton histogram models

As mentioned before, the texton histogram and the bag of visual-words model both refer to a bag of quantised low-level features. The difference lies in the terminology where the BOW

model refers to a sparse representation and textons refers to a dense representation where a texton ID is assigned to each pixel.

In this thesis we mostly use textons based on small 3×3 or 5×5 pixel wide colour patches in the *CIE-Lab* colour space, thus resulting in a 27 or 75 dimensional feature vector for each pixel as illustrated in Figure 2.5 on page 21. Instead of using the raw image pixels the responses of a filterbank can be used as pointed out previously. A simple object-class model based on texton histograms is now presented.

Modelling classes with textons. Given the textons, there are a variety of models that can be used to model an object-class. Histograms (distributions over textons) are the most widely used method to exploit the bag of visual-words methodology, see for example Fei-Fei and Perona (2005), Zhang *et al.* (2007). Quelhas *et al.* (2005), Sivic *et al.* (2005) and Bosch *et al.* (2006) use those histograms in connection with probabilistic Latent Semantic analysis (pLSA) [Hofmann (2001)] (see Section 2.2.3). Bosch *et al.* (2006) use dense SIFT features for the visual-word retrieval. Sivic *et al.* (2005) present an approach which "learns" object categories without any supervision. They compute SIFT features on a Caltech and MIT image dataset and then use pLSA and Latent Dirichlet Allocation (LDA) to discover "topics", in their case object categories. Shotton *et al.* (2006) use boosting on simple localised histograms to classify each pixel in an image for a semantic segmentation into object regions. In addition to the Bayes model in (2.1) we now introduce another simple, but often used model.

Nearest neighbour matching is a good baseline method that is still used often due to its simplicity and relatively good performance. One interesting point to note is that the k-NN classifier with majority vote is universally consistent [Devroye *et al.* (1996)]. A classifier is called consistent if with increasing data the risk of the classifier approximates the Bayes risk of the underlying distribution. It is called universally consistent if this property holds for all distributions of the data. Some even claim future image classification methods can be based on massive amounts of training data and the nearest-neighbour framework (*e.g.* Malisiewicz and Efros (2008)). Varma and Zisserman (2003) apply nearest neighbour matching to texture classification. Their textons are based on filter banks or intensity patches. During classification normalised frequency histograms that describe the texture distribution of the test image $\mathbf{h}$ are compared to the histograms of the training images. Each training image defines an exemplar

histogram $\mathbf{p}^j$. The distance between the histograms is measured using χ^2 distance, with the subscripts describing the histogram bins:

$$\chi^2(\mathbf{h}, \mathbf{q}^j) = \sum_i \frac{(h_i - q_i^j)^2}{h_i + q_i^j} \quad .$$

One important parameter in these kinds of models is the size of the visual-codebook $|V|$. Generally it seems to be the case that the optimal size and composition of the texton vocabulary depends on the specific application and cannot be determined analytically. Usually it is tuned based on cross-validation, or fixed to "reasonable" values. In Winn *et al.* (2005) they represent each class by a Gaussian distribution over texton histograms computed from the 17 dimensional filter-response presented in earlier sections. The authors propose a method that merges codebook entries to result in a smaller more compact, though discriminative codebook.

Geometric and explicit shape models

The object-class models introduced so far mostly employ the bag of visual-words (or textons) model. They are based on relatively simple low-level features (*e.g.* colour patches or filterbank responses) and slightly more advanced and very successful orientation histograms (*e.g.* SIFT or HOG). Although the bag of visual-words model performs almost surprisingly well given the fact that it ignores any spatial relation between image features, it is understandable that the incorporation of such spatial relations can boost performance. Now a few such approaches are introduced.

Fergus *et al.* (2003) proposed an object-class model, called constellation model. Their approach is based on the work of Burl *et al.* (1998), which models objects as random constellations of parts. Each part has an appearance, relative scale and can be occluded or not. The object's shape is modelled by the mutual position of the constellation parts. Weber *et al.* (2000a) and Weber *et al.* (2000b) proposed a maximum likelihood unsupervised learning algorithm for the constellation model. Unlike in the BOW approach, here the connections between all parts are modelled and the model is fully parametrised in the sense that each part is represented as a distribution over appearance, scale and occlusion and the shape is modelled as a joint Gaussian density of the part's locations.

Fei-Fei *et al.* (2003) build on this constellation model and provide a description of a "hierarchical Bayesian" method for unsupervised learning of object categories using a mixture of constellation models. Its parameters are defined via a hyper distribution, thus accomplishing the hierarchical Bayesian idea. The feature extraction and modelling is a simplified version of Fergus *et al.* (2003). After learning the hyper parameters from images of objects a new object-class (distinct from the ones that were used for the hyper parameter estimation) can be learnt with very few sample images (*e.g.* 1 to 5).

Fergus *et al.* (2005b) simplify the original fully connected constellation model and reduces it to a simpler star model which is compared to the original approach and to a variation of the classification stage, where "exhaustive search" instead of extracted feature points is used. This exhaustive search is possible due to the highly decreased complexity of the star-model.

Kumar *et al.* (2004) describe a generative probabilistic model which falls into the same group of pictorial structure models as Fergus *et al.* (2003). In contrast to Fergus *et al.*, parts are explicitly learnt in a first step from moving objects in videos as rigid components of those objects. The proposed method enables the recognition of various deformable objects from images. Their approach extends the articulated pictorial structure of Felzenszwalb and Huttenlocher (2000) in a number of ways. The model includes both the outline and the enclosed texture of the part of the object that is modelled. The outline of the object parts is modelled using Chamfer distance and the texture is modelled with Gaussian mixtures. The structure of the graph is modelled using a completely connected Markov Random Field.

Leibe *et al.* (2004), successor of Leibe and Schiele (2003), use sparse raw image patches as the main feature. Each of those patches stores the relative translation with respect to the object centre. This corresponds to a star model as proposed by Fergus *et al.* (2005b), but it is centred on the object's centroid rather than one of the parts. During the classification stage the test-patches are matched to the codebook and a generalised Hough voting is used to retrieve an initial hypothesis for the object centre, which is determined using mean-shift on the voting space. The codebook is learnt by agglomerative clustering on images patches extracted from the training data. Each codebook entry is represented by a patch and a list of all corresponding object centres that occurred in the training images. Leibe *et al.* (2004) provide a thorough experimental evaluation and add the Minimal Description Length (MDL) criteria

to their previous work. MDL is an information theoretic formulation describing the preference for simpler explanations over more complicated ones and is used to remove wrong hypothesis, especially false positive detections of cars. Fritz *et al.* (2005) extend these more generative detection models and provide a system which integrates a representative and discriminant model for object category detection. The representative part is based on the Implicit Shape Model (ISM) introduced by Leibe and Schiele (2003). The discriminant part uses SVMs with local kernels. The ISM finds a set of promising hypotheses and estimates the support of those. As a second step the discriminative model discards false positive hypotheses. One of the interesting points is that both models use the appearance codebooks as internal representation and the SVM kernel uses a computationally efficient representation of the scalar product which is expressed as a codebook matching problem.

Both Shotton *et al.* (2005) and Opelt *et al.* (2006) propose geometrically very similar approaches which use boundary/edge information instead of image patches to perform the object classification. In both cases a star model with respect to the object's centroid is used to represent geometric properties of the object. One objective of Opelt *et al.* (2006) is to show that object category recognition is possible, solely based on information describing the boundary of the objects. The method uses class discriminative boundary fragments combined with a description of the object's centroid, similar to Leibe *et al.* (2004). To match the boundary fragments Chamfer distance and the pixel distance of the predicted and the real centroid are used. AdaBoost with weak detectors based on a few boundary fragments is used as a strong classifier. Each boosted classifier votes in a Hough voting space during the detection step. The votes are then accumulated in a circular search window around the candidate points, using Mean-Shift mode estimation. The proposed method results in good object category recognition and also returns a segmentation of the detected objects, based on the relevant boundary fragments. Shotton *et al.* (2005) use a variation of Chamfer matching called oriented Chamfer matching which takes the edge orientation into account. GentleBoost is employed to learn the features and the classifier. These approaches are very similar to Leibe *et al.* (2004) and differ in the use of boundary/edge features and boosting instead of patch based features and a probabilistic model. Leordeanu *et al.* (2007) also aim to model the object's shape. Their underlying feature is an edge point together with its normal. The shape is then modelled as a fully connected

graph. The proposed method performs very well compared to the state of the art.

Implicit shape models

The methods introduced in this section also aim to model object shape, but they are somewhat less explicit in the sense that they do not directly use pictorial structure like models or connections of object parts, but rather combine the information in a more implicit manner.

Lazebnik *et al.* (2006) introduce a spatial pyramid matching scheme that consists of BOW histograms computed in image sub-regions. This model thus extends the global bag of visual-words model by adding additional visual-word distributions computed over smaller sub-regions. It transfers the idea of the "pyramid match kernel" proposed by Grauman and Darrell (2005) to match two sets of features in feature space into the image space. This is done by quantising all features into $|V|$ types and consecutively matching those of the same type. Lazebnik *et al.* (2006) show improvement over the standard BOW model for the scene recognition and image classification tasks. Bosch *et al.* (2007a) build on top of this pyramid match kernel and use HOG instead of a SIFT and GIST [Torralba (2003)] like feature representation. They also learn the weighting of the different pyramid levels automatically and show how an appearance kernel can be combined with the spatial kernel.

Winn and Jojic (2005) introduce a model to learn object segmentation from unannotated images. The authors introduce a generative probabilistic model that combines low-level features as colour and edge information with top-down shape information that is learnt during training. The shape is modelled by means of a shape mask, which defines the probability for each pixels to lie inside the object in a canonical pose, and an edge mask which models the probability of edge occurrences as a Gaussian distribution over Canny edge strengths. Each of these masks is transformed using a deformation field, such that it can adapt to specific object instances that differ from the canonical pose.

Savarese *et al.* (2006) model appearance and shape jointly. They use correlations between textons to model the spatial relations between textons. The features are similar to the ones used in the Random Forest framework in Chapter 5.

Winn and Shotton (2006) introduce an approach to handle recognition of partially occluded objects. A so called layout consistent random field model is introduced. It is represented as

an ordered set of parts arranged on a grid as opposed to the previously described star- or fully connected models. During recognition an image patch is labelled as a specific part using a Random Forest. The random field allows deformation of the labels and thereby the recognition of slightly different shapes than present in the training examples.

2.1.3 Summary

This section gave an introduction into image representations, which are the underlying ingredient to all computer vision algorithms. We presented the concept of sparse and dense feature representations and explained several commonly used interest point detectors as well as feature extraction methods and various approaches for feature description. Then we introduced the nearest neighbour framework based on the example of the commonly used bag of visual-words or texton model. The nearest neighbour classifier based on texton histograms is the key part of Chapter 4, where we build on top of this exemplar based nearest neighbour method. We also gave an overview of related work that focuses on modelling object shape. The next section introduces further classifiers that are commonly used in the object recognition community.

2.2 Classifiers and Generative Models

Here we give an introduction into the machine learning techniques that are heavily employed throughout this thesis. *Classifiers* refer to a subset of pattern recognition methods that, in general, given a training set $\mathcal{T}$ consisting of n data points $\mathcal{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and their corresponding feature descriptions as well as classlabels $\{y_1, \ldots, y_n\}$ are able to predict the class-label y for a previously unseen test data point $\mathbf{x}$ [Duda *et al.* (2001), Bishop (2006)]. The feature description $\mathbf{x}$ can be of any form, but they are often represented in an Euclidean vector space. They can also be transformed into a different vector space using a transformation function Φ. This transformation into (often) higher dimensional vector spaces can improve the discrimination between classes. See Schölkopf and Smola (2002) for more details about kernel learning. Examples for such feature descriptors $\mathbf{x}$ are the previously introduced texton or visual-word histograms in the bag of visual-words model (BOW), or the 128 dimensional SIFT vectors embedded into the Euclidean vector space.

In the first two subsections we focus on two discriminative classifiers: (i) the support vector machines which are widely used in the machine learning as well as the computer vision community, and (ii) Random Forests which experienced a recent surge of applications to computer vision problems. In addition to *discriminative* classifiers there are *generative* classifiers. Some of the topic models presented in Section 2.2.3 fall into this category. Generally speaking, discriminative classifiers model the class probability given the feature description directly, *i.e.* $P(y = c|\mathbf{x})$, also called posterior probability. Generative classifiers model what is common between classes, *i.e.* $P(\mathbf{x}|y)$, called likelihood. The classification is then determined using Bayes' rule: $P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) \cdot P(y)}{P(\mathbf{x})}$, where $P(y)$ denotes the class priors and $P(\mathbf{x})$ the normalisation factor, also called evidence. Finally, Section 2.2.4 gives an introduction to energy minimisation techniques. We introduce the energy formulation for the image segmentation problem and present a few commonly used methods for its minimisation.

Before going into the details it should be noted that in addition to noise in the feature description $\mathbf{x}$, which is often to be assumed Gaussian, there can be noise in the class-labels y as well. The latter is often not modelled explicitly, but it is taken care of by the classifier being robust to noise. Brodley and Friedl (1999), for example, investigate different filters to remove *inconsistent* training data. Their method is inspired by work about outlier detection for regression models, but aims to remove data points that would be outliers in *any* model. We use a similar option for the support vector machine training in Chapter 6. Lawrence and Schölkopf (2001) on the other hand approaches the whole classification problem with a generative model, which enables modelling of the label noise explicitly. This method improves classification significantly in cases of large amounts of mislabelled training data, but does not help much if the noise level is low.

2.2.1 Support Vector Machines

Support vector machines (SVM) are a widely used tool in the machine learning and computer vision community. They were motivated by results of statistical learning theory and originally developed for pattern recognition they are now described in various books [Vapnik (1999), Schölkopf and Smola (2002)] and tutorials, *e.g.* Burges (1998). The basic idea is to learn a hyperplane in some feature space in order to separate the positive and negative training

examples with a maximum margin, thus called maximum margin classifiers. Also see Cortes and Vapnik (1995) for an early reference. There have been various extensions and improvements over the years. One example is the recent variation [Tsochantaridis *et al.* (2004)] to enable the learning of structured output spaces instead of simple two- or multi-class classification problems. This method was employed in the work of Blaschko and Lampert (2008) where structured output SVMs are applied to object detection and the output consists of a bounding-box and the class-label. Szummer *et al.* (2008) use the structured output SVM framework to learn parameters for CRFs in an efficient manner. Bach *et al.* (2004), Varma and Ray (2007) extend SVMs to multi-kernel learners, which combine various base kernels into an optimal domain-specific kernel.

In our work in Chapter 6 we use a variation of the standard form that takes into account different weights for positive and negative training errors. It was introduced by Morik *et al.* (1999) and the SVM training minimises the following equations:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_+\sum_{i:y_i=1}\xi_i + C_-\sum_{j:y_j=-1}\xi_j \tag{2.2}$$

$$\text{subject to} \quad y_l\left(\mathbf{w}^T\Phi(\mathbf{x}_l)+b\right) \geq 1-\xi_l, \tag{2.3}$$

$$\xi_l \geq 0, l = 1,\ldots,(n_+ + n_-) \quad . \tag{2.4}$$

Here the class-labels $y_l \in \{1,-1\}$. C_+ and C_- are the false classification penalties for the positive and negative data points, with $\boldsymbol{\xi}$ being the corresponding slack-variables. For most of our experiments we use a radial basis function kernel (RBF) based on a Gaussian kernel: $k(\mathbf{x},\mathbf{x}') = \exp\left(-\gamma\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right)$. Here $k(\mathbf{x},\mathbf{x}') = \Phi(\mathbf{x})\cdot\Phi(\mathbf{x}')$ is defined as the dot product of feature transformations Φ of the input data points. In the case of the Gaussian kernel the implicit definition of Φ describes a transformation into an infinite dimensional feature space, but the kernel function k can be used directly if the "kernel trick" is applied (the dual formulation of (2.2)–(2.4), see for example Bishop (2006) for a detailed introduction).

2.2.2 Random Forest Classifier

The Random Forest classifier is another discriminative classifier that has become very successful in computer vision. We explore their suitability to object segmentation in Chapter 5. Random

Forests are based on decision-trees [Quinlan (1993)] and have been introduced to the machine learning community by Amit and Geman (1997), Breiman (2001). Dietterich and Fisher (2000) did related work for constructing ensembles of decision-trees and compared methods based on bagging, boosting, and randomisation. Their popularity in the computer vision community arose mainly from the work of Lepetit and Fua (2006), Ozuysal *et al.* (2007), and a large number of papers have applied them to various supervised classification tasks [Marée *et al.* (2005), Moosman *et al.* (2006), Winn and Criminisi (2006), Bosch *et al.* (2007b), Deselaers *et al.* (2007), Yin *et al.* (2007), Shotton *et al.* (2008)]. Decision-tree classifiers have been known for a long time, but they have shown problems related to over-fitting and lack of generalisation. The main idea behind Random Forest is to try and mitigate such problems by

1. injecting randomness into the training of the trees, and
2. combining the output of multiple randomised trees into a single classifier.

Random Forests have been demonstrated to produce lower *test* errors than conventional decision-trees [Yin *et al.* (2007)] and performance comparable to SVMs in multi-class problems [Bosch *et al.* (2007b)], while maintaining high computational efficiency. Sharp (2008) showed how to implement them on a GPU very efficiently. We now summarise the appealing features of Random Forests as well as laying down the notation before we give a detailed introduction into the training and the parameters of the Random Forest classifier. This is followed by a discussion of the classification step. Finally, we discuss some theoretical properties and pre-conditions for Random Forests.

Random Forest features and notation

It can be advantageous to use Random Forests over, for example SVMs, because of the following properties:

1. their computational efficiency in both training and classification
2. independence of the trees allows for easy implementation and parallelism
3. their probabilistic output
4. the seamless handling of a large variety of visual features (e.g. colour, texture, shape, depth etc.)

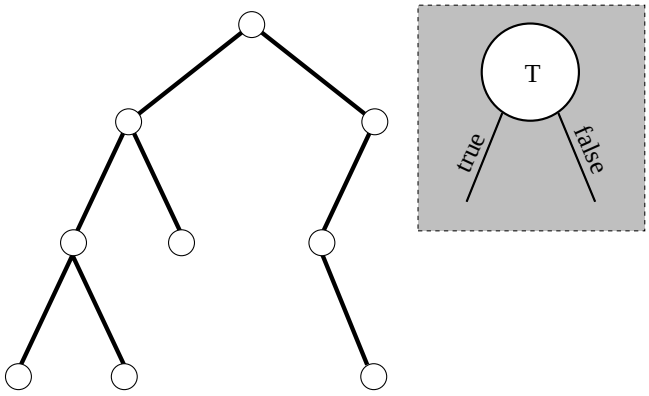


Figure 2.7: Decision-trees and node-tests. Unbalanced binary decision-tree with its node-tests T. In a general binary decision-tree the node-test $T : D \rightarrow \{true, false\}$ can be any function from the input domain D into $true$ and $false$. It is common to use a function $t_p : D \rightarrow \mathbb{R}$ and a threshold λ as a special case of T.

5. the inherent feature sharing of a multi-class classifier (see also Torralba *et al.* (2004) for feature sharing in boosting).

In this work we focus on *binary* decision-trees as shown in Figure 2.7 as the basic part of our Random Forests. The node-tests of these decision-trees can be of the most general form $T : D \rightarrow \{true, false\}$. Commonly, functions of the form

$$t_p : D \rightarrow \mathbb{R} \tag{2.5}$$

together with a threshold λ are used. This defines the node-test as:

$$T = \begin{cases} t_p < \lambda & \text{go to left child} \\ \text{else} & \text{go to right child} \end{cases} . \tag{2.6}$$

Note that the underlying features (*i.e.* D) can be of general nature, *e.g.* RGB, HOG or the output of a filter bank. Also the function t_p is of a very general nature: it could be a sum of feature responses in a defined spatial extent (see Figure 5.1 on page 95), a linear classifier, the output of another tree, the output of AdaBoost, to give a few examples.

Training the Random Forest

Given the general definition of decision-trees, their structure and decision nodes are learnt discriminatively as follows. Starting from the root, given the labelled training data, the function

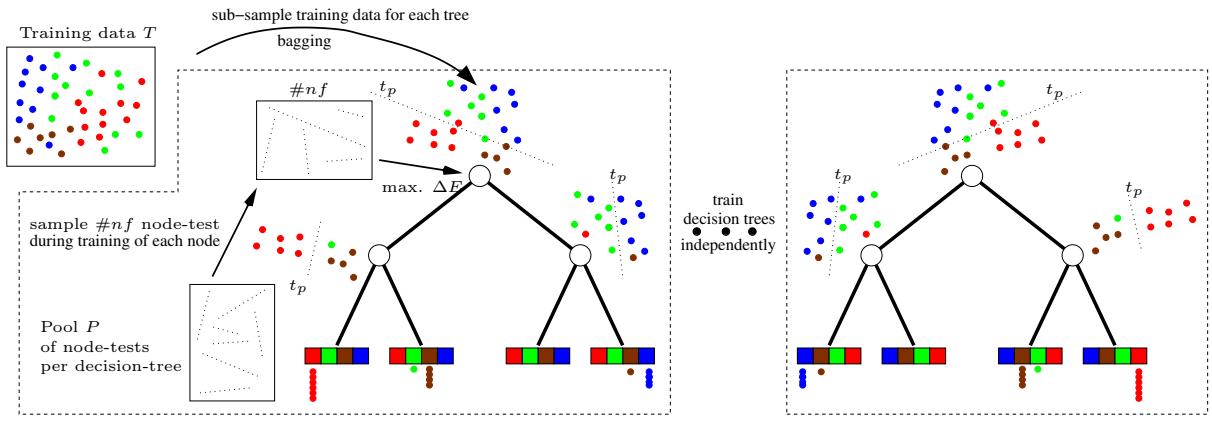


Figure 2.8: Random Forest. Illustration of the Random Forest classifier in the 2D case with a linear node-tests. Each decision-tree is trained independently. The training data for each tree is sub-sampled from the pool of all training data (bagging). During training of a specific node a number of node-tests is sub-sampled from the pool of all those functions and the one that maximises the information gain ΔE is selected as *node-test*. This corresponds to splitting the feature space into regions that primarily contain training data of one class. The leaf nodes represent the empirical class distribution of the corresponding feature space regions.

t_p and threshold λ which maximise the information gain ΔE are found.

$$\Delta E = E(P) - \sum_j \frac{|Q_j|}{|Q|} E(Q_j)$$

where Q is the set of data points at the given node and $Q_j \subsetneqq Q$ are the *left* and *right* subset caused by partitioning the data with $t_p < \lambda$. $E(Q) = E(\mathbf{q}) = -\sum_{i=1}^{N} q_i \ln(q_i)$ with $\mathbf{q}$ being the histogram over the classes of the data in Q. This basic idea is similar to the C4.5 algorithm introduced by Quinlan (1993) for "ordinary" decision-trees and it aims to maximally separate the classes in Q. The algorithm proceeds iteratively with the left and right subsets Q_j at the children nodes until Q_j is empty or a threshold for $E(Q)$ or ΔE is reached. Sometimes it is beneficial to "normalise" the training data during computation of ΔE, by weighting each training point with its inverse class prior probability. Note that this is different to normalising the empirical class posteriors in the leaf nodes.

See Figure 2.8 for an illustration of how training works in a two dimensional case. The node-test t_p and λ define separating hyperplanes/lines. During training of the tree each node only "sees" a *randomly* chosen subset of the entire pool P of possible node-tests. Training is achieved by finding the node-test t_p and threshold λ which yields maximum information gain ΔE within this restricted, randomised search space. The "randomisation" can be tuned by the size and composition of the pool P and the amount of optimisation of λ (*e.g.* Lepetit and

Fua (2006), Moosman *et al.* (2006) do not optimise λ, but pick it randomly). The training data is sub-sampled (bagging) to increase independence of the trees Breiman (2001) and reduce training time. During training the empirical class posterior distributions are stored in the leaf nodes, *e.g.* in the form of histogram counts over the class-labels of the training data as shown in Figure 2.8. One advantage of Random Forests is that the decision-trees in the Random Forest can be trained in parallel. Crucial, however, is that they are "independent" (randomised enough), *i.e.* result in independent classifications of the data (see Breiman (2001)).

Random Forest Parameters The following summarises the set of parameters that guide the training of the Random Forest.

1. **Number of decision-trees** $|RF|$ in the Random Forest: The number of decision-trees greatly influences the performance of the Random Forest. Additional independent/randomised decision-trees add further information over the training data, as each tree partitions the feature space into different cells and collects the empirical class posteriors for those cells in the leaf nodes. Our experiments show that the performance increases the more decision-trees are used, but the improvement is small for more than 15–20 decision-trees are used (see Section 5.4.2).

2. **Node-test pool**: The pool of node-tests P describes the set of possible functions that can be selected for a node during training. The following parameters describe the possible options for the node-tests described by (2.5).

   (a) number of functions $\#nf$ created for the "per node" pool: This is the main parameter to influence the "randomness" of the decision-trees. If $\#nf = 1$ there would be no optimisation of the information gain ΔE and the decision-trees would define a random partitioning of the feature space. The bigger $\#nf$ the more the partitioning will be driven by discriminating between the object classes, as for each node the *one* node-test out of $\#nf$ tests that maximises the information gain ΔE is selected. These $\#nf$ node-tests are sampled from the initial pool of node-tests P that is created for each decision-tree.

   (b) type of low-level features used: This defines the type of features $\mathbf{x} \in D$ used to compute the response t_p. It can be based on texton histograms or low-level features

such as RGB, HOG, filter-banks, or textons.

3. **Decision-tree parameters**: These parameters describe properties of the decision-trees only.

   (a) maximum depth of each tree: Experiments indicate that deeper trees tend to improve performance, and thus the depth is mostly determined by computational and memory considerations (see Section 5.4.2), but depending on the specifics of the implementation and the number of trees in the forest it can also lead to overfitting. Ho (1998), however, proposes to learn the trees to maximal depth, *i.e.* each leaf node only contains one data point, thus resulting in 100% performance on the training data.

   (b) information gain or entropy stopping criterion: In order to avoid "over"-fitting of the tree, *i.e.* partitioning of feature space areas into areas where there is no further partitioning necessary, these two criteria can be used to stop expansion of the nodes before the maximum depth is reached.

Classifying with the Random Forest

During testing the data is classified independently by each decision-tree. Each data point is sent to the left or right child node depending on the evaluation of the node-test at a specific node until it reaches a leaf node. This classification results in the assignment of the empirical class posterior distribution to the test data point. It is often better to not use the empirical class posteriors directly, but to weight them with the class prior probabilities. There are two main methods to combine the multiple class posterior distributions $P_i(c|\mathbf{x})$ with $i \in \{1, \ldots, |RF|\}$: (i) The *Product of Experts* (PoE) introduced by Hinton (1999) and (ii) the *Mixture of Experts* (MoE) that is commonly used, *e.g.* Gaussian mixture models. Breiman (2001) uses a plurality voting scheme instead, but as we want to retrieve the class posterior distribution voting is not suitable for our work. In the following Z denotes the normalisation such that $P(c|\mathbf{x})$ is a proper probability distribution.

Product of Experts. The resulting class probability is modelled as the product of the individual probabilities (in our case all decision-trees have the same weight):

$$P(c|\mathbf{x}) = \frac{\prod_i^{|RF|} P_i(c|\mathbf{x})}{Z} \quad .$$

In a sense, in a PoE each individual tree can "veto" a specific class by assigning a low probability to it. The PoE can also be seen as a MoE in log space, which might help to understand the conceptual differences better. Hinton (2000) argues that one advantage for PoEs lies in the modelling of probability distributions in high dimensional feature spaces. Each expert can focus to restrict only a subset of dimensions and the product of all the experts will then constrain all the dimensions. The PoE can produce much sharper probability distributions than the individual experts, this is not possible for MoEs. It seems to be important to learn the model as a *whole* to ensure that not all experts agree on unobserved data [Hinton (1999)]. In the Random Forest case, however, the individual decision-trees are trained independently and the forest is *not* learnt as *one* model. This makes the MoE more suitable for our task.

Mixture of Experts. The MoE is also called averaging classifier as the individual probability distributions are averaged:

$$P(c|\mathbf{x}) = \frac{\sum_i^{|RF|} P_i(c|\mathbf{x})}{Z} \quad .$$

Here each individual tree has a bigger influence in voting for a specific class. Amit and Geman (1997) use the MoE to recognise shapes, *e.g.* distorted letters and digits. Biau *et al.* (2008) show that the voting and averaging classifiers are consistent and also investigate the consistency of Random Forests. It turns out that if the individual decision-trees are consistent the averaging classifier is consistent as well. Ho and Kleinberg (1996) also proposes the MoE. Their work is based on Stochastic Discrimination (SD) [Kleinberg (1990)]. Another justification for using the averaging of decision-trees can be derived from a Bayesian view. Let us assume a distribution $P(\mathtt{T}|\mathcal{T})$ that describes the probability for each decision-tree $\mathtt{T}$ to be sampled from all possible trees given the training data $\mathcal{T}$. Then the class posterior for a data point $\mathbf{x}$ is given

by:

$$P(c|\mathbf{x}) = \sum_{\mathtt{T}} P(c|\mathbf{x}, \mathtt{T}) \cdot P(\mathtt{T}|\mathcal{T}) \quad .$$

Random Forests approximate this by selecting a subset of all possible decision-trees during training and assume equal likelihood for each of those *trained* trees and zero probability for all other decision-trees.

Discussion

Although Random Forests have experienced great success in the computer vision community as well as the machine learning community in general the underlying principles don't seem to be fully understood [Biau *et al.* (2008)]. Ho (1998) investigates some of the principles that allow Random Forests to learn arbitrarily complex decision boundaries while maintaining good generalisation capabilities. The complex decision boundaries result in almost 100% training accuracy which we also confirmed in our experiments. Many of these observations can be explained by Stochastic Discrimination (SD) theory introduced by Kleinberg (1990). Kleinberg (2000) gives a more algorithmic view of SD. Random Forests are a special case of SD that starts with guaranteed *enrichment*[1] and *uniformity*[2] (if the trees are fully-split) and seek optimisation on *projectability*[3] [Ho and Kleinberg (1996), Ho (1998)]. Other methods start with highly projectable classifiers with minimum enrichment and try to optimise uniformity [Ho and Kleinberg (1996)]. Chipman *et al.* (2006) introduce BART - Bayesian "sum-of-trees" model, which is similar to the MoE or boosting [Freund and Schapire (1996)], but is completely defined by a statistical model. The model is based on decision-trees, but the training is performed by sampling from the posterior distribution using Markov chain Monte Carlo [Bishop (2006)]. A

[1] *Enrichment* denotes the fact that the partitions of the subspace contain a larger fraction of data points of one class than the other classes. This is achieved by the maximisation of information gain in the Random Forest framework.

[2] *Uniformity* means that all data points of a given class must be "viewed equally" by the full set of weak classifiers, *i.e.* the weak classifiers should be unbiased for any particular training point. In a sense the weak classifiers are not really classifiers, but describe subsets of the feature space with a slightly higher than average class distribution (*enrichment*). In a fully trained decision-tree each leaf node only contains one training point thereby achieving maximal enrichment and also fulfilling uniformity as each training point is covered and represented in exactly one leaf node.

[3] *Projectability,* also known as generalisation, refers to the fact that the weak classifiers (node-tests) should also capture points that are considered "neighbours" of specific training points, *i.e.* it is important that the "decision boundaries" are not too tight around training points. Limiting the depth of the trees can ensure this to some extent.

prior is defined for each of the parameters such as the depth of the trees and the values of the leaf nodes.

2.2.3 Topic Models

There is a range of methods that can be summarised as topic models. Their underlying idea is to find a set of reoccurring topics given a body of documents $\mathcal{D} = \{d_1, \ldots, d_N\}$. For example, in the case of text retrieval some documents might talk about penguins and related topics such as birds and Antarctica. Others might talk about cars, racing and related things. Each document is made up from terms of a vocabulary $\mathcal{W} = \{w_1, \ldots, w_M\}$ and each document can be presented using the common bag of words model where the order of words in the documents is disregarded and the corpus can thus be described with a document-word co-occurrence matrix N. Topic models try to identify the words that, *e.g.* talk about penguins, Antarctica, or cars and thus define common topics that occur in several documents. Intuitively each document is described as a distribution over topics and each topic defines a distribution over vocabulary words. Now the specifics of three different topic models are introduced.

The idea of Latent Semantic Analysis (LSA) is to map the documents from the M dimensional space over words onto a lower dimensional space over topics. In LSA this is performed by singular value decomposition of the co-occurrence matrix $N = U\Sigma V^t$ [Deerwester *et al.* (1990)]. Now the k vectors for the largest singular values are used to define this mapping onto the k dimensional topic space, where each document is described by a topic weight vector and each topic by a word weight vector. This mapping is optimal with respect to the Frobenious norm. Lee and Seung (1999) use *non-negative* matrix factorisation to learn parts of objects. This imposes different constraints on the decomposition than LSA for example.

Hofmann (2001) introduce a probabilistic extension to this feature reduction – *probabilistic Latent Semantic Analysis* (pLSA). Using the common notation for pLSA we have $P(w|d) = \sum_z P(w|z)P(z|d)$ where z are the topics. This factorisation is visualised in Figure 2.9. Each topic represents a distribution of words occurring in this topic. Those topic word distributions $P(w|z)$ are learnt together with the document topic distributions $P(z|d)$ by minimising the Kullback-Leibler divergence of the model's document word distribution $P(w,d) = P(d)\sum_z P(w|z)\,P(z|d)$ to the real document word distribution N given by the

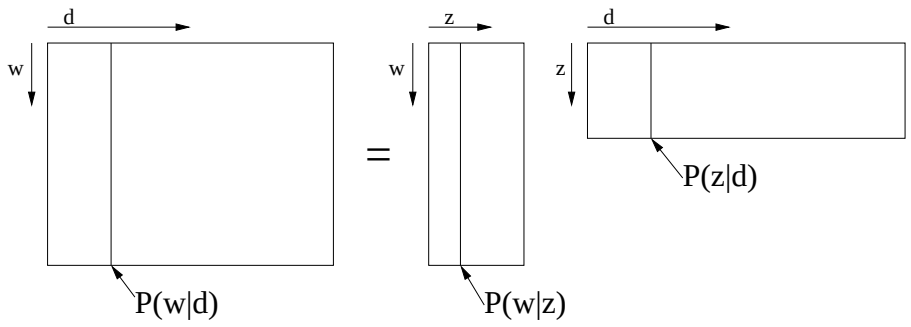


Figure 2.9: pLSA models word frequencies in a document $P(w|d)$ as topics in a document $P(z|d)$ given the word distribution for a topic $P(w|z)$. pLSA then minimises this equation with respect to the KL divergence.

documents. The model parameters $P(w|z)$, $P(z|d)$ and $P(d)$ are estimated using Expectation Maximisation (EM) [Bishop (2006)]. EM alternates an *expectation* (E) step where the posterior probabilities are computed for the latent variables given the current parameters and a *maximisation* (M) step where the model parameters are updated given the new posteriors for the latent variables.

Blei *et al.* (2003) describe *latent Dirichlet allocation* (LDA), a generative probabilistic model for collections of discrete data. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modelled as a finite mixture over an underlying set of topics. The authors compare the structure and performance of LDA to the unigram model[4], the mixture of unigram model and pLSA. LDA is far less sensitive to over-fitting than pLSA and mixtures of unigrams. The paper also provides a good geometrical representation of those models. One improvement over pLSA is that LDA provides a probabilistic model at the level of documents. pLSA does not model a probabilistic distribution of the topics of documents, which leads to a problem when assigning probabilities to a document that is not contained in the training set. Blei *et al.* (2003) also introduces a method to estimate the parameters (approximately). The method uses convexity-based variational inference and works in an EM manner. Teh *et al.* (2004) introduce hierarchical Dirichlet Processes (HDP) which avoid the need to explicitly specify the number of topics as pLSA and LDA do and additionally model relationships between topics.

[4]Under the unigram model the words in a document are drawn independently from a *single* multinomial distribution.

Discussion Topic models can be seen as dimensionality reduction techniques. However, the newer versions such as LDA and HDP are full generative models. In contrast to other dimensionality reduction techniques as principal component analysis (PCA) and also LSA they are able to model non-linear dependencies between topics and documents.

2.2.4 Energy Minimisation for Multi-Label Image Segmentation

Image segmentation as introduced in Chapter 1 can be formulated in terms of an energy minimisation problem. The idea is to find the optimal segmentation into constituent object regions given a cost or probability for each pixel to belong to a certain object-class or not. This corresponds to assigning an object class-label to each pixel, which then defines the image segmentation. The energy formulation takes into account that neighbouring pixels tend to belong to the same object unless there are strong gradient edges in the image (discontinuity preserving energies).

This multi-label problem can be modelled using Markov Random Fields (MRF). A MRF models the joint probability of an image and its labels $P(\mathbf{c}, \mathbf{I})$ in a generative framework [Geman and Geman (1984), Li (2001), He *et al.* (2004)]. One problem with MRFs is the inherent estimation of the partition function, as we eventually want to retrieve $P(\mathbf{c}|\mathbf{I})$. A Conditional Random Field (CRF) [Lafferty *et al.* (2001)] models $P(\mathbf{c}|\mathbf{I})$ directly. Unlike the MRF the CRF model can depend on arbitrary non-independent characteristics of the observation [He *et al.* (2004)], whereas the MRF as a generative model is forced to account for dependencies in the image. He *et al.* (2004) describe an approach to incorporate contextual features into the CRF for the task of image segmentation by means of a so called "multiscale conditional random field" (mCRF).

In this work we are interested in graphical models that capture the pixel structure of images. Therefore, we model an image $\mathbf{I}$ as a CRF where each pixel corresponds to a node and each node is connected to its four neighbours. This defines an undirected graph with edges $\{ij \in \mathbf{I}\}$, where i and j denote the pixels in the image. Different neighbourhoods such as an eight neighbourhood can be used too. The labelling problem can then be formulated as an energy minimisation problem [Boykov *et al.* (2001)], where $E(\mathbf{c}, \mathbf{I})$ corresponds to the negative log-

likelihood ($-\log P(\mathbf{c}|\mathbf{I})$) and is modelled as follows:

$$E(\mathbf{c}, \mathbf{I}) \quad = \quad E_{smooth}(\mathbf{c}) + E_{data}(\mathbf{c}, \mathbf{I}) \tag{2.7}$$

where E_{smooth} measures the smoothness of the labelling $\mathbf{c}$ and E_{data} the disagreement between $\mathbf{c}$ and the observed data $\mathbf{I}$. The smoothness term is often described in the form of pairwise energies between the neighbouring pixels, *i.e.* $E_{smooth}(\mathbf{c}) = \sum_{ij} V_{ij}(c_i, c_j)$ where $V_{ij}(c_i, c_j)$ defines the cost of disagreeing labels for neighbouring pixels. Instead of pairwise interactions higher order cliques have been used in the past, but the lack of efficient optimisation techniques limited their applicability. The minimisation of this energy function (*i.e.* the negative log-likelihood) corresponds to the maximum posterior probability (MAP) estimate and the restriction to pairwise energies implies the Markov property, *i.e.* $P(c_i|\mathbf{I}, c_j, i \neq j) = P(c_i|\mathbf{I}, c_j, i \sim j)$ where $i \sim j$ denotes that i and j are neighbours.

Discussion Given the energy formulation in (2.7) there are various recent algorithms that can be used for finding the minimum energy labelling corresponding to the MAP estimate. We now discuss a few common techniques and their relations.

Lets first look at a simpler problem with a tree structured graph instead of the grid structure normally used for image labelling. Inference in these models can be performed exactly using variations of *dynamic programming*. The first example for dynamic programming is the well known Viterbi algorithm [Viterbi (1967)]. The so called max-product algorithm[5] is another more general version. See Cormen *et al.* (2001) for a more recent introduction. Similar methods can be used to minimise these energy formulations in arbitrary graphs. The *junction tree algorithm* [Lauritzen and Spiegelhalter (1988)] is a method to perform exact inference in arbitrary graphs, but its complexity grows exponentially with the size of the largest clique for discrete variables. *Loopy belief propagation* is the direct extension of the max-product algorithm to general graphs, but the quality of the solutions and its efficiency is limited. See Bishop (2006) for a good overview of this area. We use convergent (or sequential) tree-reweighted message passing (TRW-S) [Kolmogorov (2005)] in Section 5.4.4 that is based on the work of Wainwright *et al.* (2002). TRW-S is related to message passing algorithms such as belief propaga-

[5] In Bishop (2006) it is called max-sum algorithm as the product is often replaced by a sum over the logarithm of the probabilities to avoid numerical problems.

tion and although it is not guaranteed to find the global optimum on multi-label problems with discontinuity preserving pairwise energies it is able to provide a lower bound on the energy in addition to a very good local minimum.

Another direction of minimisation methods is based on graph-cuts (min-cut/max-flow algorithms) and uses the α-expansion or $\alpha\beta$-swap methods for minimisation of multi-label problems [Boykov *et al.* (2001), Kolmogorov and Zabih (2004)]. Recent work of Kohli *et al.* (2007, 2008) shows how certain higher order clique potentials can also be minimised in this framework and the authors underline the strength of their approach on an example of image segmentation. Graph-cuts methods do not guarantee a global optimum, similar to TRW-S, for general pairwise energies, but recent advances such as quadratic pseudo-boolean optimisation (QBPO) [Rother *et al.* (2007)] are able to deal better with non-submodular energies.

Szeliski *et al.* (2008) gives a comprehensive study of several energy minimisation methods and evaluates them on real world problems. TRW-S proved itself very useful in those experiments, and does not require constraints on the energy function as move making graph-cuts methods do in order to guarantee optimal moves.

2.3 Object Recognition

The goal of object recognition systems is to recognise objects belonging to a set of (usually) specified object classes. In the simplest setup this would just mean the classification of a whole image, *i.e.* returning if a specified object-class is present or not. Object detection refers to the more challenging task of simultaneously detecting the position (*e.g.* bounding box) of the object as well as its class. Recently the pascal Visual Object Class (VOC) challenges were organised to provide publicly available image datasets together with annotations and a well defined evaluation protocol [Everingham *et al.* (2006, 2007, 2008)]. This facilitated the comparison of different algorithms on a standard testbed and the corresponding reports present the wide range of submitted methods.

Some of the more successful object detectors are crafted towards the detection of single object-classes, such as the face detector by Viola and Jones (2001). It provides a very fast face detection system based on AdaBoost [Friedman *et al.* (2000)]. Their weak classifiers are Haar like differences of the sum of pixel intensities in adjacent rectangles. The resulting strong

classifier is integrated into a cascade of classifiers. A first simple classifier discards non-face regions with a low false negative rate and thereby reduces the computational cost. Then a better computationally more expensive classifier is applied to the remaining sub-images in the next out of possibly many steps. Another well known object detector for one specific class is the person detector by Dalal and Triggs (2005). As it was already pointed out in Section 1.1 it is often helpful or even necessary to take context into account. Not many current approaches explicitly model context for object recognition. However, He *et al.* (2004), for example, use context in a CRF framework to improve image segmentation. Kumar and Hebert (2005) use a hierarchical conditional random fields to model between object context (*e.g.* keyboard and mouse next to a monitor) and region context (*e.g.* sky tends to be at the top of an image). Rabinovich *et al.* (2007) also use a CRF based model that aims to maximise the object label agreement with respect to contextual relevance and Singhal *et al.* (2003) use the detection of typical attributes, such as grass, sky, water or snow, and their spatial relationships in the images for scene understanding. The work of Sivic *et al.* (2005) aims to discover objects in unlabelled image sets and thus represents a slightly different area of object recognition. They use the bag of words representation on vector quantised SIFT-like feature descriptors and employ the pLSA method to discover topics/object-classes. After the object-classes are discovered the distinctive visual-words for each class can be identified in the images, thence describing the rough location of the objects. Many of the methods already introduced in Section 2.1.2 also focus on the object detection task.

2.4 Segmenting Images into Object Regions

As a significant part of this thesis focuses on semantic image segmentation, *i.e.* segmenting an image into its constituent semantic regions and automatically labelling each region as belonging to an object-class, we now give an overview of related work. For image segmentation it needs to be distinguished between *bottom-up* and *top-down* approaches. Bottom-up segmentation usually refers to segmentation methods that operate purely based on the image content, and take into account image colour, colour gradients and change of texture. Segmenting natural images in a bottom-up fashion automatically has a long history, but has not been that successful – see Sharon *et al.* (2001) for a recent example and earlier references. A purely bottom-up

segmentation will tend to separate the image into many different regions rather than recognising an object, *e.g.* cow, as a single coherent entity. We give a short introduction of a few bottom-up segmentation methods in Section 4.5.1 and focus on top-down approaches in the remainder of this section. Many of the methods described here employ the previously introduced CRF models and various energy minimisation techniques.

There are a few distinctions to be made about image segmentation. Some methods just return a foreground background segmentation. Other methods generalise the segmentation problem and try to return more complete descriptions such as image parse trees. Storkey and Williams (2002), for example, use so called position encoding dynamic trees to perform image labelling, a generalisation of the segmentation problem where an image parse tree is generated for the image. Zhu *et al.* (2007) is related to image parsing and uses a hierarchical "graph" structure for detection, segmentation and parsing of articulated deformable objects. Li *et al.* (2009) propose a hierarchical generative model for the simultaneous segmentation of scenes as well as recognition of objects and assignment of tags to the image. Some people use multiple bottom-up segmentations to define larger image regions that facilitate the learning of for example consistently re-occurring objects as in Russell *et al.* (2006). The following two paragraphs focus on interactive *image segmentation* and more automatic methods that aim to segment the image into its object regions, *object segmentation*.

Interactive image segmentation. Image segmentation is an important problem in computer vision and has been addressed in many works. Interactive segmentations describes a semi-automatic approach where the user selects fore- and background using a rectangular mask or a few brush strokes. This defines the background and more specifically a colour histogram of the background. An energy function that takes the posterior of each pixel given the histogram as well as intensity differences of neighbouring pixels into account is defined. Boykov and Jolly (2001) use graph-cuts to find a global minimum of this energy function and thus a segmentation into foreground and background regions. Rother *et al.* (2004) extend this version of interactive graph-cuts by introducing an iterative procedure. After the image is segmented new priors (colour histograms) for foreground and background are computed and this updated energy function is minimised. Criminisi *et al.* (2008) propose an efficient approximate energy minimisation technique for the fast segmentation of n-dimensional images.

Object driven segmentation. Some of the object recognition algorithms introduced in Section 2.1.2 provide rough object segmentations: Shotton *et al.* (2005), Opelt *et al.* (2006) based on the boundary fragments or Leibe *et al.* (2004) by considering all patches, that contain a specific pixel, and their contribution to the object detection hypothesis.

Now we first present object segmentation methods that make use of the previously introduced energy formulation and then introduce work that makes strong use of bottom-up segmentations. Finally, a quick overview of some ideas to combine bottom-up and top-down segmentations is provided.

TextonBoost [Shotton *et al.* (2006)] describes an approach of learning a discriminative model of object classes, which incorporates appearance, shape and context information efficiently. The shape filter represents shape, texture and context and is learnt by a version of AdaBoost [Friedman *et al.* (2000)], which uses shared features. A conditional random field is used to "connect" the shape, colour and edge features to provide the final segmentation. Whereas *TextonBoost* focuses on the estimation of the unary term using a strong discriminative boosting classifier, Kumar *et al.* (2005) describe a method that uses top-down shape information in the form of pictorial structures and unmodelled information in the form of pixels that are similar to the objects' pixels. A Markov random field (MRF) is formulated to combine both of these parts and the inference in this so called Object Category Specific MRF is performed by a graph-cuts algorithm (see Section 2.2.4 for details). Verbeek and Triggs (2008) use a CRF approach that incorporates local (SIFT based textons) and global image features. They also show how their method can handle missing groundtruth labels during training. These approaches show that in addition to a strong classifier as in *TextonBoost* it is useful to also incorporate more global shape as in Kumar *et al.* (2005) or global image features as in Verbeek and Triggs (2008).

The following two approaches make use of image inherent low-level information by means of bottom-up segmentations which group pixels that are likely to belong to the same object together, thereby simplifying inference of the object regions. Csurka and Perronnin (2008) use sparse logistic regression to classify image patches that are described by a soft assignment to textons, where the quantisation is performed using a Gaussian mixture model. Bottom-up segmentation is used in order to get a more consistent pixelwise classification. See Section 4.5 for our work on combining bottom-up segmentation with top-down classification. Kohli *et al.* (2008)

also uses a CRF based approach and focuses on the combination of bottom-up segmentation using higher order cliques defined by multiple such segmentations. *TextonBoost* is used to define the unary potentials. Lempitsky *et al.* (2008) use a combination of branch and bound and graph-cuts to find global optima for a wide range of energies. They also apply their method to the segmentation problem.

Rather than the CRF based methods, which focus on "classifiers" that return good unary potentials (E_{data}), the methods in this paragraph are more related to object detection methods as they detect parts of the object and build on those detections to infer the segmentation. Borenstein and Ullman (2002) introduce a top-down approach that uses pre-learnt object fragments that consist of a template and a labelling. During testing the objects in the image are "covered" with those templates, thus defining a segmentation. The work of Shotton *et al.* (2005), Opelt *et al.* (2006) uses a similar approach where the fragments are strict edge fragments rather than patch based as in Borenstein and Ullman (2002). The patch based approach is similar to Leibe *et al.* (2004), who are also able to retrieve a segmentation. Borenstein and Ullman (2004) is an extension of Borenstein and Ullman (2002) and does not require segmented training data or bounding boxes, but can learn the fragment templates from unsegmented training images. Borenstein *et al.* (2004) combine bottom-up and top-down segmentation by minimising a global energy function using the max-product algorithm. In addition to the image segmentation this also results in a confidence map that can point out regions which require further refinement. All these methods focus on the segmentation of one object-class (*e.g.* horse) versus the background-class, but their ideas could contribute to multi-class image segmentation and supplement methods that are based on strong unary potentials and CRFs.

2.5 Conclusion

The ultimate goal of visual content analysis would be to explain each pixel in an image. In order to reach this goal it seems inevitable to take global information into account. Many of the current approaches work locally (*e.g.* sliding-window based) and do not take the global context into account. For example the person detector Dalal and Triggs (2005) or face detector [Viola and Jones (2001)], albeit these are very successful demonstrations of the possibilities local methods have. Some of the methods like Shotton *et al.* (2005), Opelt *et al.* (2006),

Leibe *et al.* (2004) and Fergus *et al.* (2005b) aim to model object shape more explicitly, *i.e.* not only local appearance, however systems that are able to give a comprehensive understanding of the whole image content, including a hierarchical representation of objects and their parts are yet to come.

After the introduction of the datasets that are used for the segmentation work in this thesis, we present our extensions and variations to the segmentation algorithms discussed here.

Chapter 3

Object Segmentation Datasets

This chapter introduces the datasets that are used to evaluate the semantic segmentation methods of this thesis. It is very important to have good standardised datasets that are used in other work, such that results can be compared to other methods without the need of implementing them. The two datasets discussed here are widely used for the semantic segmentation task and a comparison to other work is therefore easily possible.

First we introduce the MSRC (Microsoft Research Cambridge) datasets. The MSRC datasets [Criminisi (2004)] consist of a main dataset featuring 21 object-classes and a 9-class subset. Next, the more recent and much more challenging datasets suitable for object segmentation are presented. They were created for the PASCAL visual object-class (VOC) challenge [Everingham *et al.* (2006, 2007, 2008)]. Both the VOC2007 and VOC2008 dataset versions include labelled groundtruth that assigns one out of the 20 object-class-labels to each pixel, and assigns background if none of the classes match. Finally, we present the performance measures that are used throughout this thesis to evaluate our algorithms and to compare to previous work.

3.1 MSRC Datasets

The MSRC dataset [Criminisi (2004)] was introduced in Winn *et al.* (2005). The full dataset provides labelling for 21-classes. Figure 3.1 gives an idea of the kind of pixelwise segmentation that is available for the training and test images of this dataset. The whole dataset includes labelled objects of the classes *aeroplane, bicycle, bird, boat, body, book, building, car, cat, chair,*

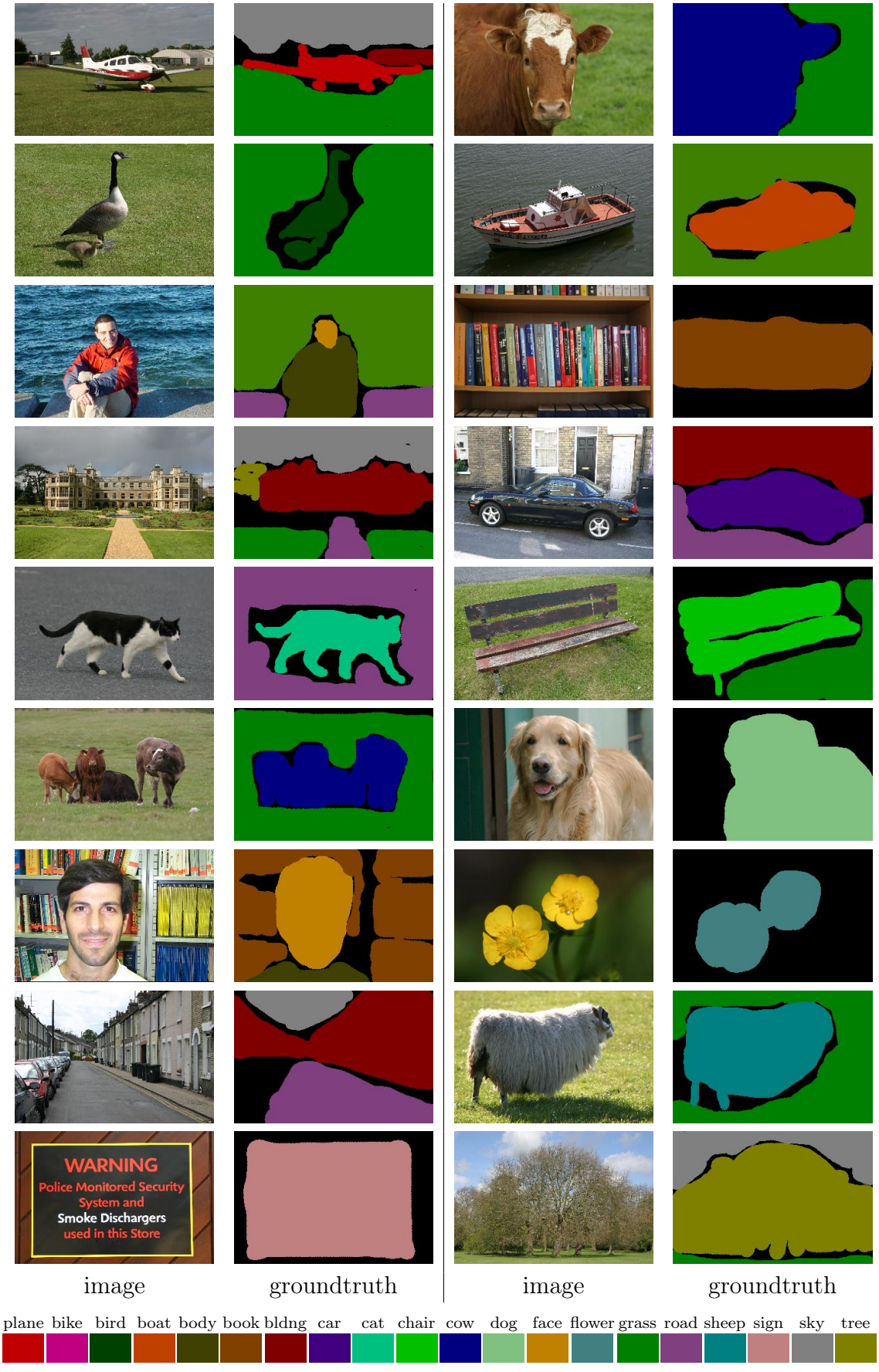


Figure 3.1: Rough pixelwise groundtruth segmentations and the MSRC images are shown. Each object-class is represented by a unique colour. Object boundaries and classes not belonging to one of the 21 object-classes are marked as "void" (black). For each of the classes *aeroplane, bicycle, bird, boat, body, book, building, car, cat, chair, cow, dog, face, flower, grass, road, sheep, sign, sky, tree, water* at least one example is given.

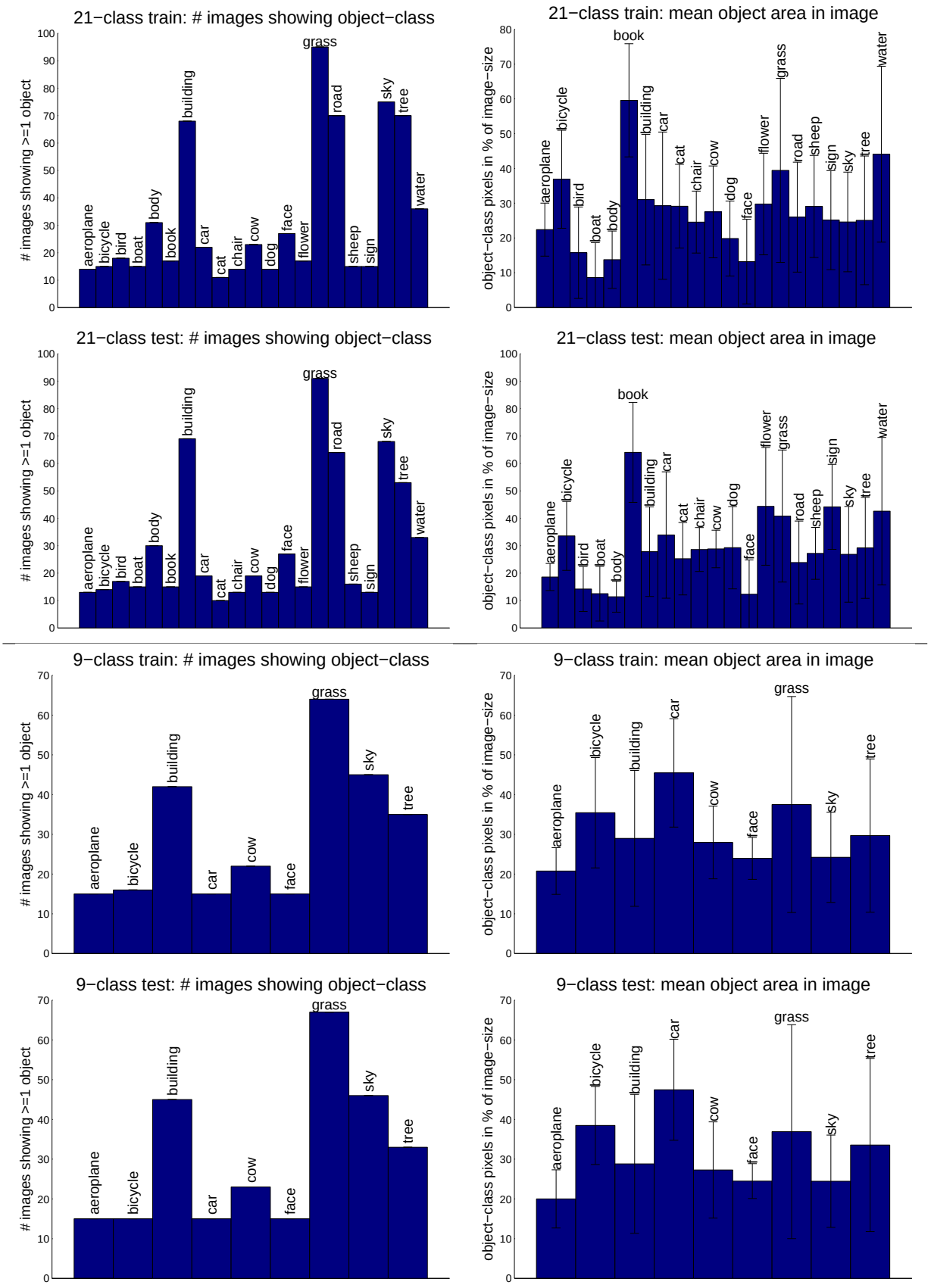


Figure 3.2: Object-class statistics for the MSRC datasets. Shown are the number of images that contain each object-class and the average number of object-class pixels in percent of the number of image pixels (± one standard deviation indicated by the black error-bars around the blue mean).

cow, dog, face, flower, grass, road, sheep, sign, sky, tree, water. Pixels at object boundaries or showing object-classes not part of the aforementioned 21 classes are labelled "void" and ignored during classification. The images are split into three sets: 276 training-, 256 test-, and 59 validation-images. The number and size of the object regions are shown in Figure 3.2.

This 21-class dataset is based on a 9-class dataset that was introduced earlier and contains the following object-class-labels: *building, grass, tree, cow, sky, aeroplane, face, car, bicycle.* It provides a training and test set each containing 120 images. We also use a subset of the 9-class dataset consisting of 6-classes (*cow, sheep, dog, cat, bird, grass*) in Chapter 4. This small dataset consists of 125 training and 50 test images. The detailed statistics for each object-class are also shown in Figure 3.2.

This dataset was one of the earliest widely available datasets that provided a consistent pixelwise groundtruth labelling suitable for the evaluation of the semantic segmentation task. The images contained in the MSRC dataset are not too challenging (*i.e.* good lighting, little clutter) and often display one object in the centre of the image. The groundtruth labelling "overdraws" foreground objects (*e.g.* cow, person, car, bicycle), meaning their annotation often overlaps the object boundaries and "spills" into the background (*e.g.* grass, road). This can be seen in Figure 3.5 on page 56 that shows the groundtruth masked cow region in the middle. The VOC dataset introduced in the next section tries to overcome some of these problems.

3.2 Pascal Visual Object-Class Dataset

The PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning organised object recognition challenges and simultaneously released image (*V*isual *O*bject-*C*lass) datasets with groundtruth labelling suitable for the evaluation of object detection and segmentation methods. There are currently two versions which include manually labelled groundtruth segmentation [Everingham *et al.* (2007, 2008)]. In this thesis we use the VOC2007 dataset which contains 20 object-classes *and* the background class. The provided images are divided into three sets: the training set consists of 209 images, the validation set of 213 images, and the test set of 210 images. The number of images containing a specific object-class as well as statistics about the size of the objects is presented in Figure 3.4.

See Figure 3.3 for example images of VOC2007 together with their groundtruth segmen-

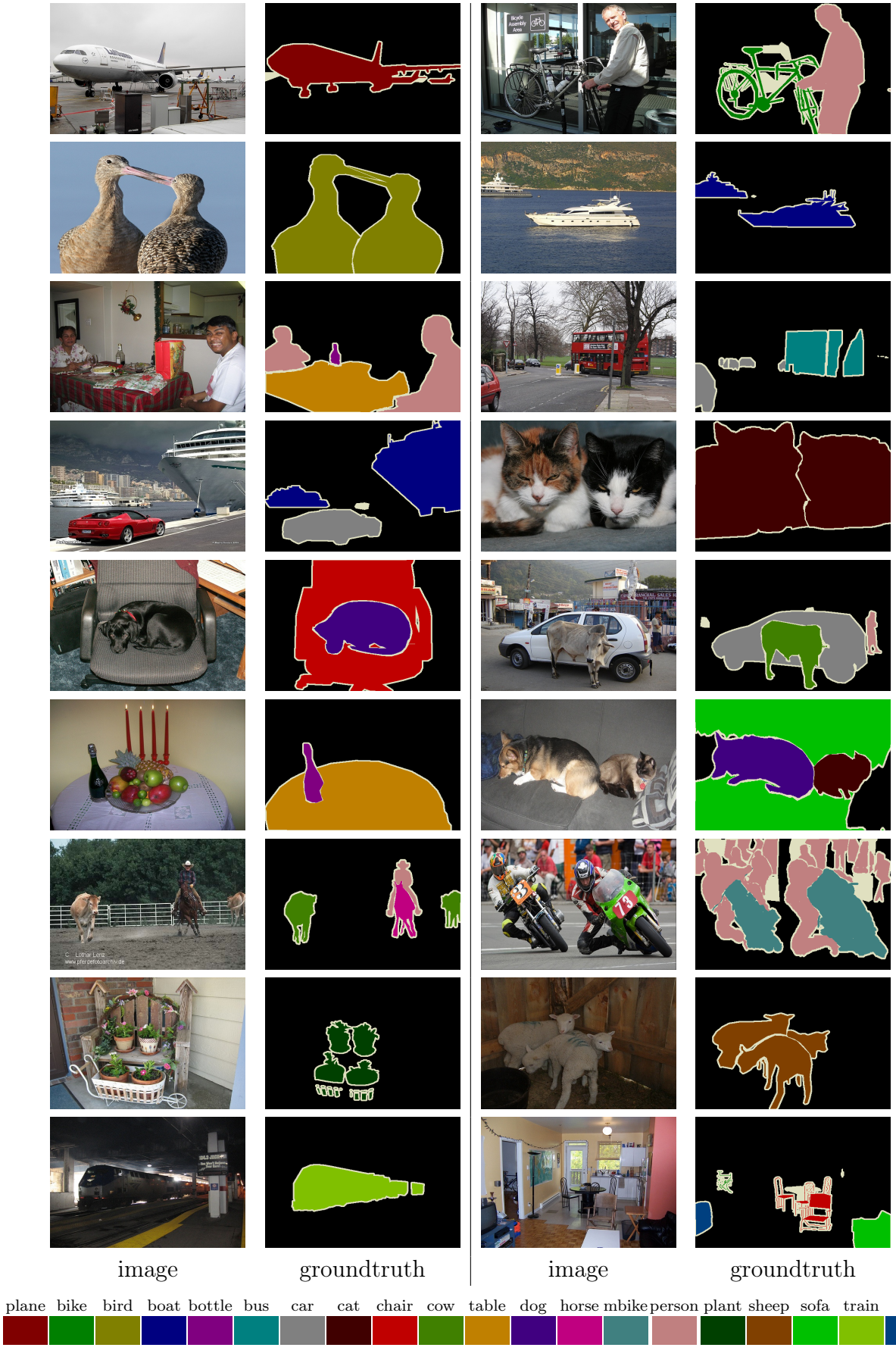


Figure 3.3: Pixelwise groundtruth segmentations for the VOC2007 dataset are shown together with the images. Each object-class is represented by a colour. Object boundaries ("void") are marked beige. For each of the classes *aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor* and *background* at least one example is given.

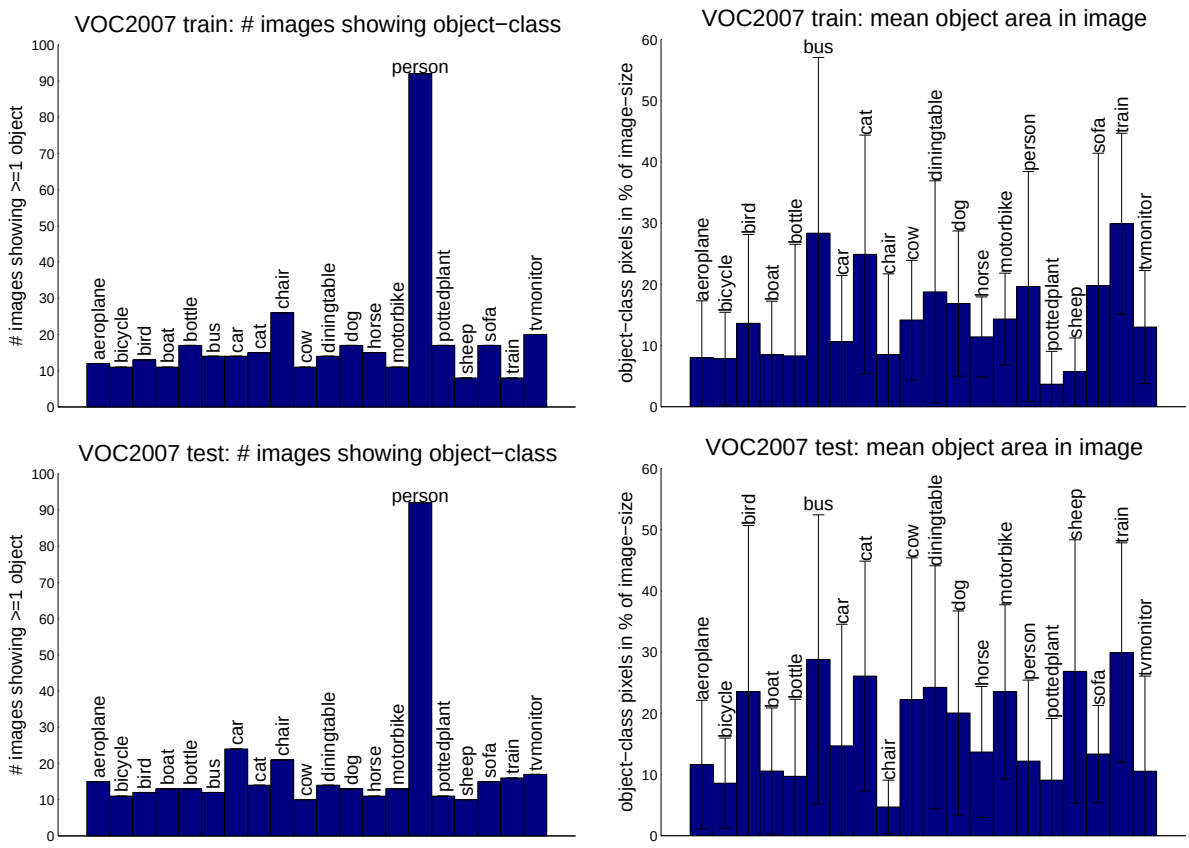
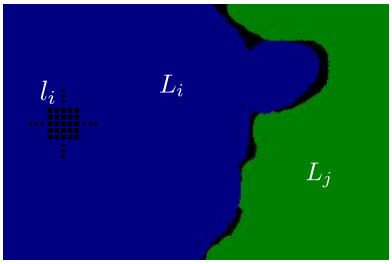


Figure 3.4: Object-class statistics for VOC2007. Shown are the number of images that contain each object-class and the average number of object-class pixels in percent of the number of image pixels (± one standard deviation indicated by the black error-bars around the blue mean).

tation. These images contain significantly more clutter than the ones in the MSRC datasets. Also, some of the object-classes are very difficult (*e.g.* potted-plant, dining-table). Furthermore, the addition of the background class makes the segmentation task much more challenging, as it summarises all objects not belonging to one of the 20 classes. This dramatically increases the range of appearances for the background class and potentially leads to more confusion with other classes.

For completeness it should be mentioned that the VOC datasets also provide groundtruth for the object detection and classification task. The groundtruth annotations contain bounding boxes around objects together with the class-labels and a rough orientation description (*e.g.* rear, front, facing left, facing right) as well as further information, for example if the object is truncated or fully visible.



Groundtruth

L_i =label for region;

l_i =label for pixel;

Region-Level Classification:

$$P_{reg} = \sum_{i=1}^{n} \frac{\delta(R_i, L_i)}{n}$$

n = number of groundtruth regions

Pixelwise Classification:

$$P_{pix} = \sum_{i=1}^{N} \frac{\delta(p_i, l_i)}{N}$$

$$P_{avg} = \sum_{c=1}^{C} \frac{P_{pix}(c)}{C}$$

N =total number of pixels

Figure 3.5: Visualisation of performance measures. The left column illustrates the two types of labelling that are available: (i) region labels and (ii) pixel labels. The middle column shows an example of a cow region from the MSRC dataset and the corresponding region-level performance measure P_{reg}. The right column specifies the performance measure P_{pix} when each pixel is evaluated independently ($P_{pix}(c)$ denotes the pixelwise classification performance for one object-class only) and P_{avg} for the average class performance. δ denotes the Kronecker delta. The middle images also shows how the groundtruth labelling for the MSRC dataset often overlaps the object boundaries and "spills" into the background.

3.3 Performance Measures

Here we introduce the performance measures that are commonly used on these datasets and we report the performance of our algorithms based on these measures in the following chapters.

There are three different kinds of performance measure that are used throughout this work to evaluate the semantic segmentation experiments. *Region-level classification* is based on the groundtruth regions and can therefore only be used for a preliminary evaluation. The other two measures take every pixel[1] into account (*pixelwise classification*) and only differ in how correct and incorrect classifications and object-class specific classifications are weighted. For the VOC2008 challenge another measure was introduced, the *intersection/union* measure also referred to as the *overlap score*, but it has not been widely used in other work yet.

[1]Pixels labelled as "void" are never taken into account. For the MSRC datasets these are all areas shown in black in the groundtruth images (*i.e.* pixels near the object boundaries, and for object-classes that are not included in the list). For the VOC datasets only a few pixel wide area around the objects and very small objects or ambiguous pixels are marked "void".

3.3.1 Region-Level Classification

Region-level classification was introduced in Winn *et al.* (2005) and poses a simpler classification problem. Instead of classifying a whole image (*e.g.* Csurka *et al.* (2004)), here sub-regions that contain only one object are classified. As this method relies on the groundtruth regions to define the object regions (see Figure 3.1 on page 51 and Figure 3.3 on page 54) it is not suitable to evaluate segmentation results, but is used for preliminary evaluations. Figure 3.5 gives an example of those groundtruth regions on the left. A label L_i is assigned to each such region, and during classification the agreement of the classification for a specific region R_i is compared to this label (δ denotes the Kronecker delta). The number of all correct assignments normalised by the total number of regions provides the performance measure we use, as shown in the middle column of Figure 3.5.

3.3.2 Pixelwise Classification

This measure is more suitable to evaluate our ultimate goal of classifying each pixel in the image correctly, *i.e.* assigning an object class-label to each pixel. Figure 3.5 visualises this idea in the right column as P_{pix}. The *pixelwise classification performance* is computed as the percentage of all correct assignments of class-labels to pixels p_i (where l_i denotes the groundtruth label for this pixel). Pixels labelled as "void" are not considered during this evaluation. In the case of the MSRC datasets this disregards all boundary pixels and pixels that do not belong to one of the 21 object classes. For the VOC datasets all pixels need to be assigned a label (aside from a small area around each object), either one of the 20 object class-labels, or the *background* label.

3.3.3 Average Class Performance

This measure relates to the pixelwise classification performance, but computes the average of the per object-class c pixelwise accuracies $P_{avg}(c)$ (*i.e.* the percentage of *e.g.* cow pixels classified as cow). This *average class performance* reduces the bias towards object-classes that cover more pixels in the test-data than others. In the case of the VOC datasets this is true for the background class for example.

3.3.4 Average Precision

Average precision refers to the average precision from recall 0 to 1. In order to retrieve precision recall curves the algorithm returns a ranked list based on its confidence that the object-class is present in the image or not. Recall is then defined as

$$r = \frac{\#\text{positive images returned}}{\#\text{total positive images}}$$

and the precision is defined as

$$p = \frac{\#\text{positive images returned}}{\#\text{images returned}}$$

where "images returned" corresponds to the position in the ranked list. These measures are computed for each position in the ranked list, *i.e.* each recall value r between 0 and 1, to define a precision-recall curve.

3.4 Summary

In this chapter we introduced two image datasets: the MSRC dataset and the VOC dataset. Both are widely used in the computer vision community. We focus on the MSRC datasets in Chapter 4 and Chapter 5, but also report results for the VOC2007 dataset in the latter.

The MSRC datasets have been used in various papers to evaluate object segmentation methods. Winn *et al.* (2005) introduce this dataset and evaluate the results of their method using the region-level classification described in Section 3.3.1. Shotton *et al.* (2006), Csurka and Perronnin (2008), Shotton *et al.* (2008), Verbeek and Triggs (2008) use the MSRC dataset to evaluate their semantic segmentation methods and we compare to their results in the following chapters.

Shotton *et al.* (2008), Csurka and Perronnin (2008) both report results on the challenging VOC2007 dataset. Everingham *et al.* (2007, 2008) discusses the methods submitted to the challenge and reports the newest submitted results. The performance for the image classification and object detection tasks are reported in average precision (AP). The top performing methods in Everingham *et al.* (2007, 2008) reach around 80%-90% AP for "easier" classes, *e.g.* airplane

and person. The more difficult classes are recognised with 30%-40% AP, *e.g.* bottle, dining table, or potted plant. The performance of the detection task is more difficult to be interpreted intuitively and we refer to the reports for details.

Chapter 4

Semantic Segmentation via Texton Models

This chapter explores semantic image segmentation approaches, as defined in Section 2.4, based on texton models, whose basics were introduced in Section 2.1.2. Here we focus on an approach known as *pixelwise classification*, that assigns an object-class-label to each individual pixel of an image. We investigate the two fundamental parts of such an image segmentation system: (i) the object-class models together with a suitable classifier; (ii) the context around each pixel that is used to describe that pixel. We suggest a method that assigns a class-label to each image pixel separately. The focus lies on the efficiency and simplicity of the proposed method. We compare the new proposed class models to comparable exemplar[1] based models with various distance measures in the nearest neighbour framework and show that our proposed model, together with the Kullback-Leibler divergence measure, corresponds to a generative model that returns the classification with the maximum a posteriori probability.

First we give an overview of the components involved in a texton based segmentation algorithm in Section 4.1 and introduce the role played by the two main components (i) object-class model and (ii) pixel context. This sets out the basic nearest neighbour based classifier and our proposed compact version of it, which is explained in full detail in Section 4.2. Furthermore, the crucial question of the image content around the pixel that is used during the classification step is defined and two solutions are given: sliding-window based context, and context extraction based on bottom-up segmentation. Both possibilities will be explored in the experimental

[1] An *exemplar* describes one single object-class region in a training image.

section 4.3 and in 4.5. In Section 4.4 we introduce extensions to this baseline that overcome some of the restrictions inherent to the sliding-window based context selection.

4.1 Texton Based Segmentation Algorithm

This section introduces the basic segmentation algorithm. In this chapter we use the textons as, introduced in Section 2.1.2 for the *image description,* and then focus on the specifics of *learning the classifier* and the *classification* step which are both based on texton histograms. Section 4.1.1 shortly summarises the learning of the classifier, followed by Section 4.1.2 which describes different methods to define the pixel context necessary for a pixelwise classification and greatly influences the performance of the classifier. A range of distance measures for the proposed nearest neighbour like framework is introduced in Section 4.1.3.

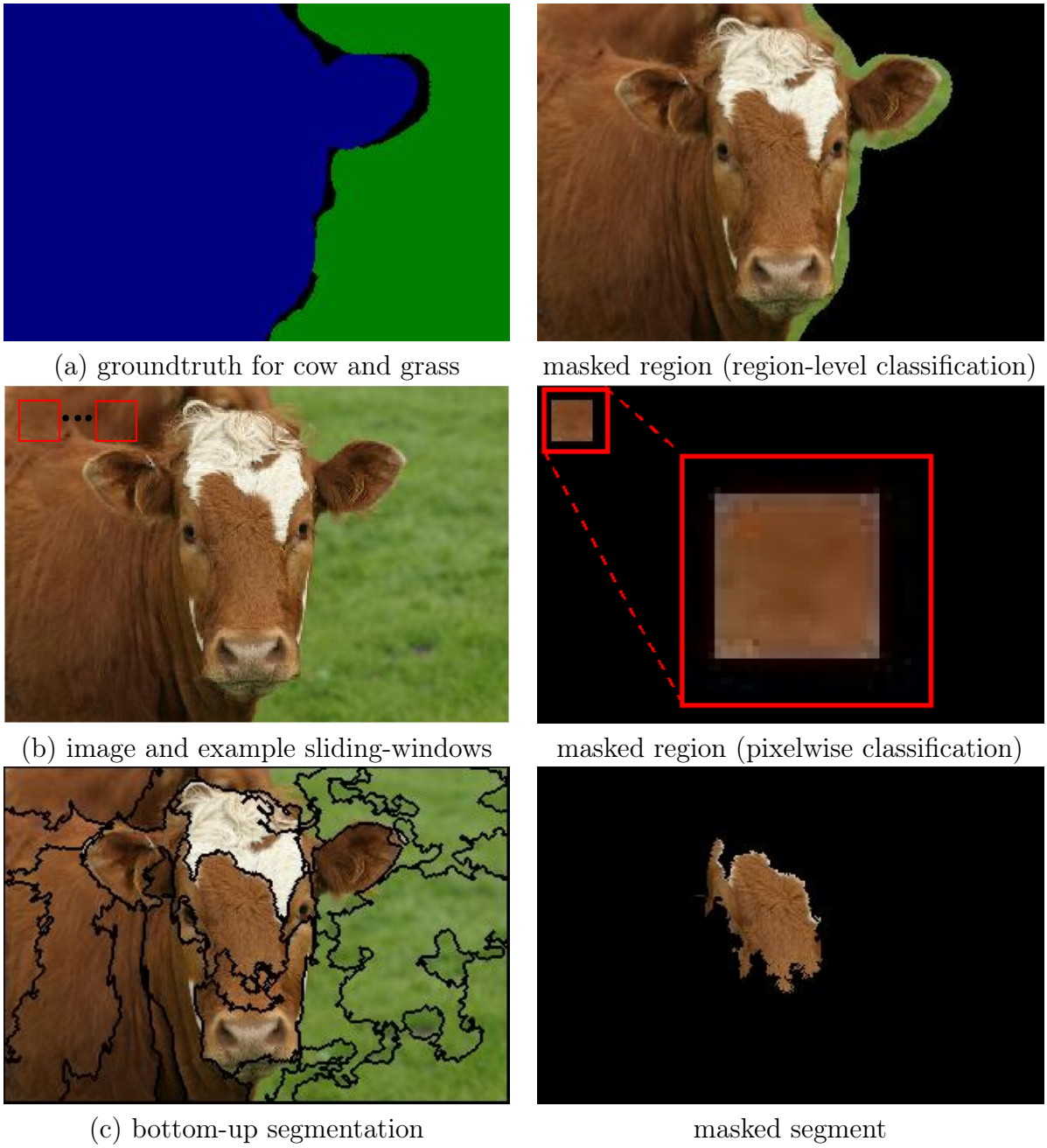
4.1.1 Texton Histogram Models

We model the training data with distributions over textons as described in Section 2.1.2. The texton features are computed densely, *i.e.* a texton ID is assigned to each pixel location.

We build on the previously introduced steps. After a vocabulary of V textons was learnt by k-means clustering on the features extracted from the training images, it is possible to associate each pixel position in the training images with the closest texton from the vocabulary. Now histograms over textons $\mathbf{q}^i$ for each of the training regions (exemplars) defined by the groundtruth labelling are computed in the learning step. The questions of how to model the object-classes and the nature of the pixel context, define the two main areas of focus for this chapter. The two models investigated are based on

1. storing those exemplar histograms $\mathbf{q}^i$ separately for standard nearest neighbour (k-NN) classification (see Section 2.1.2), or

2. combining all exemplar regions for an object-class to produce compact and yet discriminative models of object categories, called *single-histogram class models* (SHCMs).



(a) groundtruth for cow and grass

masked region (region-level classification)

(b) image and example sliding-windows

masked region (pixelwise classification)

(c) bottom-up segmentation

masked segment

Figure 4.1: Pixel-context. The three different types of context regions used in our experiments: (a) the manually segmented groundtruth on the left and the masked cow region using the blue region to compute **h**; (b) a sliding-window (red) defines the test region that is used to estimate **h** in the sliding-window based pixelwise classification; (c) bottom-up segmentation defines pixel-context.

4.1.2 Context Regions for Histogram based Classification

Before the actual classification is performed a *query-histogram* **h** needs to be computed. This histogram describes the context of the pixel p that is being classified. We explore three different approaches to defining this pixel-context. Figure 4.1 visualises the three spatial regions that are evaluated in this chapter and described in the following:

Groundtruth regions: The first method uses the groundtruth regions defined by the manual labelling. Figure 4.1(a) gives an example. The query-histogram is computed for all pixels of one object groundtruth region. This means that the object contained in that region is classified. This method is only used to compare our class model to previous work [Winn *et al.* (2005)] and to evaluate the proposed method together with k-NN on a simpler problem. We refer to it as *region-level classification*.

Sliding-window: The second method uses a sliding-window s_p centred around pixel p to compute the query-histogram $\mathbf{h}$. The sliding-window extends w pixels from the centre pixel into each direction, thus it is of size $(2 \cdot w + 1)^2$ pixels. At the image borders the sliding-window is cropped by the image. This method is used to assign a class-label to the centre pixel p of the window s_p, and therefore results in a *pixelwise classification*. Figure 4.1(b) gives an example for a sliding-window with $w = 12$.

Bottom-up segmentation: The third method uses an image segmentation S_i, where each segment defines the image region to be classified. This image region defines the context of each pixel that is part of the region, and so all pixels in this region will be assigned the same class-label. This method is described in detail in Section 4.5, where we will also illustrate how multiple bottom-up segmentations can be combined to improve the classification.

4.1.3 Classification

The query-histogram is compared to the model-histograms as in Section 2.1.2 and the closest exemplar histogram according to a distance measure D is found. This corresponds to the nearest neighbour framework, but the focus of this chapter lies on the computation and "merging" of these model-histograms $\mathbf{q}^c$. We now formalise the problem and then introduce our proposed model in the next section. Given the query-histogram $\mathbf{h}$, we assign a class-label $\hat{c}$ to the histogram $\mathbf{h}$ by minimising the following function.

$$\hat{c} = \arg\min_c \left(D\left(\mathbf{h} || \mathbf{q}^c\right)\right) \quad , \tag{4.1}$$

where D denotes the distance measure that is used to compute the distance between the query-histogram and the class-histograms. The following options for D are explored:

Kullback-Leibler divergence D_{KL}:

$$D_{KL}(\mathbf{h} \parallel \mathbf{q}) = \sum h_i \cdot \ln\left(\frac{h_i}{q_i}\right) \quad . \tag{4.2}$$

Euclidean Distance D_{L2}:

$$D_{L2}(\mathbf{h} \parallel \mathbf{q}) = \sum_i (h_i - q_i)^2 \quad . \tag{4.3}$$

χ^2 Distance D_{χ^2}:

$$D_{\chi^2}(\mathbf{h} \parallel \mathbf{q}) = \sum_i \frac{(h_i - q_i)^2}{h_i + q_i} \quad . \tag{4.4}$$

Here $\mathbf{h}$ denotes the query- or test-histogram and $\mathbf{q}$ denotes a model histogram (see Section 2.1.2) and the subscript on the corresponding non-bold letters specifies the histogram bin.

4.2 Single-Histogram Class Models

This section introduces and evaluates our compact class models (called *single-histogram class models* or *SHCMs*), where each object-class is modelled by a *single* histogram representing the distribution of textons for that class. This can significantly reduce the computational cost compared to standard *nearest neighbour methods* (k-NN) which require storing all training instances. It is crucial to have such a compact representation for each class in order to perform efficient classification of image regions. Chapter 5 exploits this classifier as a "weak"-classifier in the Random Forests framework. The single-histogram class models are evaluated using the MSRC dataset (see Section 3.1 and Figure 3.1 on page 51). Each image may contain instances from different object-classes and several instances from the same class. Groundtruth regions (connected pixels) in one image belonging to the same object-class form *one exemplar*.

The SHCMs, which will be presented in detail in the next sections, represent a generative bag of textons model for each object-class. We show that the nearest neighbour framework together with the Kullback-Leibler divergence as the underlying measure corresponds to a

maximum likelihood estimator for the class assignments.

4.2.1 Learning the Single-Histogram Class Models

This section describes details of how our models, specifically the SHCMs, are learnt. During training the histograms corresponding to different groundtruth regions (exemplars) belonging to the same class are combined together into a single, optimally estimated histogram. During testing the aforementioned query-histogram is compared to each of the C (number of classes) class histograms, as opposed to each of the (possibly many) exemplars, as in the case of k-NN classification. The use of single-histogram models clearly reduces the time complexity for classification. In Section 4.3 we show that they also improve performance.

The key question is how to compute such single-histogram models. Histograms are represented as V-vectors, with V being the vocabulary size. For a given class c, we want to compute the Maximum-Likelihood Estimate (MLE) for our class model, *i.e.* the histogram $\mathbf{q}$, given the texton distribution in the exemplars. The following section shows that this relates to using an average over the exemplar histograms together with Kullback-Leibler divergence (D_{KL}). Furthermore, it is shown how Euclidean distance (D_{L2}) fits into the same framework. Different distance measures such as Bhattacharyya, Alpha-Divergence [Hero *et al.* (2002)], or histogram intersection may be used as well. In general, however, the optimal SHCM does not correspond to the simple average of the exemplar histograms, and a more complex optimisation might be required. This for example, is the case for χ^2 distance (D_{χ^2}). As D_{χ^2} is a common distance measure for histogram distributions, we still report the results using the same SHCMs as for the KL divergence and Euclidean distance.

Maximum-likelihood estimate and Kullback-Leibler divergence. It is now shown that the maximum-likelihood estimate for the texton distribution in the single-histogram class model corresponds to minimising the Kullback-Leibler divergence of the SHCM to all the exemplar regions. Given the training data $\mathcal{T} = \{x_1, \ldots, x_n\}$ (all textons in the exemplar regions) for one object-class and their underlying distribution P and using the assumption that those exemplars are independent and identically distributed, it follows [Duda *et al.* (2001)]:

$$p(\mathcal{T}|\theta) \quad = \quad \prod_{i}^{n} p(x_i|\theta) \quad . \tag{4.5}$$

The MLE of θ (the parameters to estimate) is by definition, the value $\hat{\theta}$ which maximises $p(\mathcal{T}|\theta)$. Taking the logarithm we can define

$$l(\theta) := \ln p(\mathcal{T}|\theta) \quad = \quad \sum_{i=1}^{n} \ln p(x_i|\theta) \tag{4.6}$$

which we want to maximise with respect to θ. Let P_n be the empirical distribution which puts probability $1/n$ on each sample x_i, and Q_θ be the parametrised model (class model in the training case) that describes P_n. Then the expected value of $l(\theta)$ over distribution P_n is,

$$L(P_n, Q_\theta) := \mathcal{E}_{P_n}\{\ln p(x|\theta)\} = \frac{1}{n}\sum_{i} \ln p(x_i|\theta) = \frac{1}{n} l(\theta)$$

and the expectation over P of the empirical version $L(P_n, Q_\theta)$ is $L(P, Q_\theta)$ for any n, *i.e.* $\mathcal{E}_P\{L(P_n, Q_\theta)\} = L(P, Q_\theta)$ [Eguchi and Copas (2005)]. Maximising $l(\theta)$ is equivalent to maximising $L(P_n, Q_\theta)$, which again is equivalent to maximising $L(P, Q_\theta)$. Using the definition of KL-Divergence:

$$\begin{aligned} D_{KL}(P \parallel Q_\theta) \quad &= \quad \sum_{x} p(x) \ln \frac{p(x)}{q(x)} \\ &= \quad \sum_{x} p(x) \ln p(x) - \sum_{x} p(x) \ln q(x) \\ &= \quad L(P, P) - L(P, Q_\theta) \quad , \end{aligned}$$

it follows that maximising $L(P, Q_\theta)$, and therefore $l(\theta)$, is equivalent to minimising $D_{KL}\left(P \parallel Q_\theta\right)$. During training, this results in the maximum-likelihood estimate for our class model Q_θ. During classification, the class model (out of the learnt models: $Q_{\theta_1}, \ldots, Q_{\theta_C}$) which minimises $D_{KL}\left(P_t \parallel Q_{\theta_i}\right)$ or equivalently $L\left(P, Q_{\theta_i}\right)$ is found, thereby assigning a class-label. P_t denotes the visual-word distribution of the region to be classified and corresponds to the query-histogram $\mathbf{h}$ that was introduced previously.

Hence, given the texton counts from all the training images, we have the probability dis-

tribution of the textons P in one object-class. Given that, we want to determine the MLE for our class model (the histogram $\hat{\mathbf{q}} = (q_1, \ldots, q_V)$), we minimise $D_{KL}(P \parallel \mathbf{q})$ (see (4.2)).

P is the distribution of all V visual-words in the given class c. P is therefore the histogram of all visual-words of this class and thus:

$$P = \frac{\sum_j n^j \mathbf{p}^j}{\sum_j n^j} \quad , \tag{4.7}$$

with $\mathbf{p}^j$ being the normalised histograms of the exemplar j and n^j being the number of pixels in it. Thus, $\hat{\mathbf{q}} = P$ is our class model.

Model estimation as minimising the average distance to exemplar regions. Intuitively, it is equally justified to compute the class model such that it minimises the distance (Kullback-Leibler in this case) to all training histograms (exemplars). By doing so, it can be expected that the class model is as close as possible to the test histograms of the corresponding class. This constraint is formulated as follows:

$$E_{KL} := \sum_{j=1}^{N_c} n^j D_{KL}(\mathbf{p}^j \parallel \mathbf{q}) \quad \text{subject to} \quad \parallel \mathbf{q} \parallel_1 = 1,\ q_i \geq 0\ \forall i \quad , \tag{4.8}$$

where N_c is the number of exemplars for the object category c.

Standard manipulation (see Appendix 8.1.1) yields the global minimum of (4.8) as

$$\hat{\mathbf{q}} := \frac{\sum_j n^j \mathbf{p}^j}{\sum_j n^j} \quad , \tag{4.9}$$

which is equivalent to equation (4.7). The expression E_{KL} which is directly connected to the MLE of the texton distribution for one object-class, can either be use it in its pure form, *or* with $n^j := 1$, $j = 1 \ldots N_c$. The latter case corresponds to computing the normalised histogram of each exemplar and minimising the sum of the KL divergences from our class model to those exemplar histograms. As shown before, weighting each normalised exemplar histogram with the number of pixels n^j reduces to the MLE case.

During classification, as defined in (4.1), the class $\hat{c}$ that minimises the Kullback-Leibler divergence to the query-histogram $\mathbf{h}$ is chosen. This means that the class histogram $\mathbf{q}^{\hat{c}}$ is the one which explains the query $\mathbf{h}$ best and is the most likely class, due to the aforementioned

correspondence of KL and MLE.

Euclidean distance. Starting with (4.8), we may use different distance measures. Even though the MLE interpretation does not apply in the L2 case, it still has the same very intuitive interpretation, namely that the sum of the distance of the model histogram to all the exemplar histograms is minimised.

Once again, given the class c and its exemplar histograms $\mathbf{p}^j$ we seek the histogram $\hat{\mathbf{q}}$ which minimises the following cost:

$$E_{L2} := \sum_{j=1}^{N_c} n^j D_{L2}(\mathbf{p}^j, \mathbf{q}) \quad \text{subject to} \quad \| \mathbf{q} \|_1 = 1,\ q_i \geq 0\ \forall i \quad . \tag{4.10}$$

Standard manipulation leads to the same $\hat{\mathbf{q}}$ as obtained by minimising (4.8). For more details see Appendix 8.1.1 (8.6). The classification finds the class model $\mathbf{q}^{\hat{c}}$ that is closest to the query-histogram $\mathbf{h}$, based on the used distance measure, in this case D_{L2}. With the Euclidean metric it does *not* correspond to the MLE for the class given the model.

χ^2 distance. The χ^2 distance is defined in (4.4). It turns out that the same interpretation does not work here, the histogram $\hat{\mathbf{q}}$ which minimises the distance between the model histogram and all the exemplar histograms is *not* the average over the exemplar histograms.

4.2.2 Results: Region-Level Classification

We now adopt the region-level classification as in Winn *et al.* (2005) as a baseline comparison of k-NN (see Section 2.1.2) versus the proposed single-histogram class model and thus demonstrate the strength of our model, also in comparison to Winn *et al.* (2005). In region-level classification each of the aforementioned groundtruth regions (which are available for both training and test data) is classified as a whole, see 4.1(a) on page 62 for an illustration.

In this experiment the single-histogram class models are compared to the performance of the Gaussian models proposed in Winn *et al.* (2005). Following the evaluation methodology in Winn *et al.* (2005), we classify each input test region[2] as belonging to one of the classes in the dataset and measure the error with respect to groundtruth. Table 4.1 shows that the proposed

[2]The area belonging to a region and its groundtruth labels are known.

	1-NN (χ^2)	SHCM-KL	1-NN [Winn *et al.* (2005)]	1-NN T [Winn *et al.* (2005)]
9-class	92.34% (for V = 4000 and V =32000)	93.43% (V=64000)	93.4%	92.7%

Table 4.1: Region-level classification. Comparing the *region-level* classification performance obtained by our single-histogram class models using KL-Divergence (SHCM-KL) to that obtained from the conventional nearest neighbour classifier (1-NN) using the χ^2 distance. Shown are the *best* results if V is varied (V is shown in brackets). Results are comparable to previous published performances for this dataset [Winn *et al.* (2005)]. The authors use nearest neighbour (1-NN) on histograms over their full 1200 visual-word vocabulary and the discriminatively learnt reduced version (T).

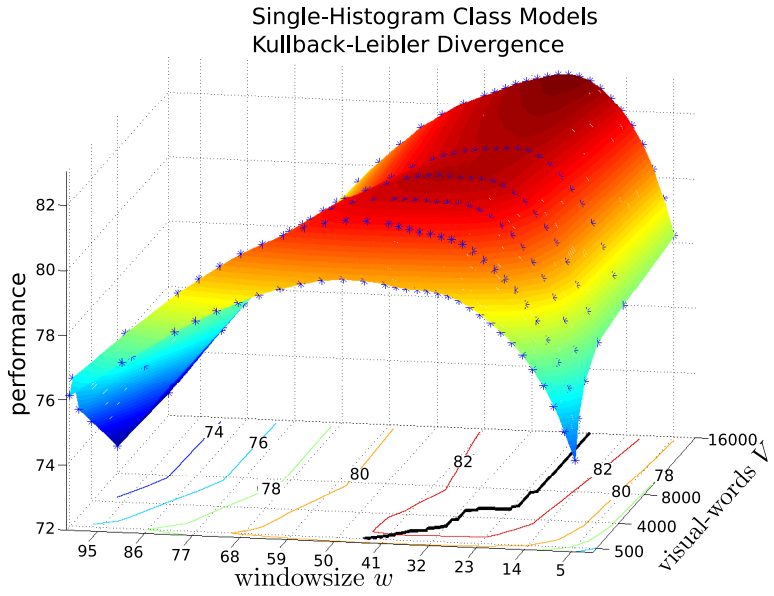
single-histogram class models perform comparably. For this comparison the exact training/test splits provided by the authors of Winn *et al.* (2005) were used. Each of the methods (k-NN using χ^2 on exemplars, and Kullback-Leibler divergence for single-histogram models) are optimised separately over the size of the vocabulary V, and the best result is reported. χ^2 is reported for k-NN as this gives superior results to L2 and it is a commonly used distance measure for region classification on exemplars [Varma and Zisserman (2003)]. In both cases the features are 5×5 patches and the visual vocabulary was constructed with k-means (10 iterations). In addition to the results given in the table we also experimented on the 6-class dataset (see Section 3.1) using $V = 8000$. The results are similar in that the single-class histogram reaches 85.5% and outperforms the 1-NN χ^2 classifier, which only reaches 79.5% on the region-level classification task.

4.3 Object Segmentation Results and Comparative Evaluation

In this section we investigate the segmentation performance of the model and algorithm introduced in the beginning of this chapter. We use the datasets introduced in Section 3.1, but especially the exploratory experiments where carried out on the smaller 6- and 9-class MSRC datasets.

All results, if not noted otherwise, indicate the pixelwise classification performance in percent. This is the proportion of correct classified pixels compared to the groundtruth, the measure was introduced in Section 3.3.

The visual vocabulary and class models are learnt from the training data only. As mentioned before, during testing a window of dimension $(2w + 1) \times (2w + 1)$ is slid across the image to



V ($w = 11$)	Acc. (%)	w ($V = 8000$)	Acc. (%)
500	79.1	5	80.3
1000	80.7	11	**82.4**
2000	81.7	15	**82.4**
4000	82.3	20	82.1
8000	82.4	26	81.1
16000	**83.0**	30	80

Figure 4.2: Parameter analysis on the 6-class dataset. Pixelwise classification performance as a function of the size w of the sliding window and the size V of the visual vocabulary. The features are 27-dimensional 3×3 CIE-LAB patches. The vocabulary is learnt by k-means clustering run for 500 iterations. KL divergence is used for histogram comparisons.

generate a query histogram $\mathbf{h}$ of visual-words assigned to the centre pixel of this window. In Section 4.3.1 performance over the system parameters using the 6-class and 9-class datasets together with single-histogram models is evaluated. Section 4.3.2 compares various versions of k-means clustering and how they affect the quality of the visual-word vocabulary. Finally, Section 4.3.3 compares the proposed single-histogram class model to the commonly used nearest neighbour classifier on the segmentation task.

4.3.1 The Effect of the Window and Vocabulary Size

This first set of experiments is designed to evaluate different values for the size of the sliding-window w and the vocabulary size V.

Figure 4.2 plots the pixelwise classification accuracy as a function of both the window size w and the vocabulary size V. The thick black line on the xy-plane denotes the sliding-window size dependent on the cluster size that result in highest performance. Two cross-sections of the accuracy function through the maximum are shown in the table. For the production of this graph k-means clustering until convergence with a maximum of 500 iterations was used. The features were 27 dimensional 3×3 CIE-LAB patches. The maximum performance is reached for $w = 11, 15$ and $V = 16,000$. Accuracy does not vary much over the range $V = 8,000$ –

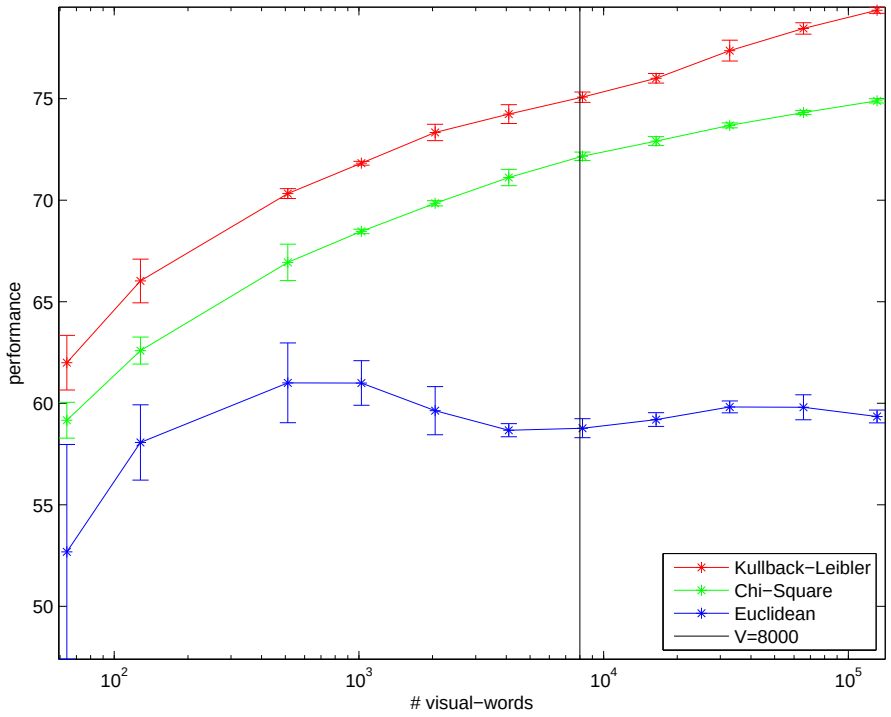


Figure 4.3: Accuracy for D_{KL}, D_{χ^2}, and D_{L2} depending on the size V of the visual-word vocabulary evaluated on the 9-class dataset using $w = 12$. Note that the x-axis shows the number of visual-words on a logarithmic scale. The error-bars show *two* times the standard deviation from the mean over five runs of k-means clustering with 10 iterations. The vertical line denotes $V = 8000$ – the number of visual-words we use in most of our experiments.

$128,000$, so from here on a vocabulary of size $V = 8,000$ and $w = 12$ are used to reduce the computational cost. The performance is found not to depend much on the size of the feature (*i.e.* size of colour patch), and we chose to use 5×5 colour patches from here on. Figure 4.3 shows a comparison of D_{KL}, D_{χ^2} and D_{L2} with varying size of the visual-word vocabulary. It shows that the performance increases with increasing vocabulary size for both D_{KL} and D_{χ^2} and levels out or decreases for D_{L2}. For even larger vocabularies a decrease in performance should be expected for D_{KL} and D_{χ^2} as well, but due to the significantly longer runtime we did not confirm this in experiments.

4.3.2 Influence of Methods to Build the Texton Vocabulary

In Table 4.2 the influence of different numbers of iterations in k-means clustering for the construction of the visual vocabulary is shown. Zero iterations denotes randomly sampled cluster centres from the feature space, which is how k-means is initialised in all cases. Interestingly the performance is only slightly affected by the number of iterations. In particular, there is only a small gain in increasing from 10 to 500 iterations. $V = 8000$ visual-words and 5×5 patches

		KM, 0 iters	KM, 1 iter	KM, 10 iters	KM, 500 iters
KL	6-class	$81.96 \pm 0.20\%$ (50)	$82.24 \pm 0.20\%$ (10)	$82.54 \pm 0.15\%$ (5)	**82.56** $\pm 0.13\%$ (5)
"	9-class	$74.72 \pm 0.22\%$ (10)	$74.92 \pm 0.17\%$ (10)	**75.07** $\pm 0.15\%$ (10)	–
χ^2	6-class	$76.59 \pm 0.17\%$ (50)	$76.66 \pm 0.16\%$ (10)	$76.60 \pm 0.09\%$ (5)	$76.45 \pm 0.03\%$ (5)
"	9-class	$69.64 \pm 0.20\%$ (10)	$70.91 \pm 0.10\%$ (10)	$72.00 \pm 0.07\%$ (10)	–
L2	6-class	$76.57 \pm 0.20\%$ (50)	$76.83 \pm 0.23\%$ (10)	$77.00 \pm 0.11\%$ (5)	$76.57 \pm 0.18\%$ (5)
"	9-class	$60.54 \pm 0.40\%$ (10)	$60.15 \pm 0.40\%$ (10)	$58.64 \pm 0.26\%$ (10)	–

Table 4.2: Variations of k-means (KM) clustering. The mean ($\pm$ one standard deviation) pixelwise classification accuracy computed over multiple runs of k-means; with the number of runs used in each case shown in brackets. Different numbers of iterations of k-means for constructing the visual vocabulary on the *6-class* and *9-class* sets are compared using single-histogram class models.

were used for the experiments in Table 4.2. The computational time increases drastically from 10 to 500 iterations, which is why this experiment was only performed for the 6-class dataset.

As a baseline classifier we compared the texton based single-histogram class models to pixelwise colour histograms of size 8000 (20 bins for each colour). This results in a pixelwise classification performance of 67.3% on the 9-class dataset and thus coincides with the findings of Verbeek and Triggs (2008), who report 67.1% for simple appearance based features.

4.3.3 Keeping all Exemplar Histograms *vs.* Single-Histogram Class Models

This section compares the performance of single-histogram class models to the nearest neighbour approach, where all training exemplars are stored and the query-histogram is classified according to the nearest neighbour exemplar. The experiments show that the performance of the single-histogram models is comparable to the classification accuracy retrieved by using k-nearest neighbour. However, the classification complexity is several-fold decreased, since only the closest class histogram has to be found instead of the closest histogram out of all training exemplars (about 200 training exemplars defined by the groundtruth regions in the training images, versus 6 or 9 class histograms).

Table 4.3 summarises the performance of k-nearest neighbour (k-NN) and the single-histogram models. It is apparent that the performance of the single-histograms model competes or even outperforms the k-NN method. Figure 4.4 on page 74 compares the single-histogram models (using KL divergence) to the k-NN version using all training exemplars and χ^2 dis-

	D_{KL}(6-class)	D_{L2}(6-class)	D_{χ^2}(6-class)	D_{KL}(9-class)	D_{L2}(9-class)	D_{χ^2}(9-class)
k-NN	82.1%	76.6%	78.7%	71.7%	64.9%	72.1%
single hist.	**82.4%**	77.0%	76.5%	**75.1%**	58.5%	72.0%

Table 4.3: k-NN *vs.* single-histograms. Comparing the *pixelwise* classification performance obtained by our single-histogram class models with that obtained from conventional nearest neighbour. In this case we used $V = 8000$ and 5×5 patches as features. K-means with 10 iterations was used to construct the visual vocabulary. For the 6-class set the best performing k out of $k = 1 \ldots 100$ and for the 9-class set *only* the performance for k-NN with $k = 1$ is reported, due to the runtime cost. Using single-histogram class models in conjunction with KL divergence produces the best results.

GT\Cl	grass	cow	sheep	bird	cat	dog
grass	95.61	2.0	1.2	1.2		0.1
cow	3.8	71.9	6.4	1.0	5.4	11.5
sheep	3.2	12.0	62.7	4.3	4.9	13.0
bird	5.5	27.1	24.0	27.7	10.4	5.4
cat		5.5	12.4	6.9	69.8	5.5
dog	1.1	24.7	2.3	6.5	18.2	47.2

(a) conf. mat. for 6-class set

GT\Cl	building	grass	tree	cow	sky	aeroplane	face	car	bicycle
building	56.2		5.0	3.3	2.0	13.0	1.4	11.4	7.7
grass	0.5	84.6	10.2	3.6		1.1			
tree	6.3	5.5	76.8	1.2	0.3	1.4		2.4	6.3
cow	2.3	2.6	2.6	83.6		0.3	4.6	3.2	0.8
sky	7.2		2.0	0.1	80.9	5.5		4.5	
aeroplane	17.0	0.8	4.8	3.3	0.2	54.6		15.0	4.4
face	4.2		0.5	18.4		0.9	69.0	3.7	3.4
car	6.5		0.8	3.8	0.7	2.4	1.9	70.4	13.5
bicycle	9.1	0.1	4.7	2.8		1.5	0.1	8.9	73.0

(b) conf. mat. for 9-class set

Table 4.4: Confusion matrices for the single class histogram method (see Table 4.3). (a) for the 6-class set; achieving an overall pixelwise classification accuracy of **82.4**%. (b) for the 9-class set; achieving a pixelwise classification accuracy of **75.1**%. KL divergence is used for both cases.

tance on a per image basis. It can be seen that the SHCMs perform similar to k-NN on many images, but also significantly better on a few images. Figure 4.5 shows two of the extreme outlier images. The experiments were performed using a vocabulary size of $V = 8000$ and 5×5 patches as features. The construction of the visual-word vocabulary was done by k-means with a maximum of 10 iterations (if not mentioned otherwise), a trade off between performance and computational time.

The optimal k in the k-NN was $k = 1$ for KL divergence, and $k = 3, 4$ for L2; for the 9-class set only $k = 1$ was used. Next we show full confusion matrices rather than just overall classification accuracies.

Table 4.4 shows the confusion matrices for selected experiments of Table 4.3. The matrices are row normalised (so that the percentages in each row sum to 100%). Only pixels belonging to one of the classes are considered (*i.e.* pixels labelled as void are disregarded). For the 6-class set, the grass class is recognised most reliably, followed by cows, cats and sheep. This provides

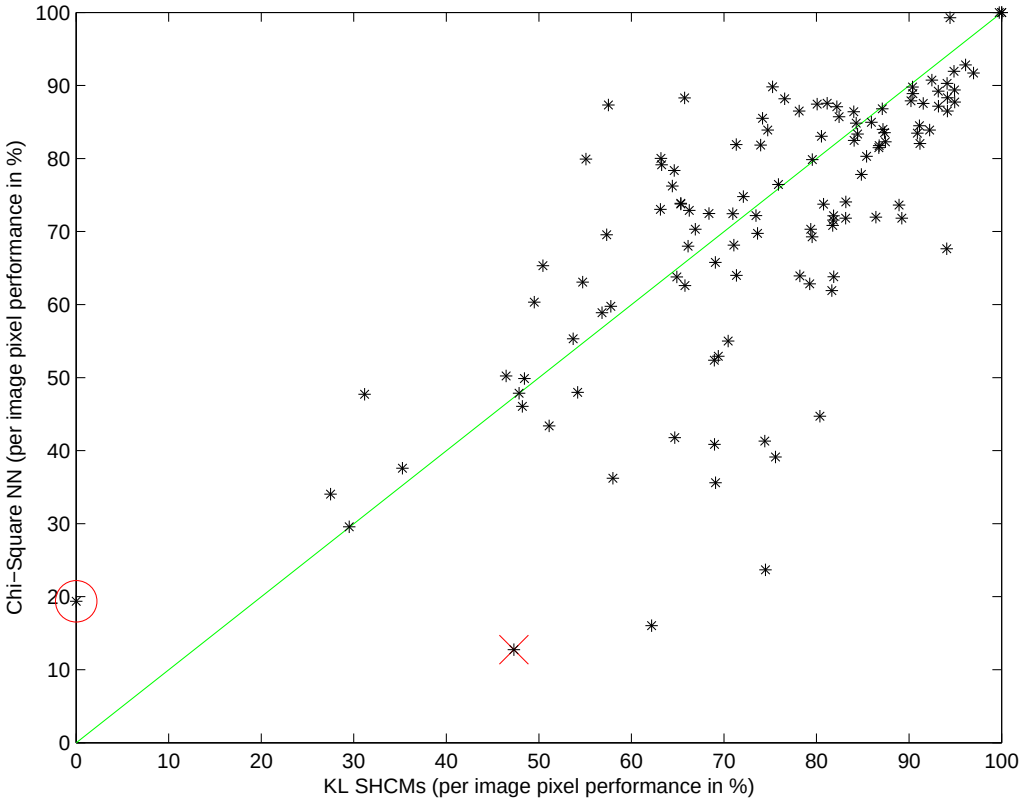


Figure 4.4: SHCMs *vs.* NN image scatter plot: This plot visualises the pixel performance for each test-image and compares SHCMs with KL divergence to the NN version (all training exemplars) with the χ^2 distance measure. The two outlier images marked with $\circ$ and $\times$ are shown in Figure 4.5. SHCMs perform better on 69 of the 120 test images and ties on 3 images.

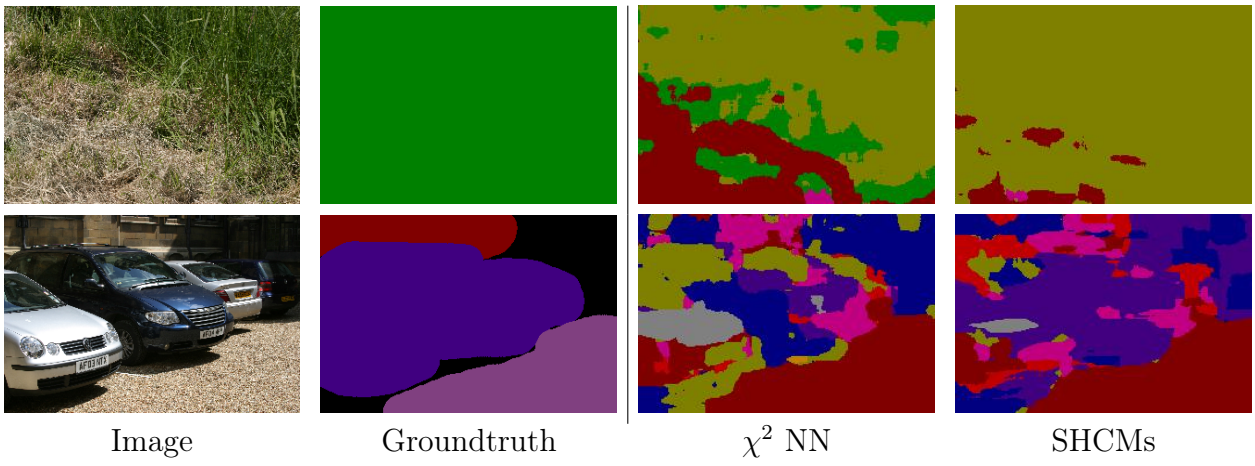


Figure 4.5: SHCMs *vs.* NN: Shown are two of the extreme outliers in Figure 4.4. The top row shows the image marked with $\circ$ the bottom row shows the one marked with $\times$. Given is the color coded classification for both χ^2 NN and SHCMs.

an idea of the relative difficulty of modelling each class. At this point one may think that our models work well only with texture-defined objects (grass, woolly sheep, *etc.*). However, we also include classification of man made (less texture-like) objects such as cars and bicycles in the 9-class dataset (also used in Winn *et al.* (2005)). Table 4.4(b) presents the confusion matrix

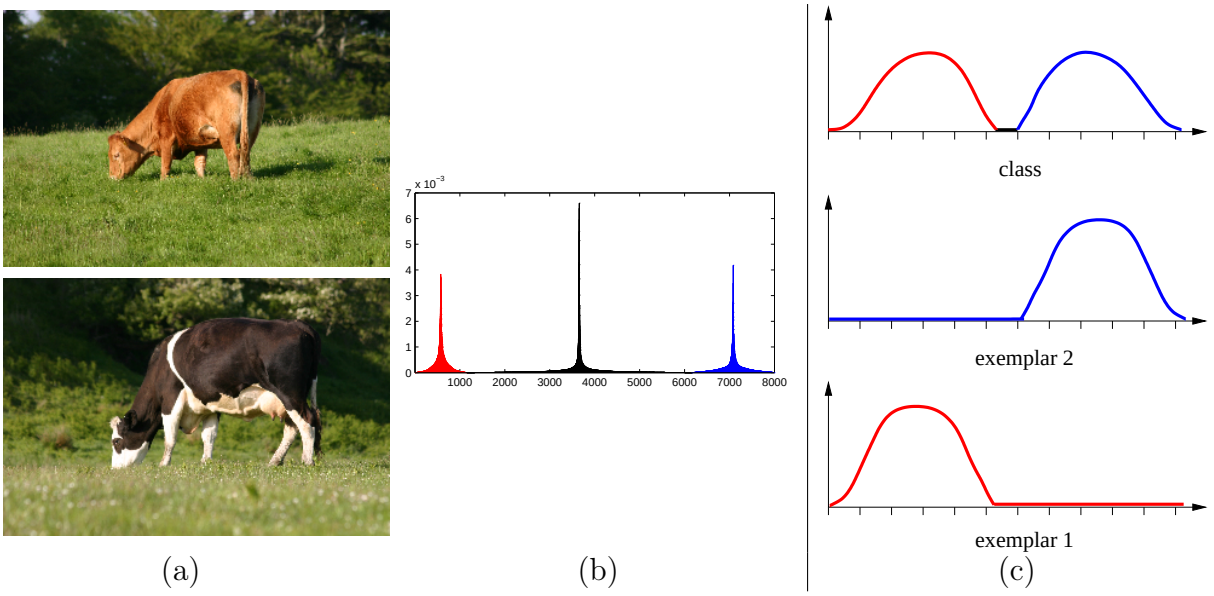


Figure 4.6: Multi-modality of single-histogram class models. Different instances of cows induce different proportions of visual-words. A unified "cow" histogram (b) will contain different "modes" for the different visual aspects and species of the class. In (b) the mode corresponding to the top cow in (a) is shown in red, and the mode corresponding to the bottom cow is shown in blue. The remaining visual-words of the *cow*-model are shown in black in the middle. Note that a simple sorting of the visual-words has been employed to bring out the different modes. (c) provides a schematic visualisation.

for this dataset. The performance is still well above 70%, thus confirming the modelling power of the proposed single-histogram models. Figure 4.7 shows some classification results.

KL Divergence *vs.* Euclidean Distance. The experiments of Table 4.3 on page 73 show that the KL divergence performs significantly better than the Euclidean distance measure for both k-NN and SHCMs, although the effect is more pronounced for the latter due to the multi-modality illustrated in Figure 4.6 and discussed in the following section. This confirms the better suitability of the KL divergence *for* single-histogram models. As mentioned before, for the χ^2 distance the minimisation of the objective function (4.1) is more difficult and so we use the average as defined in (4.9).

4.3.4 Discussion

As the experiments demonstrate (see Table 4.3 on page 73 and Figure 4.3 on page 71), KL divergence is superior to both $L2$ and χ^2 distance when the single-histogram models are used. This observation can be explained by the fact that the KL divergence deals properly with multi-modal distributions. Different instances of a class may induce a highly multi-modal class

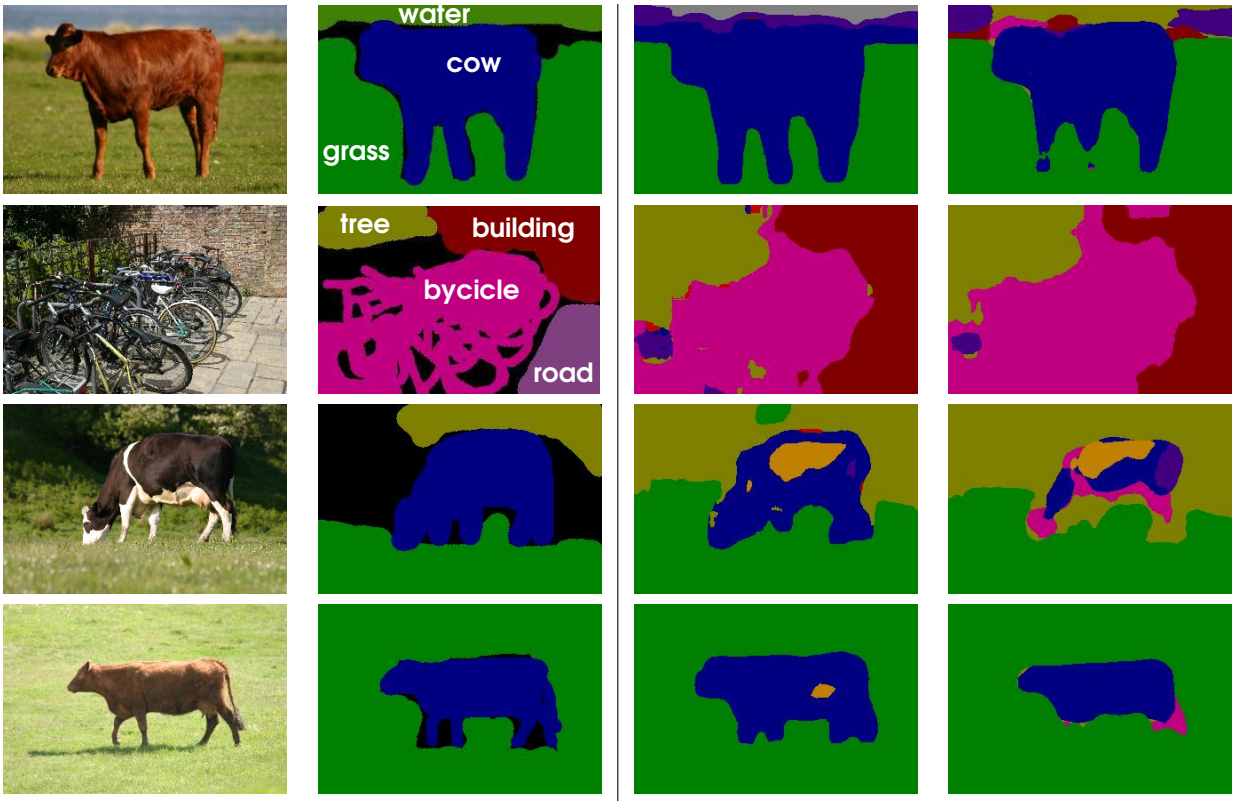


Figure 4.7: Class segmentation results. (a) Original photographs. (b) Groundtruth class-labels. (c) Output class maps obtained with KL divergence. (d) Output class maps obtained with L2 distance. In most cases L2 gives less accurate segmentation. In both cases our single-histogram class models were used, together with 5×5 patch features, $V = 8000$ and k-means clustering.

histogram. Consider the three schematic histograms shown in Figure 4.6(c). If $L2$ (or χ^2) distances are used then each exemplar histogram will have a large distance from the class histogram (due to bins q_i of the mode in the class histogram which are not present in each of the exemplars). However, the KL divergence ignores all the null bins of the query histograms $\mathbf{h}$ (as these evaluate to zero in $\sum_i h_i \log \frac{h_i}{q_i}$), thus making it a better suited distance (although it still penalises histograms which are not identical to the class histogram, *e.g.* an exemplar histogram having one mode of the class histogram does *not* have distance zero to the class histogram, as the histograms are *normalised* visual-word distributions). Figure 4.6(b) provides an example of such a multi-modal class histogram (here the cow model), and two exemplar regions inducing modes in the class model.

Until now we were only minimising the intra-class distance of the model histograms, based on MLE for KL. Generally, it would be desirable to also maximise the inter-class distance when building the single-histograms, which would result in a more discriminative classifier.

The comparison to k-NN, however, seems to justify the simplification of only minimising the intra-class distance.

Finally, Figure 4.7 shows some results of images into their constituent object-class regions. Note that the (visual) accuracy of the L2 classification results is inferior to that obtained with KL divergence.

Informal experiments revealed that the classification method is quite robust with respect to the vocabulary. Choosing *not* the nearest visual-word but the second closest, or using approximate nearest neighbour methods gave virtually the same results. The latter provides a significant speedup with comparable performance.

4.4 Modelling Test Regions as a Mixture of Classes

In this section we generalise the assignment of only one object-class to each image region (including sliding windows, *i.e.* pixels) in a natural way. Here the classification of an image region is modelled as a mixture of classes. This seems reasonable since a given region (sliding-window) often contains more than one object-class. Section 4.2.1 focused on the training phase by introducing a compact model for each class; in contrast this section focuses on the testing phase using the models we introduced before. The single-class histograms are the basis of this section and they facilitate the proposed model due to their simplicity. A similar approach would be challenging with more sophisticated models, *e.g.* the Gaussian distribution over histograms proposed in Winn *et al.* (2005). There is a significant amount of literature about mixture classification in the field of remote sensing and geographical information systems, where it is often referred to as fuzzy classification. Kent and Mardia (1988) is one of the earlier papers and introduces two models based on multivariate Gaussian random fields. Wang (1990) also gives a good overview. In remote sensing applications usually a good model of the mixing effects does not exist. In our case however, given our visual-word class model, we can explicitly model the mixed signal inside a sliding-window and thus solve for the mixing coefficients.

4.4.1 Two-Class Mixture Model

Given a region in an image and its corresponding histogram of visual-words, instead of finding only *one* closest class histogram, we find a mixture of a pair of model histograms together with

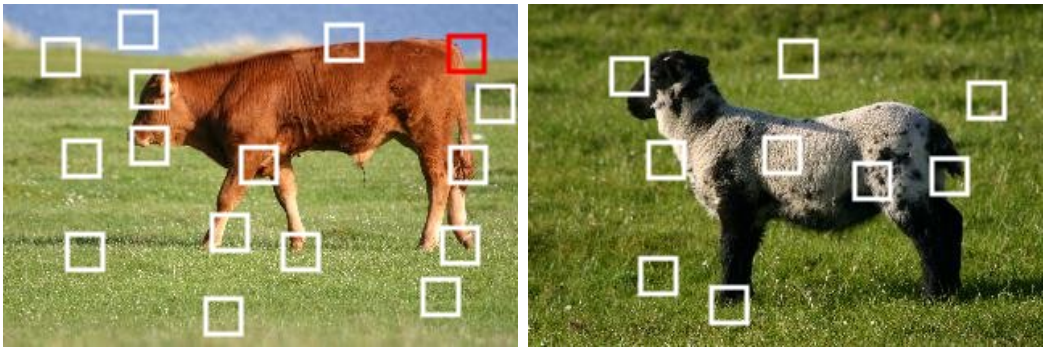


Figure 4.8: Sliding-windows at different locations. Assuming the presence of two object-classes within *small* windows is reasonable. It is possible that a windows can contain more than two object-classes, as shown in the red rectangle on the left, however, this will not occur very often.

the corresponding mixing coefficient.

During classification, each pixel will be explained as belonging to two different classes with different coefficients. In this way we hope to better model transitions between different object-classes and, capture thin structures such as animal legs. Furthermore, as shown in Figure 4.8 having one or two object-classes within small sliding-windows is a common event, while finding more than two classes in such small areas is much rarer (*e.g.* red rectangle).

Let $\mathbf{h}$ be the test-histogram and $\mathbf{q}^i$ and $\mathbf{q}^j$ two single-histograms class models. Then $\mathbf{h}$ can be interpreted as a mixture of the two model histograms as follows

$$\mathbf{h} \overset{D}{=} \alpha\mathbf{q}^i + (1-\alpha)\cdot\mathbf{q}^j \quad \text{with} \quad i \neq j \quad , \tag{4.11}$$

where $D \in \{D_{KL}, D_{L2}\}$ denotes the distance measure that is used and $\alpha \in [0,1]$ refers to the mixing coefficient. Thus, a new minimisation problem needs to be solved for all $\{(i,j)\colon i \neq j, 1 \leq i < j \leq C\}$ (C denotes the total number of object-classes). The result of the minimisation is a pair of histograms $(\hat{i},\hat{j})$ and the value of the mixture parameter $\hat{\alpha}$ which minimise (4.11). Both D_{KL} and D_{L2} are presented. Assuming a tuple (i,j) and $\mathbf{h}$ are given $\hat{\alpha}$ can be determined as follows. For details see Appendix 8.1.2.

KL divergence. The aim here is to minimise the objective function $F_{KL}\colon = \sum_{i=1}^{V} h_i \log\left(\frac{h_i}{\alpha a_i + (1-\alpha)b_i}\right)$ subject to $0 \leq \hat{\alpha} \leq 1$. Since F_{KL} is convex a Gauss-Newton optimisation algorithm is employed to determine $\hat{\alpha} = \arg\min_\alpha F_{KL}$.

Euclidean distance. The convex energy to be minimised is $F_{L2} := \sum_i (h_i - (\alpha a_i + (1-\alpha) b_i))^2$. Algebraic manipulation leads to the global minimum $\hat{\alpha} = \frac{\sum_i (a_i - b_i)\cdot(h_i - b_i)}{\sum_i (a_i - b_i)^2}$ with $0 \leq \hat{\alpha} \leq 1$ to be fulfilled in addition.

Efficiency. Rather than searching for each pair (i, j) and then compute $\hat{\alpha}$, a greedy approach can be used. First (i) the single-histogram closest to the test histogram is found, then (ii) the second closest histogram is chosen and finally (iii) the optimal mixing coefficient is determined as described above. The results are comparable to the global minimum. Clearly, there are further approximations which could be analysed.

4.4.2 Multi-Class Mixture Model

Instead of modelling the histogram as a mixture of only two classes the mixture can consist of all C classes, called the n-class case. The *n-class* case denotes the method were

$$\mathbf{h} \stackrel{D}{=} \sum \alpha_c \mathbf{q}^c = \underbrace{\left[\mathbf{q}^1, \ldots, \mathbf{q}^C\right]}_{\mathtt{Q}} \boldsymbol{\alpha} \tag{4.12}$$

with $c = 1 \ldots C$ denoting all classes is used as mixture model, with $\mathtt{Q}$ being the matrix consisting of $\mathbf{q}^c$ as columns and $\boldsymbol{\alpha}$ denotes the vector consisting of all α_c. Thus, the mixture of histograms $\hat{\boldsymbol{\alpha}}$ which minimises the distance $D \in \{D_{L2}, D_{KL}\}$ to the query-histogram $\mathbf{h}$ should be determined.

Euclidean distance. This problem can be efficiently solved for D_{L2} using the pseudo inverse of a matrix. This leads to the minimisation problem given in (4.13) which can be solved using the pseudo-inverse of $\mathtt{Q}$ if it is invertible (see (4.14)). This is the case if $\mathtt{Q}$ has full column rank, which is usually the case, since the class histograms $\mathbf{q}_c$ are most likely to be linearly independent. It should be pointed out that this problem formulation does not impose the constraint of $\alpha_c \geq 0$ as was done for the two-class formulation. We get the following nice compact solution.

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{V} (\mathbf{h} - \mathtt{Q}\boldsymbol{\alpha}) \right)^2 \tag{4.13}$$

$$\hat{\boldsymbol{\alpha}} = \left(\mathtt{Q}^T \mathtt{Q}\right)^{-1} \mathtt{Q}^T \mathbf{h} \tag{4.14}$$

Kullback-Leibler divergence. If KL-Divergence is used instead of Euclidean distance the multi class mixture model is related to probabilistic Latent Semantic Analysis (pLSA) [Hofmann (2001)] used in statistical text analysis. The difference in the method proposed here is that the topic vectors are restricted to only be formed from histograms of a single class, but each entity (pixel) is modelled as a mixture of topics (here classes).

Thus, once the single-histogram models are learnt, pLSA can be used to compute the mixture. Using the common notation for pLSA we have $P(w|d) = P(w|z)P(z|d)$ (see Figure 2.9). In our case w are the visual-words, d are the images and z are the classes, thus this translates to $P(w|d) = \mathbf{h}$, $P(w|z) = \mathtt{Q}$ and $P(z|d) = \boldsymbol{\alpha}$ which exactly gives (4.12) with $D = D_{KL}$. This can be solved using the "folding-in" EM-algorithm for pLSA.

The multi-class mixture model on its own does not significantly improve performance (see Table 4.5 on page 82 for the D_{L2} case), but it can be used if consecutive steps require a confidence value for each class instead of just the maximum likelihood estimate.

4.4.3 Evaluation of the Two-Class Mixture Model

In this section we assess the advantages of using the mixture model in terms of both classification and segmentation performance. Most of the experiments used D_{L2} since its current implementation is much faster than D_{KL} and it is sufficient in order to illustrate the properties of this idea.

Modelling ambiguity of classification. Given an input image, for each pixel estimate the most likely mixture of two object-classes and also their mixing coefficient $\hat{\alpha}$. The mixing coefficients are the ideal indicator of "ambiguity" of a pixel. In fact the difference $\overline{\alpha} = |\hat{\alpha} - 0.5|$ measures the amount of "mixing" of a pixel. As shown in Figure 4.9(d,i) the largest amount of mixing effects occur at the transition between different object-classes (shown in *black*).

Dealing with the "background" class. A problem with the technique in Winn *et al.* (2005) is that no "background" class is defined. Therefore, during testing the classification is forced to produce an answer within the set of known object-classes. There is no "unknown" class.

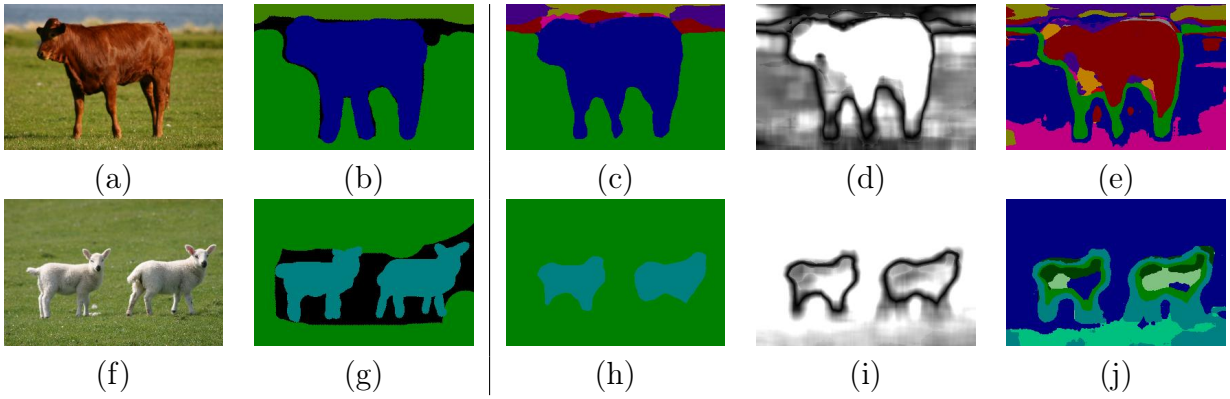


Figure 4.9: Two-class mixture model. (a,f) original test images, (b,g) groundtruth images, (c,h) colours at each pixel correspond to class of histogram $\mathbf{q}^i$ in equation (4.11) if $\hat{\alpha} > 0.5$, $\mathbf{q}^j$ otherwise. (d,i) map of $\bar{\alpha}$ with black for $\bar{\alpha} = 0$ and white for $\bar{\alpha} = 0.5$, (e,j) as for (c,h) with $\mathbf{q}^i$ if $\hat{\alpha} < 0.5$, $\mathbf{q}^j$ otherwise. $V = 8000$, $w = 12$, 9-class set, and L2 distance were used and $\overline{\alpha} = |\hat{\alpha} - 0.5|$.

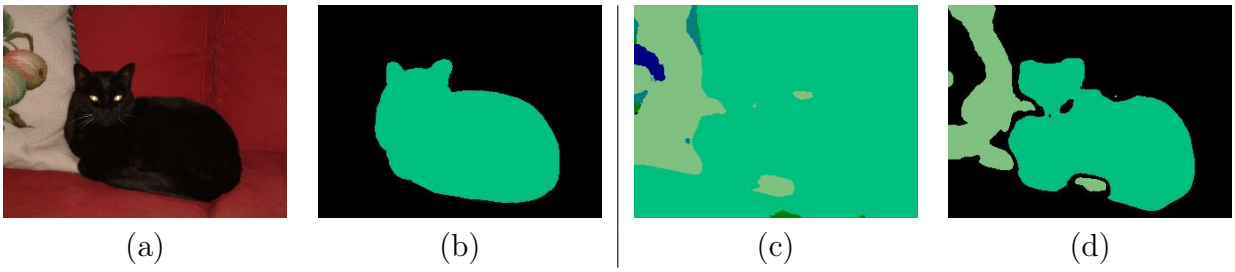


Figure 4.10: Background class. Another example illustrating the advantage of capturing classification ambiguity. (a) original image, (b) groundtruth labelling. Note that the background sofa is labelled as cat (c) and rejected in (d) with $\bar{\alpha} < 0.3$. $V = 8000$, $w = 12$, 6-class set, L2 distance and 3×3 patches were used.

Figure 4.10(c) illustrates this problem. The red sofa (the class "sofa" does not belong to the known set) is incorrectly labelled as "cat", due to the mixture model however this region has a high mixing factor and can be rejected using a threshold α_t on $\bar{\alpha}$.

The conventional solution to this problem is to learn a model for the background class, from training examples. However, the problem is finding the right example images. Also, every time the set of known object-classes changes, the example images of the background class change and thus the background model needs to be re-trained. This problem can be avoided if the mixture model is used.

Performance as a function of rejection. Table 4.11 illustrates the pixelwise classification performance as a function of α_t, for KL divergence. As α_t increases the number of retained pixels decreases, but their confidence increases and so does the classification performance.

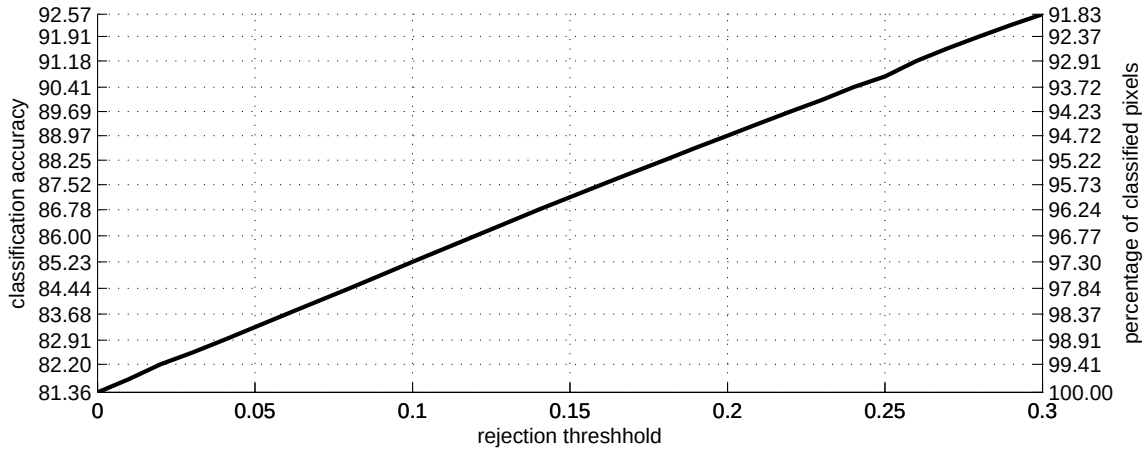


Figure 4.11: Rejecting pixels. Classification performance as a function of the rejection threshold α_t. As α_t increases the number of retained pixels decreases, but their confidence increases and so does the classification performance. Here KL divergence was used. Note: The left y-axis (classification accuracy) is linear, however the labelling is not; it corresponds to the experimental values for certain α_t. The right y-axis is *not* linear. It is interesting to note that the classification accuracy almost linearly depends on α_t.

	no voting	voting
D_{L2} 2-class ($w = 12$)	77.5	77.9 (77.8 Gauss)
D_{L2} 2-class ($w = 5$)	76.7	77.1 (77 Gauss)
D_{L2} n-class ($w = 12$)	78	78.4 (78.3 Gauss)
D_{KL} 2-class ($w = 12$)	82.5	81.7 (81.6 Gauss)

Table 4.5: Voting. Overview of different voting methods using the SHCMs. Pixelwise classification accuracy is given in percent.

4.4.4 Exploiting the Mixture Model

We now suggest three applications that are based on the proposed mixture model and illustrate how it can address problems, that would otherwise need to be solved with ad-hoc methods.

Voting. *Voting* is a method which does not simply assign the mixture of histograms determined from the sliding-window to the centre pixel of this sliding-window, but instead adds the value(s) of the mixing factor(s) $\boldsymbol{\alpha}$ to *all* pixels contained in the sliding-window. Each pixel's mixture is therefore the sum of the mixtures from all sliding-windows which contain it.

The experiments where carried out on the 6-class dataset using 8000 cluster centres determined with a maximum of 500 k-means and 3×3 patch features. Table 4.5 shows results for D_{L2}, D_{KL} and two different sliding-window sizes ($w = 5, 12$). The first column (*no voting*) gives an overview of the classification accuracy if the class with the highest mixing factor is chosen as the final classification for each pixel. The value given in brackets refers to the pixel-

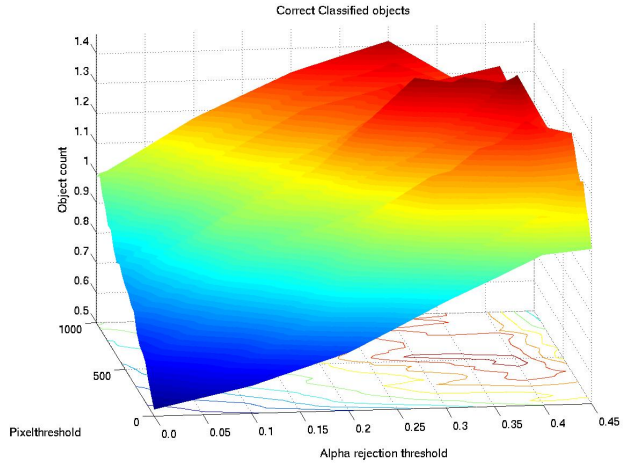


Figure 4.12: Object detection ratio as a function of the rejection threshold. Shown is the ratio $\frac{correct}{falseneg.+falsepos.}$ with respect to α_t. $V = 8000$, $w = 12$, 9-class set, and L2 distance were used.

wise classification accuracy in percent for voting weighted by a Gaussian that is centred on the centre-pixel with a standard deviation of half the sliding-window size ($\sigma = \frac{1}{2} \cdot w$). The results indicate that *voting* can improve the performance slightly.

Scene classification. The proposed mixture model is applied to the problem of multiple object detection. The goal is to detect which of the objects in our dataset appear in a given input image. Using the pixelwise classification technique described earlier a list of the classes present in an image can be compiled and compared with groundtruth. A class c is reported as occurring if $p > p_t$, *i.e.* if at least p_t pixels in a connected component are classified to belong to the class c. The results are shown in Figure 4.12 as a function of the rejection threshold. The results show how the ratio between correct and incorrect detections (*false negatives*: not detected classes; *false positives:* wrongly detected classes) increases with the rejection threshold α_t and the pixel threshold p_t. It is clear that the mixture model can improve the performance over the use of a simple pixel threshold p_t. However, as the underlying classifier is tuned towards object segmentation it cannot be expected to compete with object detectors used in the VOC challenges [Everingham *et al.* (2006, 2007, 2008)].

Foreground- & background-segmentation. Figure 4.13 gives two examples where the mixture model is used to define a *background-* and *foreground*-map to initialise graph-cuts (see Section 2.2.4). Note that the object-class that we wish to segment as foreground must be specified by classname, here dog and cow. This is an illustration of nice segmentation results

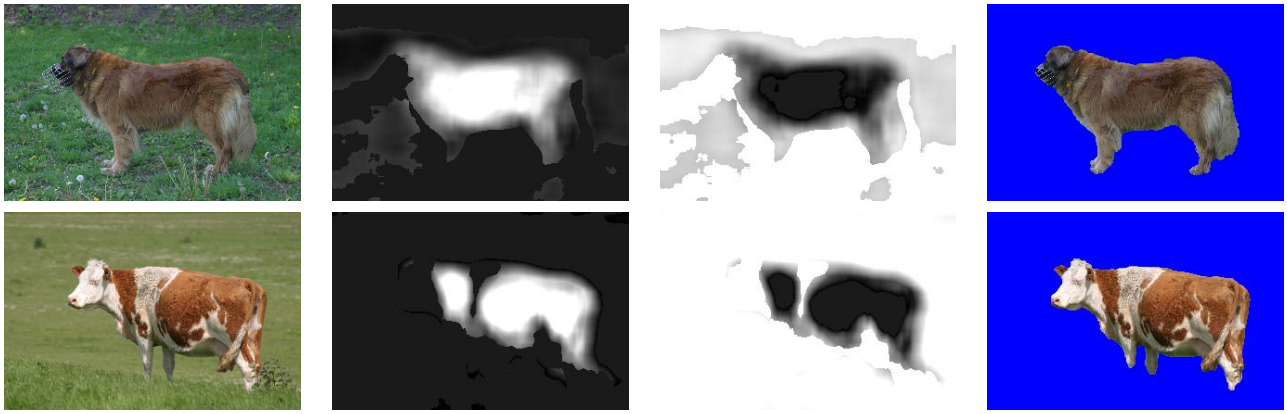


Figure 4.13: Foreground- & background-segmentation examples. The mixture model can be used to extract a foreground- and background-map which can then be used together with Graph-cuts to give an object segmentation.

that does *not* require any user interaction, unlike Rother *et al.* (2004), for example.

4.4.5 Discussion

The single-histogram class models introduced in Section 4.2 have been employed in this section to explain input test regions as a mixture of two or more different object-classes. The mixture model has enabled us to model classification ambiguity explicitly. In turn, this has allowed us to reject ambiguous pixels and increase the reliability of our recognition algorithm. Thus it enables the creation of foreground- and background-maps to define an initialisation for Graph-cuts based segmentation. Furthermore, it can avoid the explicit modelling of a background class, since the background can be modelled indirectly with the mixture model. Other applications could involve "modelling" a full "posterior" over object-classes with the mixture model approach, if no probabilistic model is used; for example, in the case of the D_{L2} or D_{χ^2}. In the case of D_{KL} the distance directly corresponds to the likelihood of the test-histogram $\mathbf{h}$ being drawn from the multinomial class model $\mathbf{q}$ and thus induces a class posterior, which is required for the experiments in the following section. Figure 4.13 provides two nice examples for the automatic generation of tri-maps, that can be used for foreground background segmentation.

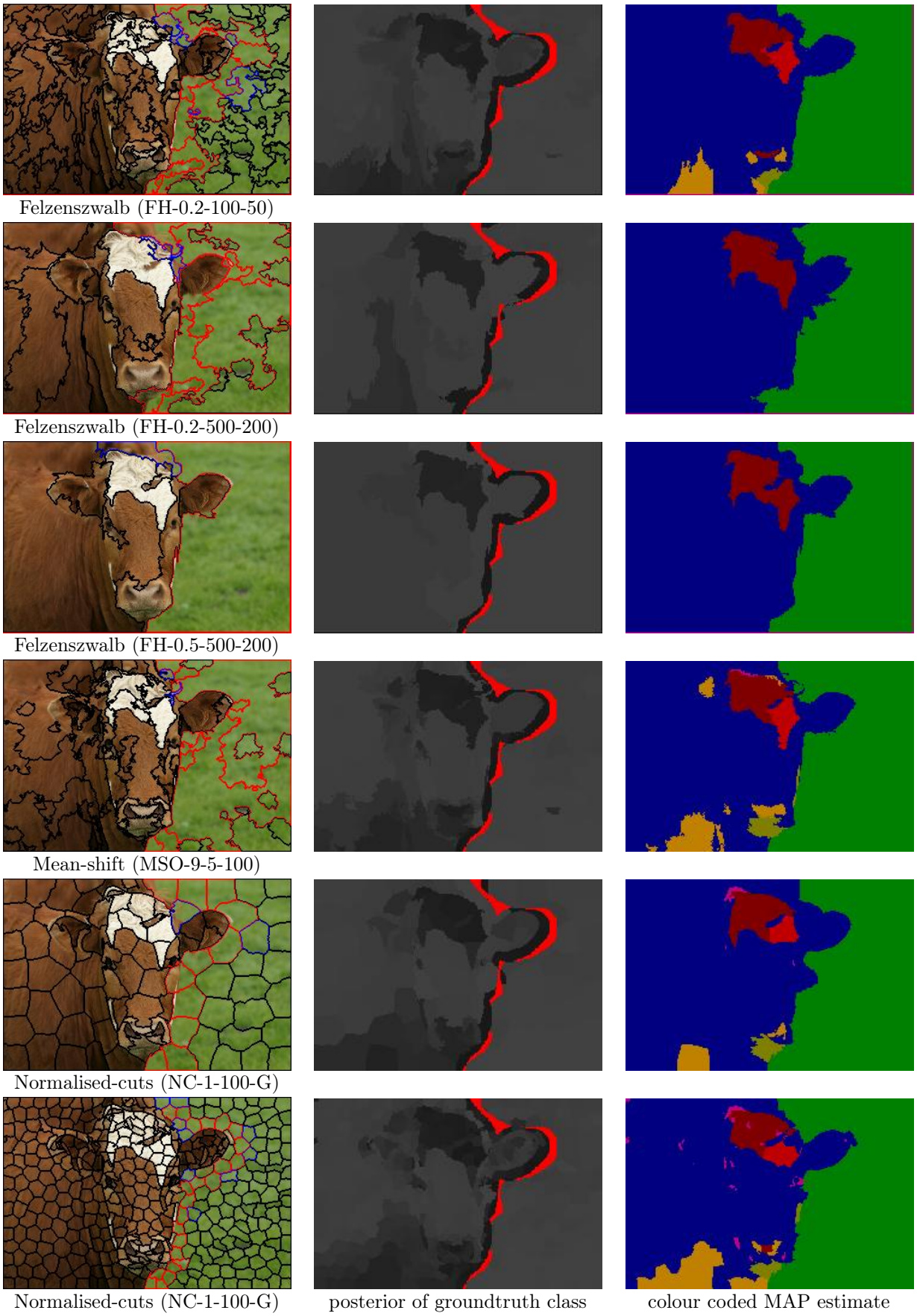


Figure 4.14: Classifying segments. The left column shows the segmented images. Once a segmentation of the whole image has been obtained the image statistics in each segment are gathered and the corresponding posterior class distribution is obtained using our SHCMs. The class posterior for the groundtruth classes are shown in the middle (*void* pixels are shown in red). The right column gives the MAP estimate.

4.5 Pixel Context based on Bottom-Up Segmentations

Here we focus on "extending" the sliding-window approach to a more image data driven approach as mentioned in Section 4.1.2 and Figure 4.1(c) on page 62. The main idea of the sliding-window approach is to mask an image area, the inside of the sliding-window, and classify its content to one of the object-classes. As already pointed out, one of the main drawbacks is that, since the window is fixed in size, it can contain multiple object categories. One possible solution of this problem was introduced in the previous section with the mixture models. In this section we propose another method. Instead of having fixed sized rectangular regions, each image is segmented into multiple regions using various approaches to bottom-up segmentation. The idea is to retrieve multiple over-segmentations[3] for each image. A short introduction into bottom-up segmentation algorithms is now given, and we then evaluate the suitability of this approach to semantic image segmentation using SHCMs as the model of choice. Technically, any visual model that classifies image areas can be used.

4.5.1 Introduction to Bottom-Up Segmentation

There is a variety of bottom-up image segmentation techniques. For the experiments here we focus on three algorithms: `normalised-cuts` [Shi and Malik (2000), Cour *et al.* (2004)], `Felzenszwalb` [Felzenszwalb and Huttenlocher (2004)], and `mean-shift` [Georgescu *et al.* (2003)]. Other common techniques, which did not perform well in our setting, are watershed segmentation (edge based) [Beucher, 1992] and k-means segmentation (colour, texture, and location based).

Figure 4.14 shows some example segmentations from the three methods we focus on in the left column. The borders of the segments are colour-coded with the following meaning: *black*=contains one object-class only; *red*=contains multiple object-classes; *blue*=contains an object-class and areas labelled as void. Note that the images are mostly over-segmented. Different segmentation parameters introduce differently shaped/sized segments for the same segmentation algorithm. It is obvious that the `Felzenszwalb` and `mean-shift` methods produce more "image driven" segments, whereas the `normalised-cuts` approach results in equally sized

[3]Here over-segmentation refers to the fact that segments break objects apart and ideally only one type of object is contained within each segment.

segments, similar to super-pixels [Ren and Malik (2003)].

The abbreviations for the segmentation parameters are as follows:

FH-0.2-100-50: Felzenszwalb and Huttenlocher (2004) segmentation algorithm. Their method defines a graph over the image pixels and finds the globally optimal segmentation, according to their criteria, using a maximum spanning tree based algorithm. The main parameters are: $\sigma = 0.2$ defines a Gaussian blurring of the image which, if small enough, removes artifacts only; $k = 100$ sets the scale of observation, where a larger k encourages larger segments; smallest segment=50 sets the minimum size of a segment and is enforced in a post-processing step;

MSO-9-5-100: Comaniciu and Meer (2002), Georgescu *et al.* (2003) introduce adaptive `mean-shift`, using locality-sensitive hashing to reduce the computational complexity of the method. The following parameters are available: $\sigma_{spatial} = 9$ defines the bandwidth for the spatial (x,y) subspace; $\sigma_{range} = 5$ sets the bandwidth for the range (LUV) colour subspace; smallest segment=100 defines a minimum size for the segments;

NC-1-100-G: Shi and Malik (2000), Cour *et al.* (2004) introduce `normalised-cuts` as a graph partitioning method that is based on the generalised eigenvalue problem and is applied to a graph defined by a global criterion. The parameters are: basis-size=1 gives an option to resize the image before application of normalised-cuts; #segments=100 defines the number of segments returned by the algorithm; G="grey-level image" means that the grey-level version of the image is used (intensity image);

4.5.2 Combining Multiple Segmentations

Given an image segmentation S_j the image regions defined by the segments can be classified. The texton histogram $\mathbf{h}^j$ for each segment in S_j is computed and classified using Kullback-Leibler divergence (see Section 4.1.3). This defines a posterior distribution over the object-classes for each pixel p. Figure 4.14 on page 85 shows such object-class posteriors for the groundtruth class in the middle row. We now use a set of segmentations $\{S_1, \ldots, S_n\}$ to assign n posterior distributions $\mathbf{d}^{j,p}$ to each pixel p. To retrieve the final classification c_p for each pixel

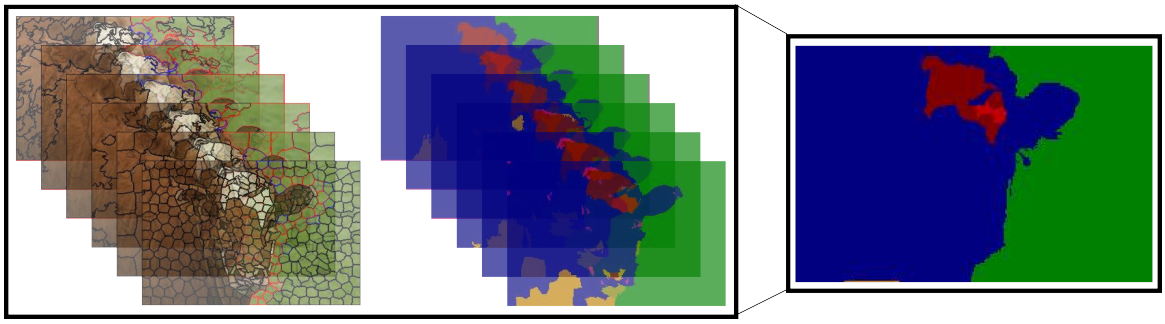


Figure 4.15: Combination of multiple segmentations. A set of segmentations (different algorithms and parameters) is used to improve the results over a single segmentation. Each segmentation results in widely differing independent image segmentation boundaries and thus classifications of those segments. Combining the various classifications obtained for each pixel improves the overall performance.

NC,1,100,G	NC,1,200,G	NC,1,300	NC,1,500	FH,0.8,800,80	FH,0.2,1000,200	FH,0.2,100,50	
71.8%	71.2%	70.8%	70.2%	71.2%	71.2%	70.9%	
FH,0.2,200,100	FH,0.2,200,200	FH,0.2,500,200	FH,0.2,50,50	FH,0.5,500,200	MSO,9,5,100	MSO,5,9,100	combined
72.2%	72.6%	72.7%	70.7%	73.7%	71.1%	71.7%	76.8%

Table 4.6: Segmentation results. The table gives the pixelwise classification performance if only one segmentation was used. The parameters are as described in Section 4.5.1. In comparison it can be seen that combination of these 14 segmentations results in a much higher performance.

(shown in Figure 4.15 on the right), the MAP estimate of

$$P(c_p|\{S_1,\ldots,S_n\}) \propto \prod_{j=1}^{n} \mathbf{d}^{j,p}$$

is computed. This corresponds to a product of experts model (also see Section 2.2.2) defined by the multiple segmentations.

4.5.3 Experimental Evaluation

The following experiments show that this data driven pixel context selection is beneficial over the crude approach of the sliding-window method. We show that a *single* bottom-up segmentation does not improve over the sliding-window method, but the combination of several different segmentations results in a clear improvement.

All experiments were carried out on the 9-class MSRC dataset using the previously introduced SHCMs with 8000 visual-words. Figure 4.17 on page 90 gives segmentation results retrieved by combining 14 different segmentations. In comparison with Figure 4.7 on page 76 it

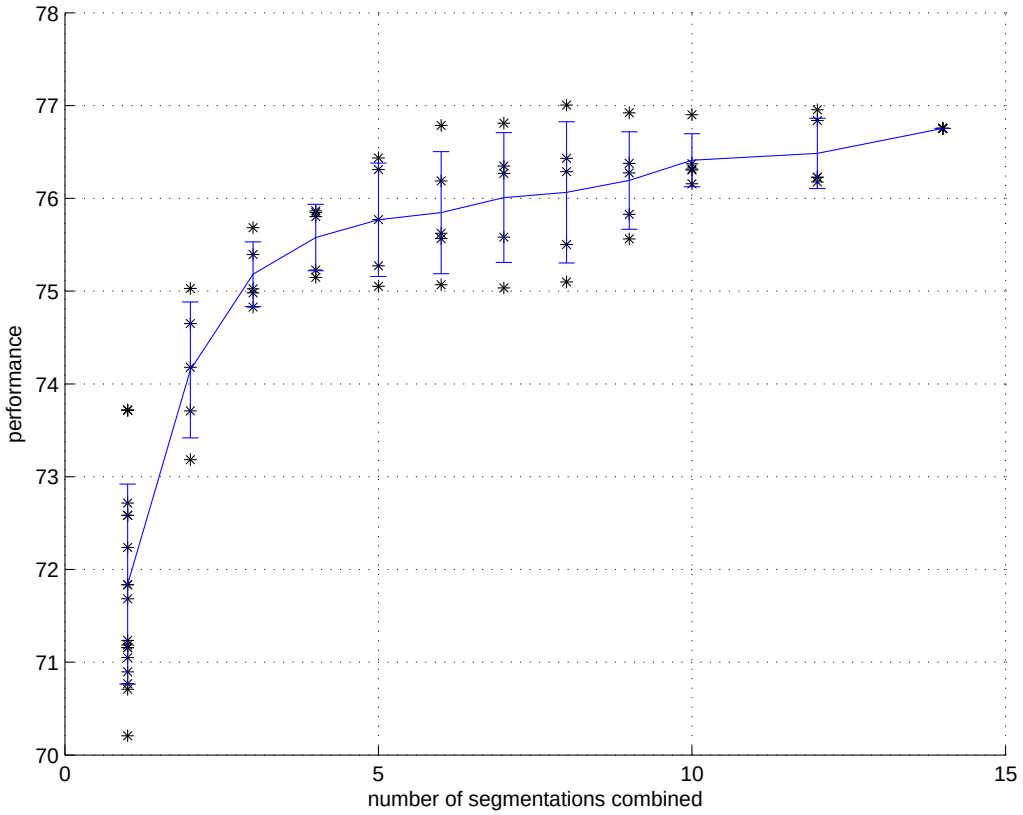


Figure 4.16: Combination of segmentations. This experiment uses a set of 14 segmentations $\mathcal{S} = \{S_1, \ldots, S_{14}\}$. Each entry gives the mean and one standard deviation of randomly sampled combinations C of n segmentations combined, *i.e.* $C \in \mathcal{P}(\mathcal{S})$. The individual data points show the performance of each combination. It is clear that the performance increases dramatically up to the combination of 5-10 segmentations.

is obvious that the final object-segmentations follow the object boundaries more closely and the "blurring" effect caused by the sliding-window can be avoided. The segments still "bleed" into the surroundings, which results in jagged object boundaries as can be seen in the top image in Figure 4.17.

Figure 4.16 and Table 4.6 give an idea of how important it is to use multiple segmentations with different parameters. The table shows that most of the segmentations perform equally well if used individually. However, the combination gives a significant improvement over the individual performances and also outperforms the sliding-window approach (multiple segmentations: 76.8% *vs.* sliding-window: 75.2%). The plot visualises the improvement of the pixelwise classification with increasing number of segmentations that are combined. For each number the mean and one standard deviation are shown, computed over randomly sampled subsets of the 14 segmentations.

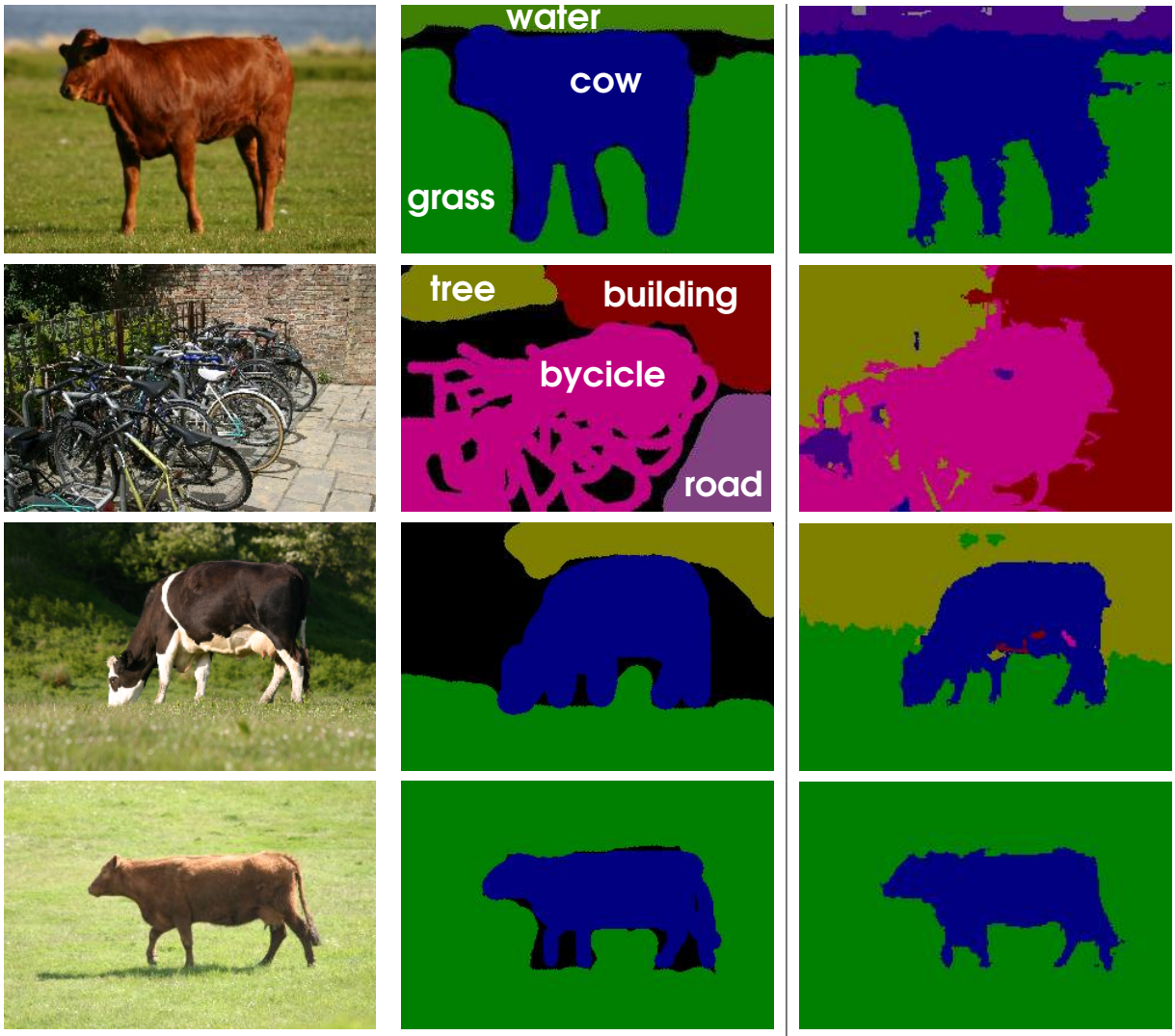


Figure 4.17: Combining multiple segmentations results. (a) Original photographs. (b) Groundtruth class-labels. (c) Output class maps obtained with KL divergence, when the class posteriors of 14 different segmentations where combined. See Figure 4.7 on page 76 for comparison.

4.5.4 Discussion

We have shown that using multiple segmentations instead of a sliding-window can improve performance. Additionally, the resulting image segmentations are visually more pleasing when the bottom-up segmentation scheme is used, in contrast to the sliding-window classifier which results in more "blurred" object segmentations. Although classifying the segments is faster than using the sliding-window method, retrieving the multiple segmentations itself can be quite slow and thus the overall system can be significantly slower depending on the type and number of segmentations that are used.

Other methods for combining the posteriors of the segments were investigated. Specif-

ically, we looked at belief propagation based approaches in order to keep the posteriors of neighbouring segments (and overlapping segments over multiple segmentations) consistent. It did improve good segmentations further, but could degrade worse segmentations as it affected the neighbouring segments as well. It is possible to take the agreement of the classifications for overlapping segments into account and thereby retrieve a confidence score for each pixel. Technically, this could also be done for the sliding-window method, but given the data driven bottom-up context it might be more meaningful in this setup.

4.6 Conclusion

In this chapter we have introduced a new technique for the estimation of compact and efficient, generative single-histogram models of object-classes. Those models were applied to simultaneously segment and recognise objects in images. Different histogram similarity functions have been compared. In the case of single-histogram class models, the Kullback-Leibler divergence has been demonstrated to achieve higher accuracy than widely used alternatives such as $L2$ and χ^2 distances. Despite their simplicity and compactness, the SHCMs have proved as discriminative as keeping around *all* exemplar histograms (and classifying via nearest neighbour approaches). Their main advantages are their storage economy, computational efficiency, and scalability. We note that the computational efficiency is a significant advantage, since methods for speeding up nearest neighbour search, such as k-D trees, do not perform well in high dimensions. Here the number of dimensions equals the number of histogram bins and is of the order of thousands. Thus, finding the closest exemplar (in k-NN classification) reduces to a linear search through all the exemplars, whilst for single-histogram class models the search is only linear in the number of classes. The more recent work of Boiman *et al.* (2008) proposes a similar approach and argues that image-to-class nearest neighbour matching leads to a strong classifier. They also argue that quantisation can hurt nearest neighbour classification, which corresponds to our findings, where the performance increased with larger codebook sizes. Instead of employing a form of nearest neighbour matching, as we do, it would also be possible to train a classifier, such as SVM, on the texton histograms extracted by the sliding-window, or to train a multi-class logistic regression model. Preliminary results with the logistic regression model show a significantly worse performance. In the *region-level classification* the logistic

regression model performs perfectly on the training data, but compared to the SHCMs significantly worse on the test data. This suggests that the logistic regression overfits and therefore does not generalise well. It needs to be pointed out that the number of training exemplars (269) is not very large considering the number of classes (9) and weights (1024 visual words in this experiment, due to memory limitations) to be estimated.

The efficiency of our method plays an important role in the next chapter, where the single-histogram class models will be employed in the Random Forest framework. Chapter 5 focuses on fundamentally different techniques, but some of the concepts presented here will be used. We introduced two possible solutions to overcome the problem of multiple object categories occurring inside the sliding-window region: *(i)* the multi class mixture model, and *(ii)* the approach based on multiple bottom-up segmentations. Chapter 5 extends the sliding-window method into yet another direction. Random Forests are used to learn the "context" for a pixel, *i.e.* the shape and relative position of the rectangular regions, discriminatively. The following chapter also reports results for the CRF based post-processing step (see Section 2.2.4). This additional step could also be applied to the results in this section to give a further improvement.

Chapter 5

Semantic Segmentation via a Discriminative Model

This chapter leverages the generative texton models from the previous chapter by embedding them into a discriminative classifier, the Random Forest. So far segmentation was accomplished by means of a fixed sliding-window to define the context of an image pixel and consequently classifying it. We also investigated the use of bottom-up segmentation to define the pixel context. Here we use the Random Forest classifier to select discriminative Haar-like feature kernels. In addition to the patch based texton features additional low-level features are employed (HOG and RGB). All these features are combined, taking advantage of the excellent feature selection properties of the Random Forest classifier.

Before we propose and evaluate the fully fledged system, the single-histogram class models (SHCMs from Section 4.2) and the nearest neighbour classifier from the previous chapter are formulated in the Random Forest framework. This way we can provide insight into the discriminative power based on the example of feature kernel selection, or more specifically, selection of the pixel context. We express the different feature kernels using a common description which enables us to discuss their relationships more easily. The impact on the performance of low-level features such as HOG, RGB, and SHCMs is evaluated on the segmentation task, using the common MSRC and VOC2007 datasets (Chapter 3). Our system shows state of the art performance on this task. As mentioned before Random Forests carry good multi-class classification capabilities, and in addition to the discriminative classification empirical class posteriors are returned as well. In Section 5.4 it is shown how these posteriors can be used in the CRF

framework (see Section 2.2.4) to further improve the segmentation of objects in an image.

5.1 Random Forests for Object Segmentation

As presented in Section 2.2.2, Random Forests are suitable for multi-class classification tasks and the simplicity of the node-tests allows for a wide range of possible input feature kernels and low-level feature descriptions. We first introduce a feature description and feature kernels that are suitable for the object segmentation task and are thus able to take advantage of the feature selection abilities inherent to the Random Forest classifier. The feature kernels we present in the next section are related to the Haar-like features used by Viola and Jones (2001). Our general description enables us to point out the relations to the node-tests used by others: pixel intensity comparisons in Lepetit and Fua (2006), pixel differences in Shotton *et al.* (2008) or texton "responses" in Shotton *et al.* (2006), Savarese *et al.* (2006). We then transfer the classifier from Section 4.2 into this same framework.

Given these features the Random Forest is trained by learning a set of decision-trees separately. These trees can be trained on a sub-sampled set of all the labelled training pixels, to further improve their independence [Breiman (2001)]. The pool P of node-tests is always randomised. Testing proceeds by applying all trees of the Random Forest to each input pixel p in the test image. The per-pixel class posterior is achieved as the average of the posteriors across all trees, as defined in the Random Forest section. Now the formulation of the node-tests is introduced.

5.1.1 A General Node-Test

Previously used node-tests[1] as in Shotton *et al.* (2006) and Yin *et al.* (2007) as well as our node-tests based on the models from Chapter 4 can be described in one framework which is presented here. In Section 5.3 we point out the correspondences between the various possible node-tests and give a unified interpretation.

[1]The node-test is the test that is performed in each node to determine if the data point is passed to the left or right child-node, also see Section 2.2.2 for the basic definition.

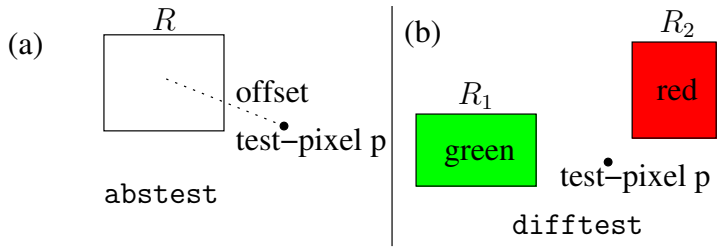


Figure 5.1: Node-tests. (a) A node-test can be based on filter responses accumulated over a single rectangular region as in Shotton *et al.* (2006). (b) Alternatively, responses accumulated over two (or more) rectangles can also be used, *e.g.* Yin *et al.* (2007). The pixel differences in Lepetit and Fua (2006) and Haar-like responses in Viola and Jones (2001) are special cases of this general description.

For a pixel in position p in image I, a general node-test t_p may be described as:

$$t_p = \sum_{r \in \mathcal{S}} \mathbf{w}^r \cdot \mathbf{f} \tag{5.1}$$

where r indexes one or two rectangles (*i.e.* $\mathcal{S} = \{1\}$ or $\{1, 2\}$), $\mathbf{w}^r$ describes a filter/mask selecting the pixels in the rectangle R_r and feature channel C_j. It defines a weighting for each dimension of the feature vector $\mathbf{f}$. This feature vector describes the feature channels C for each pixel in the image. For example, given an RGB-colour image I of size $|I| = 120 \times 200$, $\dim(\mathbf{f}) = 120 \times 200 \times 3$ and $C_j = \{1 + (j-1) \cdot |I|, \ldots, j \cdot |I|\}$ for the three colour channels C_1, C_2 and C_3.

$\mathbf{w}^r$ acts as a mask and contains 1 for the corresponding $i \in \{1, \ldots, |I|\} \cap R_r \cap C_j$ and 0 otherwise, we write this as $\mathbf{w}^r_{i|R_r C_j} = \mathbf{1}$. The rectangles R_r are positioned relative to the test-pixel p and thus $\mathbf{w}^r$ also depends on p. The size and relative position of rectangles R_i greatly influences the shapes and the amount of context that is captured or can be discriminated by the node-test.

If two rectangles are used ($\mathcal{S} = \{1, 2\}$), often the difference of the accumulated responses of the two rectangles is computed as in Yin *et al.* (2007). This is illustrated in Figure 5.1(a,b) and an example picture with the responses is given in Figure 5.2. Note that this accumulated response, together with the threshold λ, defines the node-test (2.6) on page 96. This threshold is set independently for each node-test (rectangle configuration) and corresponds to a threshold on the mean response in the rectangular area or on the weighted difference of mean responses for two rectangles. This general description encompasses the linear classifier used in Bosch *et al.* (2007b), where pyramid histograms of visual-words are used as feature $\mathbf{f}$ to describe

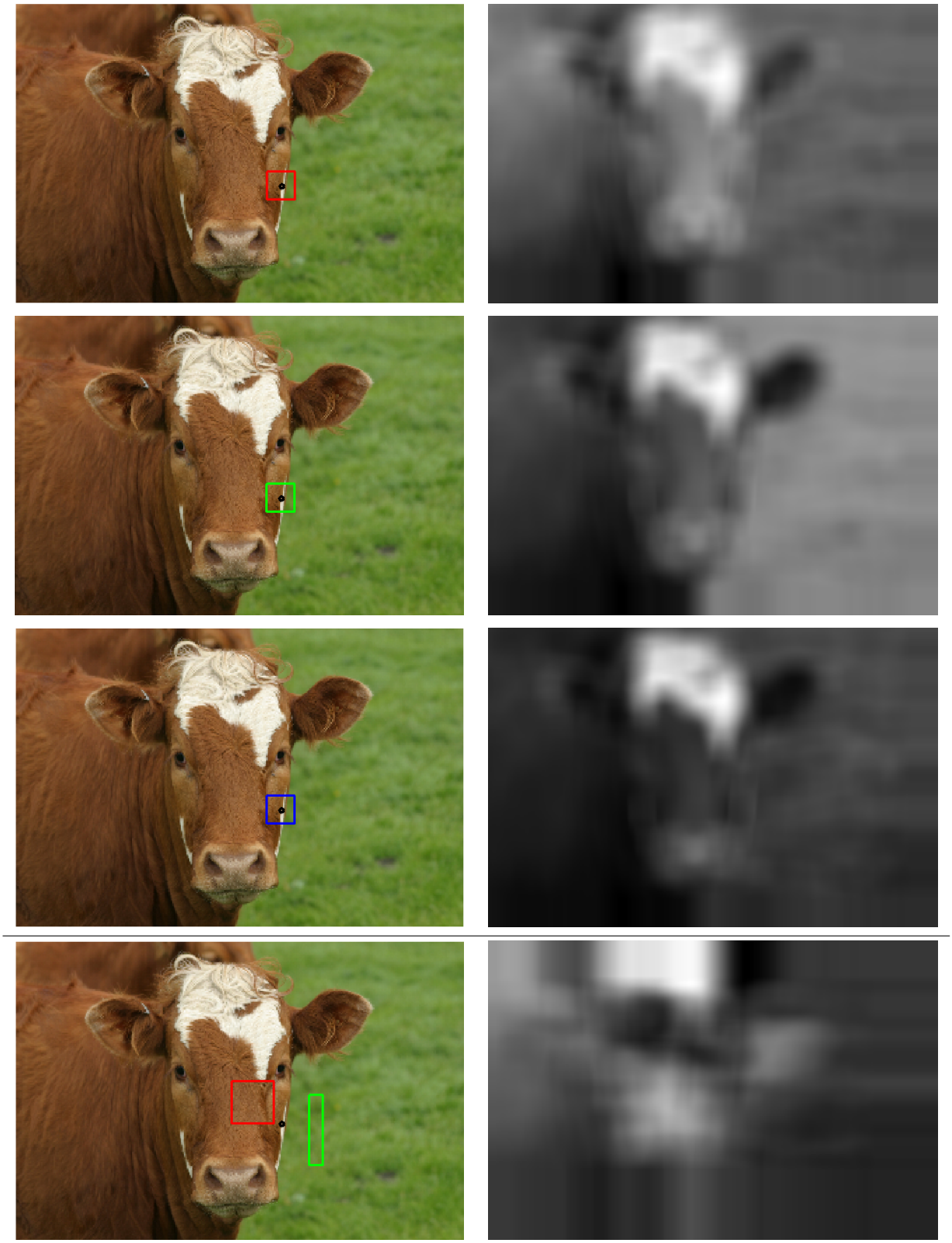


Figure 5.2: Pixel-differences. The three top rows visualise the response (right column) of a node-test that is described by a centre 20×20 pixel rectangle (left column). Shown are the three versions for the *red*, *green*, and *blue* channel (top to bottom). The bottom row shows a node-test which computes the difference between the red-channel response in R_1 and the green-channel in R_2, *i.e.* $\mathbf{w}^1_{i|R_1} = \mathbf{1}$ and $\mathbf{w}^2_{i|R_2} = -\mathbf{1}$ (see text). The test pixel p is marked by the black dot. The right image shows the corresponding response.

the whole image. Thus (5.1) simplifies to $t_p = \mathbf{w} \cdot \mathbf{f}$ and the weight vector $\mathbf{w}$ is filled with random weights. The pixel-differences used in Lepetit and Fua (2006), Winn and Criminisi (2006), Shotton *et al.* (2008) correspond to $\mathbf{w}^1_{i|R_1C_3} = \mathbf{1}$ and $\mathbf{w}^1_{i|R_2C_1} = -\mathbf{1}$ where the two rectangles degrade to two points positioned relative to p, and the red channel C_1 is subtracted from the blue channel C_3, for example. In Shotton *et al.* (2006, 2008) for example, the feature vector $\mathbf{f}$ corresponds to a texton assignment to each pixel. Responses over one rectangle are described by the count of a specific texton occurrence in the rectangle, *i.e.* $\mathbf{w}^1$ masks the positions in the rectangle R_1 that correspond to a specific texton ID/feature channel C_j.

Next we introduce how the SHCMs from the previous chapter can be used in this framework, and in Section 5.3 we relate how the commonly used special cases and low-level image features from Section 2.1 fit into this formulation.

5.2 Generalising Single-Histogram Class Models in Random Forests

In this section we embed the single-histogram class models from Chapter 4 into the Random Forest framework. We show first how the nearest neighbour classifier can be implemented by a Random Forest (decision-tree) and then discuss how the generalisation of this framework leads to the features commonly used in Random Forests. Note that the single-histogram class model based nearest neighbour classifier of Chapter 4 is *not* learnt discriminatively and does *not* learn spatial information about the object-classes. It only captures crude context by means of the sliding-window. Both these shortcomings are overcome by the use of Random Forests and their feature selection capabilities.

5.2.1 Casting the Nearest Neighbour Classifier into Decision-Tree Terminology

In Section 4.2.1 it is shown that given an image patch, the maximum likelihood estimate of its class is equivalent to a nearest neighbour classifier on single-histogram class models, where similarity is measured using Kullback-Leibler divergence. We now cast this nearest neighbour formulation into the Random Forest framework in order to motivate the additional features and

to point out the capabilities of Random Forests. This first step simply reproduces the results from the previous chapter and is only meant to simplify the direct comparison of the basic SHCMs to the capabilities added by the Random Forest classifier. In Section 4.3 the single-histogram class models are used to segment a test image into object-class regions by means of a sliding-window classifier. In detail, a sliding-window s is used to assign the object-class-label $\hat{c}$ to the centre pixel p of s. With $\mathbf{h}$ being the distribution over textons in window s and $\mathbf{q}^c$ being the single-histogram class models (SHCMs) the classification result for p is $\hat{c}$, as in (4.1):

$$\hat{c} = \arg\min_c \left(D_{KL}(\mathbf{h}||\mathbf{q}^c) \right) \tag{5.2}$$

We first note that finding the nearest neighbour class histogram $\mathbf{q}^{\hat{c}}$ using (5.2) can be accomplished by a series of hierarchical tests that compare the test window histogram $\mathbf{h}$ to a pair of SHCMs and select the more likely one. The goal is to reformulate this pairwise comparison into a single efficient node-test of the form (5.1), that can be used in a decision-tree. This can be achieved by combining two SHCMs $\mathbf{q}^i$ and $\mathbf{q}^j$ into $\mathbf{w}^{i,j} = \log\left(\frac{\mathbf{q}^i}{\mathbf{q}^j}\right)$. Now we define the node-test t_p which compares a test-histogram $\mathbf{h}_r$ (computed from one rectangle $\mathtt{R}_r$) to two class models $\mathbf{q}^i$ and $\mathbf{q}^j$ in a single test (5.3):

$$D_{KL}(\mathbf{h}_r||\mathbf{q}^i) < D_{KL}(\mathbf{h}_r||\mathbf{q}^j) \quad \Leftrightarrow \quad t_p < 0 \text{ with } t_p = \mathbf{w}^{i,j} \cdot \mathbf{h}_r \tag{5.3}$$

Note that as opposed to the examples given in Section 5.1.1 here $\mathbf{w}^{i,j}$ is not just 1 or -1, but each feature channel (here texton) is weighted with a real numbered weight induced by the object-class models (SHCMs). The node-test t_p defined in (5.3) is smaller than 0, if and only if class model i is more likely to explain the rectangle $\mathtt{R}_r$ than class model j.

If there are C object classes, *i.e.* C SHCMs $\mathbf{q}^c$ with $c \in \{1, \ldots, C\}$, a tree of depth $C-1$ can compute $\arg\min_{\mathbf{q}^c} \left(D_{KL}(\mathbf{h}||\mathbf{q}^c) \right)$, by means of a series of tests as defined in (5.3) (see Figure 5.3). These tests need to compare all possible pairs except the ones that can be excluded due to transitivity. This scheme is called *fixed-tree* in the following. One might think that using such comparisons the number of tests can be reduced to $\log(C)$, however, in general nearest neighbour search cannot be performed in $\log(C)$. See, for example, Arya *et al.* (1998) for

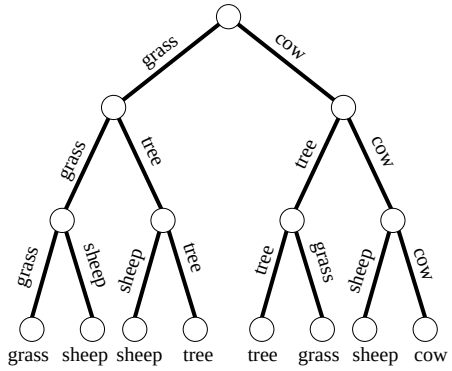


Figure 5.3: Fixed-tree. Each node compares the (so far) $\arg\min_{\mathbf{q}^c}$ to a $\mathbf{q}^c$ that is not included in any of the parent tests in that branch of the tree. The test at the root node compares an arbitrary pair of $\mathbf{q}^i$ and $\mathbf{q}^j$. Instead of C tests that are needed for the naive computation of the minimum only $C-1$ tests (the depth of the tree) are needed here. Here the tree classifies into one of the four classes (grass, cow, tree, sheep*)* and thus needs to perform 3 tests.

complexity bounds for approximate nearest neighbour searches or Friedman *et al.* (1977) for kd-trees to find best matches in logarithmic expected time.

Experiments. A few simple experiments are carried out to confirm the equivalence to the nearest neighbour framework to the single fixed decision-tree. The following experiments were carried out on the 9-class MSRC dataset (Chapter 3).

The posteriors in the leaf nodes of this tree are set manually to give a classification for the *one* object-class defined by the series of tests in that particular branch of the tree. As expected this confirms the classification performance also reported in Section 4.3. Combining multiple fixed decision-trees each using a different sliding-window size w results in a marginally improved performance.

In the second experiment *only* the node-tests in the decision-tree are fixed and the empirical class posteriors are learnt as laid out in Section 2.2.2. This results in virtually the same performance 75.2% as in Section 4.3 with the same features (25×25 pixelwide centred windows). This is a nice result indicating that the Random Forests learns good empirical class posteriors that lead to the same MAP estimate as the minimisation of $D_{KL}(\mathbf{h}||\mathbf{q}^c)$.

The third experiment was evaluated on the 9-class dataset as well and instead of using the fixed tree of depth 8 (9 classes - 1), we learn a Random Forest keeping the rectangle size fixed to 25×25 and centred on the test-pixel (comparable to the sliding-window method from Section 4.3), *i.e.* the node-tests compare $D_{KL}(\mathbf{h}_r||\mathbf{q}^i) < D_{KL}(\mathbf{h}_r||\mathbf{q}^j)$ for a pair of SHCMs (i,j), that is learnt during training, and the query-histogram $\mathbf{h}_r$ computed over the 25×25 rectangle.

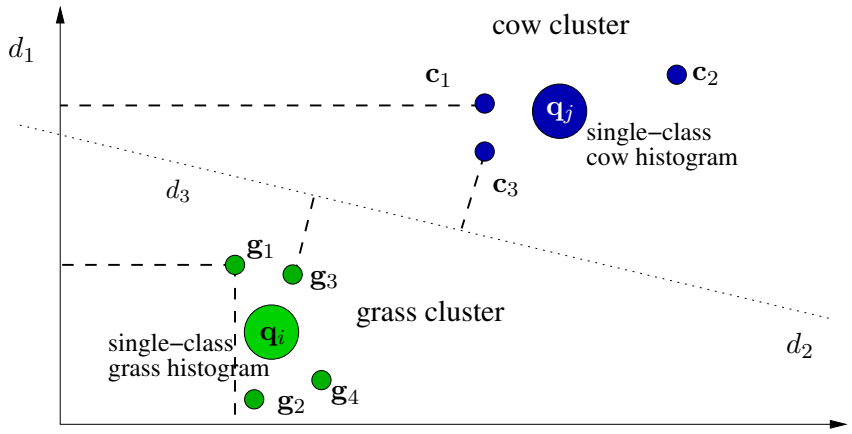


Figure 5.4: Node-tests. This figure visualises a two dimensional feature space (*e.g.* two visual-words). The dots represent single-class histograms and histograms computed from the rectangles (*i.e.* $\mathbf{h}, \mathbf{q}^i, \mathbf{q}^j$). d_1 and d_2 denote the coordinate axes, and d_3 defines a different projection direction, *e.g.* the one defined by $\mathbf{w}^{i,j} = \log\left(\frac{\mathbf{q}^i}{\mathbf{q}^j}\right)$ for a pair i, j of object-classes.

This Random Forest consists of 10 decision-trees, with each tree having a maximum depth of 15. The number of node-test that is optimised over for each node is 100, *i.e.* $\#fn = 100$. Incorporating the flexibility to select the pair (i, j) in each node-test increases the performance marginally to 76.2% (see Figure 5.6 on page 104). One possible explanation for this small improvement could be that a bias is "learnt". This bias would avoid comparison of visually close object-classes that do not occur together in the same image and do not need to be compared directly in a node-test. For example, cows and sheep do not occur in the same image and thus a direct comparison of those two object-classes could introduce misclassifications.

To conclude we can say that the fixed decision-tree reproduces the nearest neighbour classifier. The fact that relaxing the tree to learn the class posteriors and a further relaxation to non-fixed trees perform equally well or slightly better than the nearest neighbour maximum likelihood classifier (as derived in Section 4.2.1) confirms that the Random Forest in this setup is a MAP classifier and that the empirical class posteriors estimated in the leaf nodes represent the texton distributions for the object-classes very well indeed.

5.3 Relationship of Various Feature Types

Starting with our general description from (5.1), the previously introduced node-tests based on SHCMs and the pixel-differences (Section 5.1.1) we can relate and interpret those seemingly very different feature types.

Consider the case where the SHCMs $\mathbf{q}^i$ and $\mathbf{q}^j$ have identical counts (*i.e.* similar occurrence frequencies) in all but one bin/channel. Then $\mathbf{w}^{i,j} = \log\left(\frac{\mathbf{q}^i}{\mathbf{q}^j}\right)$ will be zero for all but one weight. This is the limiting case of a single discriminative texton between the two classes. Thus single-histogram class models are closely related to weak classifiers (*i.e.* node-tests) used by others, namely: *TextonBoost* (Shotton *et al.* (2006)), where textons that are discriminative for a pair of classes are learnt (in that case using boosting), *i.e.* all weights but one are zero in (5.3); and the linear classifier (with a set of non-zero $\mathbf{w}^{i,j}$s) used in the node-test of Bosch *et al.* (2007b).

Figure 5.4 gives a geometric interpretation of the various special cases of node-tests. All those node-test are described by (5.1). The following points out the properties and differences between the two main node-tests we use: *gatetest* and *difftest*. These node-tests correspond to special cases of the general formulation where *gatetest* acts like a gating function putting higher weights on more discriminative textons. *Difftest* describes the commonly used differences of responses computed over rectangular regions in different channels.

Gatetest. This corresponds to the node-test described in the previous section. *Gatetest* uses two fixed histograms as "reference points" and computes the difference in KL divergences between the query-histogram (*e.g.* $\mathbf{h} := \mathbf{c}_1$) and two single-class histograms (*e.g.* $\mathbf{q}^i := \mathbf{s}_c$ and $\mathbf{q}^j := \mathbf{s}_g$), *e.g.* $D_{KL}(\mathbf{c}_1||\mathbf{s}_c) - D_{KL}(\mathbf{c}_1||\mathbf{s}_g)$. Alternatively this can be seen as a projection onto a direction d_3 which is defined by $\mathbf{w}^{i,j} = \log\left(\frac{\mathbf{q}^i}{\mathbf{q}^j}\right)$. This is obvious as t_p in (5.3) is the inner product between the two vectors and thus interpretable as a projection onto $\mathbf{w}^{ij}$. In a sense $\mathbf{w}^{ij}$ is a gating function to select the discriminative textons. Feature dimensions that are similar in both single-class histograms have small influence and dimensions that are very different have a large influence. Thus, $\mathbf{w}^{ij}$ defines a projection onto a direction with relatively high linear discrimination between the two classes. This relates to the linear classifier as used in Bosch *et al.* (2007b), but here the hyperplane is induced by the SHCMs instead of being learnt. Using a wisely chosen projection direction as Fisher's linear discriminant might be another option. It is obvious that $\mathbf{w}^{ij}$ does not need to be based on single-class histograms, but could be based on pairs of exemplar histograms[2], for example. If we approximate even further, instead of exemplar histograms two histograms computed over rectangles R_1 and R_2 can be used. If one of those rectangles lies inside one object (*e.g.* cow) and the other one inside another object

[2] *I.e.* the histograms computed over the training exemplar regions (one histogram per object-class per image).

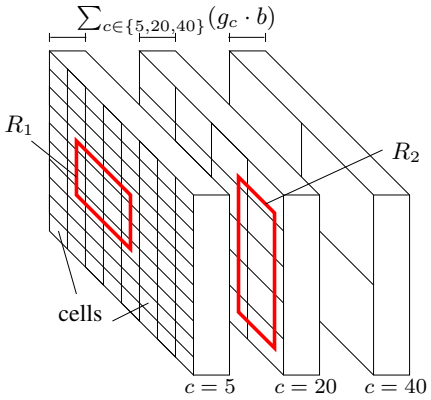


Figure 5.5: HOG features. We use stacked HOG features (see Section 2.1.1) computed with different cell-sizes $c = \{5, 20, 40\}$ and gradient bins $g = \{6, 6, 12\}$. One block always contains 4 cells ($b = 4$). Each rectangle response is computed by adding the responses for one of the HOG dimensions of a pixel. Cells partially included in the rectangle area contribute proportionally to the included area.

(*e.g.* grass), then it would approximate the two exemplar histograms and thereby the SHCMs case. If we additionally consider the limiting case of only one discriminative texton it becomes clear that the difference between the texton counts for R_1 and R_2 is proportional to the weight $\mathbf{w}^{ij}$. The case of one rectangle corresponds to the test used in TextonBoost. Theoretically (5.1) also captures multiple rectangles, but the computational complexity increases drastically and experiments haven't shown a significant improvement.

Difftest. Corresponds to a projection of the histogram onto one dimension of the feature space. This projected value can be used directly as in the texton based weak classifiers of Shotton *et al.* (2006). Alternatively, two of these projected values from the same or different dimensions can be compared (*e.g.* $t_p = d_2(\mathbf{g}_1) - d_1(\mathbf{g}_1)$ compares the projection of feature vector $\mathbf{g}_1$ onto dimension d_2 to the projection of d_1 as illustrated in Figure 5.4 on page 100). This is closely related to the node-tests laid out in the previous paragraph, but it enables us to employ virtually any low-level feature that defines a per-pixel response with possibly multiple dimensions/feature-channels in the test image.

We evaluate the following features in the experiments section:

- **RGB features**: The node-tests are simple differences of responses computed over rectangles in one of the three channels (R, G, or B), see Figure 5.1(b) on page 95. These

features are used by Lepetit and Fua (2006), Winn and Criminisi (2006), but they use simple pixel-differences instead of responses over rectangular areas.

- **F17 filter bank**: The 17 dimensional filter bank described in Winn *et al.* (2005) is used as an additional cue in the same manner as the RGB features.

- **HOG features**: Since RGB and F17 based Random Forests are not strong enough to capture the spatial variation of classes, we also introduce features based on the HOG descriptor of Dalal and Triggs (2005). The HOG descriptor is computed for the whole image using various cellsizes c in pixels, blocksizes b in cells, and number of gradient bins g. This leads to a $g \cdot b$ dimensional feature vector for each cell (see Figure 5.5). The stacked HOG consists of $c = \{5, 20, 40\}$ and $g = \{6, 6, 12\}$ oriented gradient bins for each of the c values (with $b = 4$ cells in each block), resulting in $6 \cdot 4 + 6 \cdot 4 + 12 \cdot 4 = 96$ channels for each pixel p. It was important to have cellsizes c ranging from small 5 to large 40. Aside from that the results were not very sensitive to the choice of parameters. The performance of HOG alone and combination with other features is shown in Section 5.4.

- **Textons**: Finally, textons can be used directly without SHCMs, mentioned as the limiting case in "gatetest". The straight forward way of using textons corresponds to the usage of the previously introduced feature channels, *i.e.* each texton is treated as a "feature channel" and the accumulated response in one rectangle defines t_p and is compared to a threshold λ. This method is used in Shotton *et al.* (2006, 2008).

5.4 Parameter Evaluation and Segmentation Results

This section evaluates the previously introduced ideas on standard labelled datasets introduced in Chapter 3. It is shown that our methods perform comparably to the state of the art. The main points that are investigated are the importance of spatial context, the influence of the Random Forest parameters, and the combination of several low-level features.

First the influence of the rectangle size and shape together with the offset (Figure 5.1 on page 95) are evaluated. Then, we carry out a range of experiments that shed light on the importance of the various parameters that determine the properties of this classifier, namely the influence of the number of decision-trees and their depth, as well as the variation in performance

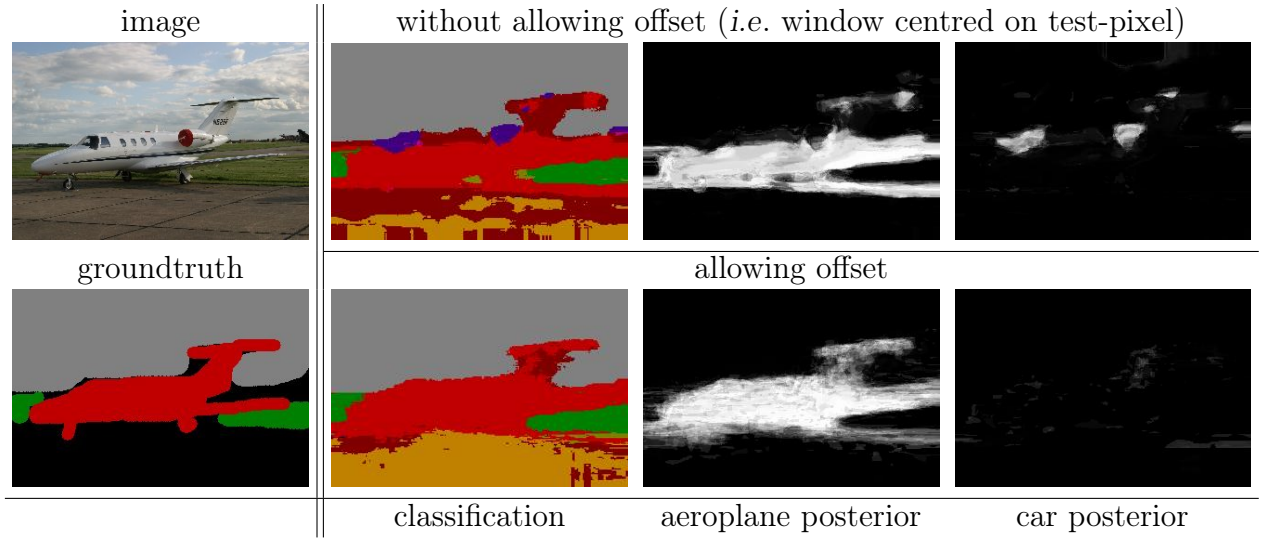


Figure 5.6: Spatial context. This figure shows how an offset incorporates spatial context. The classification with a maximum offset of 30 is in the bottom row, the one with the test-pixel centred in the rectangle in the top row. Next to the classification are shown the posteriors for the object-class (aeroplane) and the car class (marked "purple" in classification) that is confused with aeroplane, if no context is used.

depending on the "randomness" introduced into each tree. The benefit of the multiple feature types combined in the Random Forest classifier is proved with extensive experimentation on the labelled image datasets. Finally, we give results of the full system with and without the added CRF step (see Section 2.2.4).

5.4.1 Spatial Context: Offset and Windowsize

Starting with the fixed-tree using SHCMs we show that allowing for different rectangle *shapes/sizes* as well as *off centre test-pixels* (see Figure 5.1 on page 95) results in a dramatic performance boost. During training the node-tests added to the initial pool P (see Section 2.2.2) reflect these differently shaped and centred rectangles and the maximisation of the information gain ΔE takes full advantage of the discriminative learning of Random Forests by selecting the appropriately shaped rectangles. The node-tests itself still follow (5.3) in this experiment. This constraint will be relaxed too in the next section.

As in Section 5.2.1, the Random Forests consist of 10 decision-trees, with each tree having a maximum depth of 15. The number of node-test randomly sampled from the pool P is 100, *i.e.* $\#fn = 100$. Out of these 100 node-tests the one that maximises the information gain ΔE is selected for a node. Table 5.1 shows that allowing different *window sizes* (smallest edge of rectangle can be 10, largest 35 pixels wide) improves the performance, as these rectangles can

window size\offset	0	10	20	30
25:25	76.2%	78.8%	81.0%	82.2%
10:35	77.7%	79.3%	80.8%	82.0%

Table 5.1: Spatial influence. The window size [smallest extend:largest extent] describes the range of the width and height of the rectangles added to the node-test pool P. The *offset* describes the maximum distance between the centre of the rectangle and the test-pixel. Allowing for additional "freedom" especially a large offset clearly improves the performance.

adapt to the objects and encode fine/large grained structures better than a fixed window size. The rectangles selected by means of information gain maximisation are around 23 ± 7 pixels each dimension and tend to be slightly smaller towards the bottom of the decision-trees. The average ratio of small side to long side is 0.69 ± 0.19, *i.e.* to capture the full shape of the objects elongated rectangles are advantageous.

However, a significantly larger improvement is achieved by allowing an *offset* of the centre of the rectangles to the test-pixel, in which case the context of the test-pixel is taken into account. In our experiments the offset settles to around 15 ± 8 pixels in the x and y direction. Figure 5.6 illustrates how the Random Forest learns that car is unlikely to occur next to aeroplane and sky. The important observation is that confusions like aeroplane with car are suppressed. This is indicated by the posterior response for the car class, where there are two high response areas in the top right image, but not in the bottom right image. It is worth noting that the classification obtained by the context features (bottom row) has a less crisp object boundary, which can be explained by the fact that pixels occurring near the boundary have a higher likelihood for both classes when context is incorporated (*e.g.* aeroplane/grass confusion). The overall performance however is drastically increased by allowing for this incorporation of context. Together with the shape variations the performance increases from 76.2% to 82.0% (see Table 5.1), exceeding the performance of the original sliding-window based on the SHCMs dramatically.

5.4.2 Number of Decision-Trees and "Randomness"

Figure 5.7 illustrates how the performance depends on the number of trees in the random forest and the amount of randomisation introduced by means of the number of node-tests that is optimised over using information gain. Our experiments showed that the performance does not increase significantly if more than 20 trees are used. We use 20 decision-trees for most experiments and 30 or 50 for the larger datasets (21-class MSRC, and VOC2007). The amount

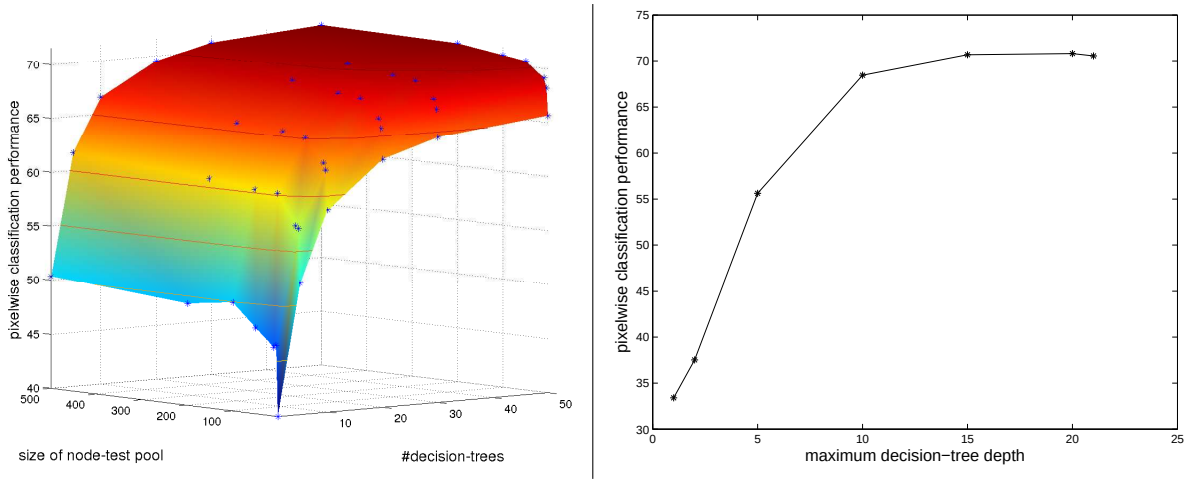


Figure 5.7: Number of decision-trees *vs.* "randomness" and tree depth evaluated on the 21-class MSRC dataset. The left images shows that the number of decision-trees has a great influence on the performance. However, the curve flattens out for more than 20 decision-trees. The amount of randomisation effects the performance much less. If a minimum of optimisation (*e.g.* $\#nf = 100$) is performed for each decision-node the performance does not improve much by further optimisation. The right image shows the performance in dependence of the maximum depth of each decision-tree. The observation here is that the performance increases with deeper trees, and the limit of 20 for most of our experiments is picked out of memory and computational considerations.

of randomness has a much smaller influence. If the number of tests in the per node pool is increased the accuracy improves slightly. This indicates that too much randomisation, *i.e.* no information gain driven selection of the node-tests, hurts the performance. If for example the size of the node-test pool is one, *i.e.* $\#nf = 1$ (see Section 2.2.2), thus selecting the node-test without any regards to the information gain the performance drops. It should be noted that we would expect a decrease in performance, if the size of the node-test pool is further increased. Assuming that the initial pool P is shared between all trees (which is not the case here) an optimisation over the full pool would correspond to *one single* decision-tree, as all decision-trees would be the same. A different initialisation of the pool P for each decision-tree assures further independence among the decision-trees.

5.4.3 Combining Low-Level Feature Types

In this section we describe how the various feature types influence the performance of the Random Forest segmentation task. In Section 5.3 we introduced RGB, HOG, F17 and textons (T) as low-level features. Here combinations of these features are investigated. During training of the Random Forest each low-level feature is treated as a separate pool of features and the

	textons	RGB	HOG	HOG,RGB	HOG,RGB, F17	HOG,RGB, SHCMs	CRF	Others
9-class (pixelwise)	72.8%	72.2%	75.9%	84.5%	86.0%	84.9%	**87.3%**	84.9% [1]
21-class (pixelwise)	40.3%	54.0%	56.3%	69.9%	71.5%	71.7%	**73.7%**	72.7% [2]
VOC2007 (class avg.)	13.6%	10.1%	17.4%	17.7%	19.7%	21.4%	**24.4%**	20.0% [3]

Table 5.2: Feature combinations. Pixelwise classification performance using various combination of features. For the 9-class dataset the forest consists of 20 trees, each of a maximum depth of 20; each node-test is picked as the best of 500 node-tests per feature using information gain. For each node-test the threshold λ which maximises the information gain is selected. For the 21-class dataset a forest of size 30 was used and for the VOC2007 dataset the forest size was 50. The CRF results are based on the HOG,RGB,AC17 features for the MSRC datasets and on the HOG,RGB,SHCMs for the VOC2007 dataset. [1] Verbeek and Triggs (2008), [2] Shotton *et al.* (2006), [3] Shotton *et al.* (2008).

one that maximises the information gain is selected for each node.

Table 5.2 shows the results of various feature combinations. Using all features combined by exploiting the feature selection properties of Random Forests gives state of the art results 84.5% on the MSRC 9-class, and 69.9% on the 21-class dataset. Shotton *et al.* (2006) achieved 69.6% with a boosted classifier. It can clearly be seen that the combination of RGB and HOG gives a *very* strong boost in performance. The addition of SHCMs based texton features (although very strong alone) is not able to increase performance (probably due to redundancy with HOG and RGB), but the discriminative learning assures that the right features are selected and performance does not decrease. Thus the classifier is able to improve over state of the art. Table 5.2 shows the performance of RGB only, and again the flexibility in selecting the shape and offset of the rectangles results in increased performance (72.2%) compared to the fixed window version based on simple visual features (colour-histograms) from Section 4.3.2 with a pixel-wise classification performance of 67.3% (on the 9-class dataset). Sometimes also the average class performance is reported for the MSRC datasets. Shotton *et al.* (2008) report 67% on the 21-class dataset, and Csurka and Perronnin (2008) report 65%. Our system performs worse here (59%), as it underperforms for some classes (*e.g.* boat 0.1%, chair 11%). Those classes are not very well represented in the training data. Thus, it is likely to happen, that during training of the decision-trees the node-tests and empirical class posteriors in the leaf nodes do not represent these classes well enough, as only very few training points lie inside boat regions for example. It is possible to account for this during computation of the information gain which we do for the VOC2007 experiments, but due to the random sub-sampling of the

training data these classes might still be underrepresented. Adjusted sampling might solve the problem, but possibly decreases the overall pixelwise classification accuracy. Also, normalising the empirical class posteriors by the inverse class priors improves the performance in certain cases.

VOC2007 [Everingham *et al.* (2007)]. On the very challenging VOC data our method using RGB, HOG, and F17 achieves state of the art performance (19.7%). The purely Random Forest based system in Shotton *et al.* (2008) reaches an average class performance of 20% similar to our system. The combination of RGB, HOG, and SHCMs results in 21.4% average class performance and improves even further in combination with the CRF (24.4%) in the following section. See Table 5.2 for a comparison. It should be noted that SHCMs based node-tests are significantly slower than the simpler low-level feature based node-tests due to the more complex computations.

5.4.4 Full System with CRF

Here we give results for the full system that employs a CRF using the unary potentials obtained with the Random Forest classifier on a combination of low-level features[3] as described in the previous section.

CRF Model for multi-label image segmentation. The energy we use for our segmentation task follows the basic formulation of (2.7). We use unary potentials ψ that are based on the output of the respective classification algorithm, *i.e.* the object-class posterior probabilities. In addition we use a colour term π to define $E_{data}(\mathbf{c}, \mathbf{I}) = \sum_i (\psi_i(c_i, \mathbf{I}) + \pi(c_i, \mathbf{I}))$. We compute π based on a Gaussian mixture model estimated from labels inferred on a *per* image basis, using (5.4) with $\pi = 0$. The colour model is included to capture the colour distribution peculiar to that image in addition to the colours captured by the respective classifier, that was trained on all training images and is already included in the posteriors/unary term ψ. The last term in the energy function is the contrast sensitive Potts model ϕ defining E_{smooth} [Boykov and Jolly (2001)]. It depends on the d dimensional colour difference vector $\mathbf{g}_{ij} \in \mathbb{R}^d$, between pixel i and j. Then,

[3]HOG, RGB, and F17 on the MSRC dataset and HOG, RGB, and SHCMs on the VOC2007 dataset.

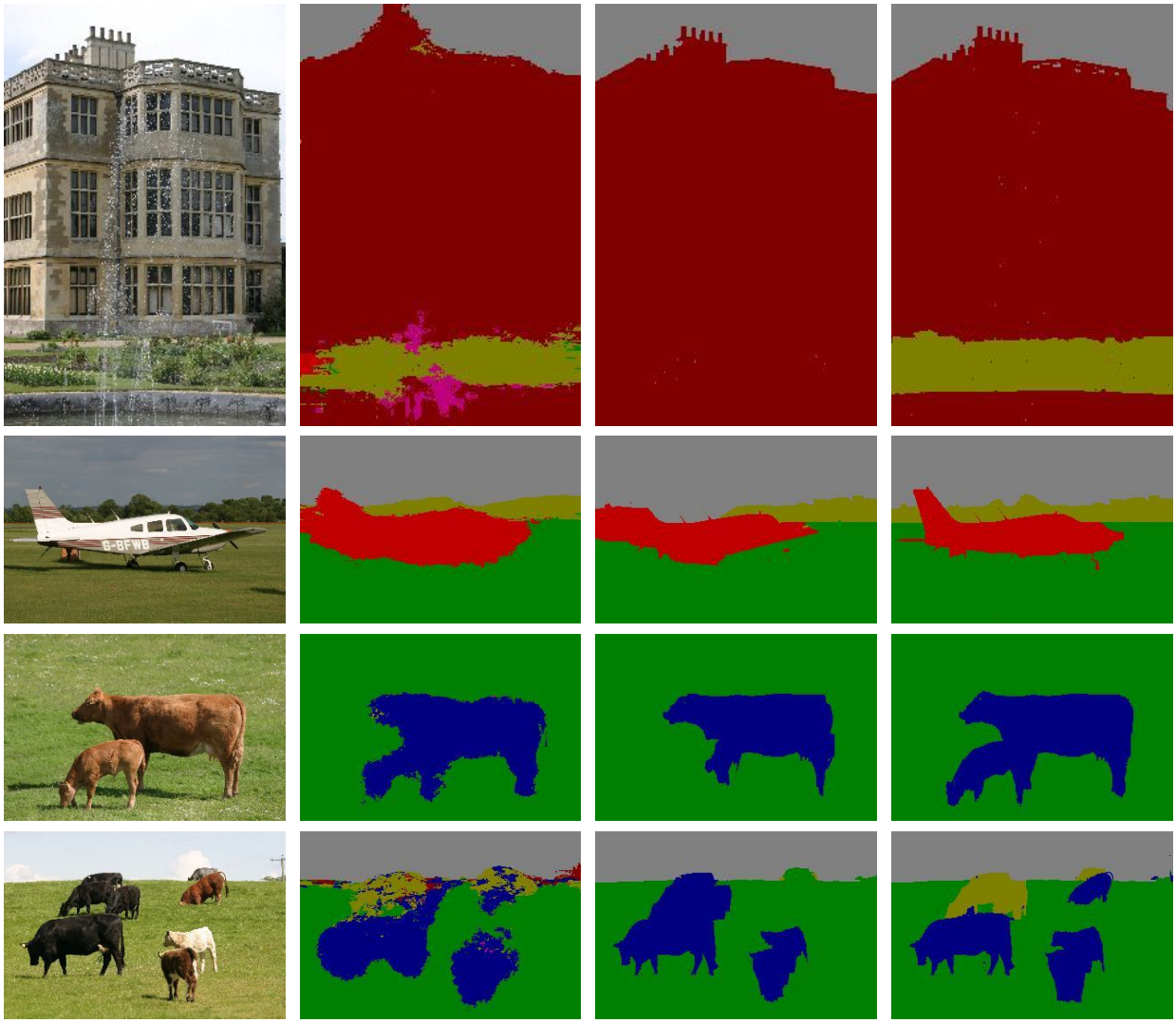


image	MAP posterior	CRF w/o colour	CRF with colour
9-class P_{pix}	86.0%	**87.3**%	85.1%

Figure 5.8: The CRF helps to retrieve crisper object boundaries. Using an additional per image colour model improves "good" classifications further, but can hurt certain cases, *e.g.* last row. (Note that CRF with colour is *not* the second iteration after CRF w/o colour, as different parameters determined with cross-validation are used)

$$-\log P(\mathbf{c}|\mathbf{I}) = \sum_i \left(\psi_i(c_i, \mathbf{I}) + \pi(c_i, \mathbf{I})\right) + \sum_{ij \in I} \phi(c_i, c_j, \mathbf{g}_{ij}) \tag{5.4}$$

describes the CRF energy that we minimise using TRW-S to retrieve the final segmentation, with c_i being the label for pixel I_i.

This CRF model is used with and without the per image colour model adaption, *i.e.* stopping after the first iteration or using a second iteration together with the estimate for π computed based on the labelling that was returned by the first iteration. We use TRW [Kolmogorov (2005)] to minimise the energy in (5.4) and use the class posteriors from the Random Forest as unary

potentials ψ. The results in Figure 5.8 show that the CRF improves the performance slightly, but more importantly improves the visual appearance of the segmentation. Using this additional colour model improves 'good' segmentations further see Figure 5.8 (aeroplane, building), but can deteriorate 'bad' classifications (black cow in the last row). In general the results are very sensitive to the CRF parameters.

The performance on the 9-classes (87.3% without colour model; 85.1% with colour model) is shown in Figure 5.8. For the 21-classes we achieve 73.7% without colour model and 71.6% with the colour model. These results exceed state of the art performance or perform equally well [Shotton *et al.* (2006, 2008), Verbeek and Triggs (2008)]. On the VOC2007 data we achieve 24.4% average class performance, which outperforms the *raw* results by Shotton *et al.* (2008) slightly. Figure 5.9 and 5.10 show example images for the 21-class MSRC and the VOC2007 datasets, together with the per class pixelwise classification accuracies in percent. Again, the colour model improves good segmentations further and can deteriorate bad ones as can be seen in Figure 5.8, 5.9, and 5.10, where we also report the per class accuracies for the 21-class MSRC and the VOC2007 datasets.

Both Shotton *et al.* (2008) and Csurka and Perronnin (2008) use additional higher level classification to improve the performance. This can be achieved by first classifying the whole image, *e.g.* using the BOW model, and based on this classification a prior can be assigned to each object class. This prior is then incorporated into the posteriors returned by the decision-tree. It is important to retrieve priors that perform sufficiently well, but assuming those the segmentation performance can be increased dramatically. Csurka and Perronnin (2008) report 40% using image level priors and Shotton *et al.* (2008) report 42% with detection level priors from the TKK system[4]. This way, Shotton *et al.* improve their raw results from 20% to 42%. Note that our raw result is 22%. These observations suggest that the use of more global context is necessary for very good object segmentation, especially in the highly cluttered images of the VOC datasets. The segmentation task of the VOC2008 is won with a method based on Csurka and Perronnin (2008), again using image level priors.

[4]TKK is the winner of the segmentation challenge [Everingham *et al.* (2007)]. The only "real" entry into this challenge was by Brooks who achieved 8.5%. The TKK score stems from an object detection algorithm where a bounding box was treated as the segmentation for the corresponding object.

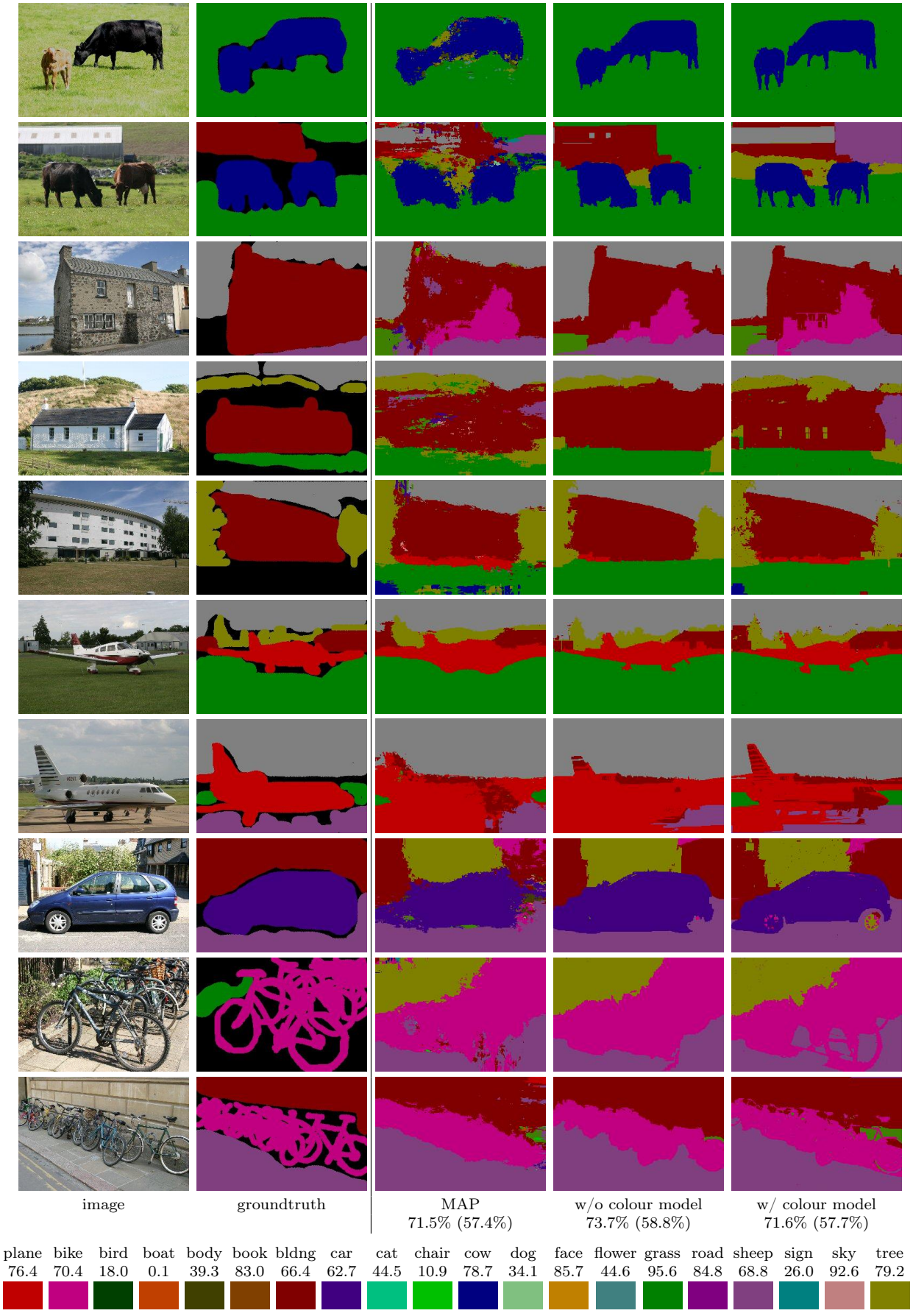


Figure 5.9: 21-class MSRC results: Example images provide an idea of the quality of the segmentations. These images are obtained using the CRF with and without the colour-model together with the unary potentials from the Random Forest with HOG,RGB,F17 as described in Table 5.2 on page 107. MAP denotes the classification according to the unary potentials only. Both P_{pix} and (P_{avg}) are reported.

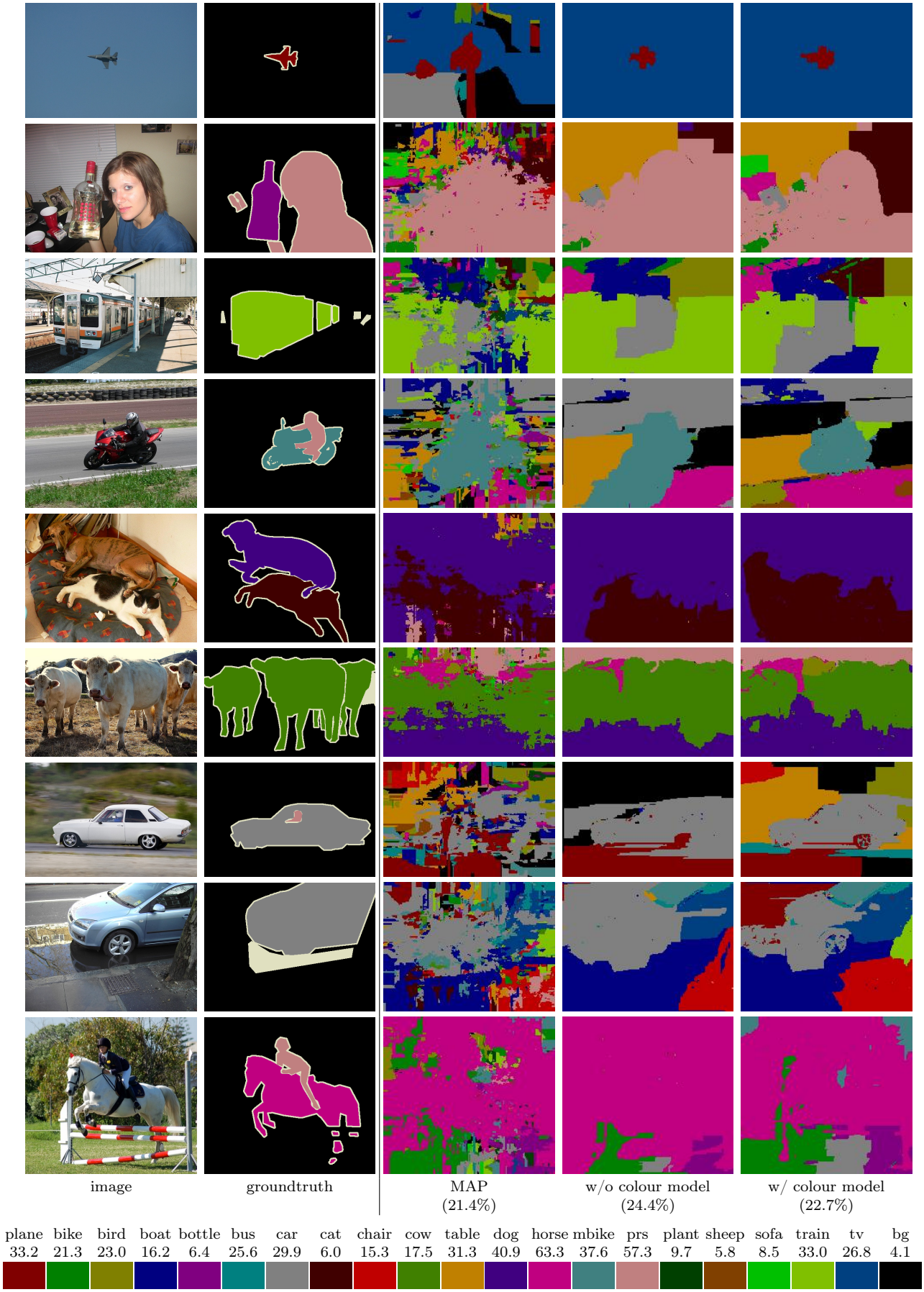


Figure 5.10: VOC2007 results: Example images provide an idea of the quality of the segmentations. These images are obtained using the CRF with and without the colour-model together with the unary potentials from the Random Forest with HOG,RGB,SHCMs as described in Table 5.2 on page 107. MAP denotes the classification according to the unary potentials only. (P_{avg}) is reported.

5.5 Conclusion

This chapter builds on top of Chapter 4 and combines the single-histogram class models (SHCMs) introduced there with the capabilities of discriminative learning provided by the Random Forest classifier. Using *a priori* knowledge in form of SHCMs to compute a weighted sum of textons occurring in a rectangular region near the test-pixel provides strong initial node-tests. A set of experiments showed that the Random Forest can select appropriate feature kernels (rectangle shapes, sizes and positions) to improve the sliding-window based classifier from the previous chapter. Instead of SHCMs we also investigated the use of exemplar histograms directly. Although the results are not reported here, in addition to massively increased computational complexity the results did not show a significant improvement.

We then relate these globally learnt class models to various global, local and context-rich low-level feature types. The feature kernels used in other work Bosch *et al.* (2007b), Lepetit and Fua (2006), Shotton *et al.* (2008, 2006) are interpreted as special cases of the general family introduced here. Our thorough experimental evaluation shows that the complete system performs similar to state of the art or better.

Chapter 6

Harvesting Images from the Web

The Internet provides vast numbers of images often together with some meta information describing the image content. Common sources are web-pages where the text describes the images contained on the web-page. Flickr also provides millions of images together with user defined tags that describe the image content. Even though there are many images available on the net, it is not straight forward to retrieve a set of images belonging to one specific object category.

Our objective in this chapter is to harvest a large number of images of a particular class *automatically*, and to achieve this with high precision. Our motivation is to provide training datasets so that a new object model can be learnt effortlessly. The idea is to use the supervision provided by the content of web-pages to learn visual object-class models. Following Berg and Forsyth (2006) we download all web-pages including their images that are returned for a specific query, *e.g.* penguin. After describing the exact procedure (summarised in Figure 6.1) as well as reporting statistics about the images retrieved, we discuss the advantages of an intermediate step to remove drawings and sketches in Section 6.3. In Section 6.4 it is shown that meta-data and text attributes on the web page containing an image provide a useful estimate of the probability that the image is in-class, and thence can be used to successfully rank images in the downloaded pool. In Section 6.5 we show that this probability is sufficient to provide (noisy) training data for a visual classifier, and that this classifier delivers a superior re-ranking to that produced by text alone. In Section 6.7 we give a thorough comparison to the work of Berg and Forsyth (2006) and show that our automatic method achieves superior ranking results. It is further shown that our visual object-class models can successfully re-rank images returned by

1. Download images and meta-data for new class (*e.g.* "lion") using `WebSearch` &`GoogleImages` (Section 6.2).
2. Filter images: remove drawings&symbolic images (Section 6.3).
3. Rank images based on text-attributes using the Bayes classifier (Section 6.4).
4. Train the visual SVM classifier on text-ranked images (Section 6.5).
5. Rank all images retrieved in step 1. (or step 2.) using the visual classifier.

Figure 6.1: Overview of the text+vision (t+v) image harvesting algorithm.

Google Image Search to significantly improve the precision at low recall.

Before presenting our own method, we give an overview of related work. Recently there have been a few major contributions to solve this or related problems, as the problem of learning visual object-class models without the need of tedious manual labelling is very important and the web provides an ideal source to gather such information cheaply.

6.1 Related Work

Now an introduction of work that uses images from the Internet together with available descriptions such as user defined tags or the context on web-pages for vision based image ranking or learning is given.

One of the earliest attempt on this problem is the work of Lin *et al.* (2003). It addressed the problem of re-ranking images returned by an image search engine. Their approach was solely based on text and context information and did not learn a visual model of any form. Global context is incorporated by using a probabilistic model that assigns a relevance to web-pages that link to specific images. Later on Fergus *et al.* (2004, 2005a) learn visual models to re-rank images returned from a web image search engine. Fergus *et al.* (2004) investigate the possibilities of unsupervised or semi-supervised learning of models, based on images returned by Google image search. The authors mainly use their approach to re-rank the returned images, using a model that was learnt from the initial set of retrieved images. The main assumption states that the returned images reflect some consistency which depends on the query (object-class). For example, searching for penguins results in a set of images which contains many

penguins. Even though this set still includes junk images a model for penguins can be learnt using a robust method. Fergus *et al.* (2005a) learn a topic model from the dataset. It assigns a topic distribution to each image and allows the authors to retrieve a ranked list of all images for each topic, *i.e.* each list ranks the images based on their probability of containing that topic. The problem of selecting the right topic, *i.e.* the penguin topic in this example, is solved by using an additional set of images with a higher proportion of in-class images. This validation set is retrieved by downloading the top five images returned by Google Image Search for a query translated into six languages, thus leading to up to 30 training images. The topic that best explains this validation set is selected and used for the final ranking. The method in Fergus *et al.* (2005a) involved visual clustering of the images by using probabilistic Latent Semantic Analysis (pLSA) [Hofmann (2001)] over a visual vocabulary. Li *et al.* (2007) use a Hierarchical Dirichlet Process instead of pLSA, and Vijayanarasimhan and Grauman (2008) use multiple instance learning to learn the visual models. However, for all these methods the yield is limited to about 1000 images by common image search engines. Another related interactive approach is presented by Collins *et al.* (2008).

Berg and Forsyth (2006) overcome the download restriction by starting from a *web* search instead of an *image* search. This search can generate thousands of images. Their method then proceeds in two stages: first, topics are discovered based on words occurring on the web pages using Latent Dirichlet Allocation (LDA) [Blei *et al.* (2003)] on *text* only. Image clusters for each topic are formed by selecting images where nearby text is top ranked by the latent topic. Following this, several relevant clusters are manually selected and a model based on text, colour, shape and texture is learnt. This model is used to re-rank the downloaded images, thereby creating datasets which contain about 81% of relevant images in the top returned 500 images. Note, the user labelling of clusters avoids the problem of polysemy, as well as providing good training data for the visual classifier. The method succeeds in achieving a greater yield, but at the cost of manual intervention.

Others have used text and images together, however in a slightly different setting. For example, Barnard *et al.* (2003) use groundtruth annotated images as opposed to noisy annotation stemming from web pages, as in our case. Other work of Berg *et al.* (2004) uses text from the Internet, but focused on identifying a specific class rather than general object-classes.

Morsillo *et al.* (2008) present a generative model that combines generative and discriminative components and uses minimal user interaction to learn this combined model. Wang and Forsyth (2008) also combine text and vision and focus on merging the text- and vision-ranking into a combined ranking. They use Wikipedia to train the text-model and Caltech and Flickr images to train the visual model. As will be seen in Section 6.6 and in Section 6.7 one advantage of our system is that once the visual model is learnt we do not rely on the text ranking anymore, and are thus able to compare our performance to previous methods such as Berg and Forsyth (2006) where only the image datasets are provided.

6.2 The Datasets

This section describes the methods for downloading the initial pool of images together with associated meta-data from the Internet. For the purposes of training classifiers and for assessing precision and recall the downloaded images are annotated manually for 18 classes: airplane (ap), beaver (bv), bikes (bk), boat (bt), camel (cm), car (cr), dolphin (dp), elephant (ep), giraffe (gf), guitar (gr), horse (hs), kangaroo (kg), motorbikes (mb), penguin (pg), shark (sk), tiger (tr), wristwatch (ww), zebra (zb).

6.2.1 Data Collection

We compare three different approaches to downloading images from the web. The first approach, named `WebSearch`, submits a *single* query word to Google web search and all images that are linked within the returned web pages are downloaded. The query could also consist of more specific descriptions such as "penguin animal" or "penguin OR penguins", but we focus on single word queries for the experiments in this chapter. Google limits the number of returned web pages to 1000, but many of the web pages contain multiple images, so in this manner thousands of images are obtained. The second approach, `ImageSearch`, starts from Google image search (rather than web search). Google Image Search limits the number of returned images to 1000, but here each of the returned images is treated as a "seed" – further images are downloaded from the web page from where the seed image originated. The third approach, `GoogleImages`, includes only the images directly returned by Google image search (a subset of those returned by `ImageSearch`). Images smaller than 120×120 and exact duplicates are discarded. In

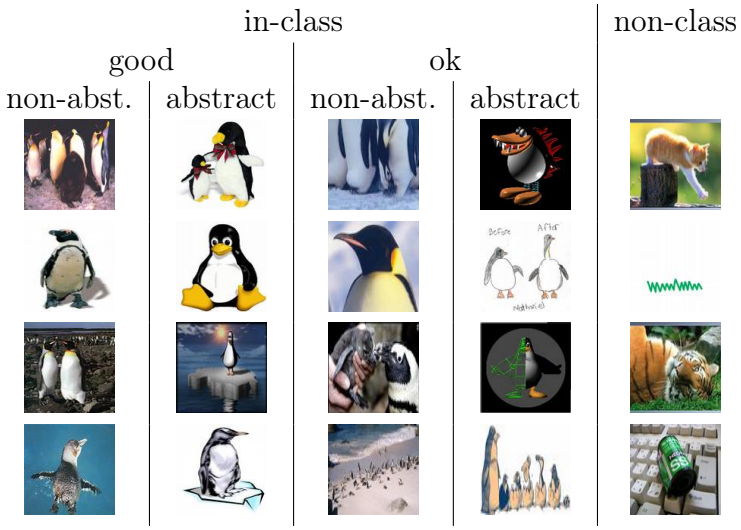


Figure 6.2: Image annotations. Example images corresponding to annotation categories for the class penguin.

addition to the images, text surrounding the image HTML tag and several textual attributes are retrieved:

context10: 10 words (no HTML-tags) before and after the image tag.

contextR: 11th to 50th words (no HTML-tags) before and after the image tag.

filename: filename part of the image URL.

filedir: directory part of the image URL.

websitetitle: text contained in the HTML title-tag.

imagetitle: text in the title attribute of the image-tag.

imagealt: text in the alt attribute of the image-tag.

Empirical tests revealed that using the plural version or more specific terms such as "king penguin" lead to different, possibly better images. This could be used to retrieve even more images and using WordNet[1] or similar services it can also be automated. Initial experiments on Flickr revealed that the available tags are very noisy, which is why we focus on web-pages.

6.2.2 Groundtruth Annotation

Even though the goal is to automatically build image datasets, groundtruth is needed for the evaluation of our method. We also need groundtruth for a few object-class queries for the training of our class independent text ranker in Section 6.4. Currently the most reliable and simplest approach is to manually label the images. There are possibilities to exploit indirect user feedback, *e.g.* from people clicking on images they receive for one of their image search queries (*e.g.* on Google). However, we do not have such data available and therefore manually assign image labels using a web interface. In a similar manner to Fergus *et al.* (2005a), images are divided into three categories:

in-class-good: Images that contain one or many class instances in a clearly visible way (without major occlusion, lighting deterioration or background clutter and of sufficient size).

in-class-ok: Images that show parts of a class instance, or obfuscated views of the object due to lighting, clutter, occlusion and the like.

non-class: Images not belonging to *in-class*.

The good and ok sets are further divided into two subclasses:

abstract: Images that do not look like realistic natural images (*e.g.* drawings, non realistic paintings, comics, casts or statues).

non-abstract: Images not belonging to the previous class.

Example annotations for the class penguin are shown in Figure 6.2. As is usual in annotation there are ambiguous cases, *e.g.* deciding when occlusion is sufficiently severe to classify as *ok* rather than *good*, or when the objects are too small. Note, the *abstract* vs. *non-abstract* categorisation is not general but is suitable for the object-classes we consider in this chapter. For example, it would not be useful if the class of interest was "graph" or "statue" or a similar more abstract category.

Table 6.1 on page 121 lists the 18 categories downloaded and the corresponding statistics for *in-class* and *non-class* images. The overall precision of the images downloaded for all 18 classes is about 29%. Table 6.2 on page 121 details the statistics for each of the three retrieval

[1] http://wordnet.princeton.edu

techniques (`WebSearch`, `ImageSearch` and `GoogleImages`). Note that some images are common between the methods. `ImageSearch` gives a very low precision (only about 4%) and is not used for the harvesting experiments. Only `WebSearch` and `GoogleImages` are used, and their images are merged into one dataset per object-class.

Due to the great diversity of images available on the Internet and because of how we retrieve the images, it is difficult to make general observations on how these datasets look. However, it is clear that polysemy affects the returned images. Interestingly, this is not a problem that could be predicted directly from the English word, since most of the classes we search for do not have direct polysemous meanings, *i.e.* they are not polysemous in the sense of "bank" (as in place to get money, or river bank) for example. It is rather that the words correspond to brands or product names ("leopard tank") or team names (the NHL ice hockey team "San Jose Sharks") or are used as attributes ("tiger shark"). Apart from that, the in-class images occur in almost all variations imaginable, as sharks crashed into houses or other oddities. Even though context [Torralba (2003)] can clearly be important in re-ranking the images (*e.g.* camel and kangaroo in desert-like images), it will have its limitations due to the variety of occurrences of the object.

6.3 Filtering Drawings & Abstract Images

Since we are mostly interested in building datasets for natural image recognition, we ideally would like to remove all *abstract* images from the downloaded images. However, separating *abstract* images from all others automatically is very challenging for classifiers based on visual features. Instead we tackle the easier visual task of removing drawings & symbolic images. These include: comics, graphs, plots, maps, charts, drawings and sketches, where the images can be fairly simply characterised by visual features (see below). Example images are shown in Figure 6.3 on page 122. Their removal significantly reduces the number of *non-class* images improving the resulting precision of the object-class datasets as shown in Table 6.1 (overall precision increases from 29% to 35%). Filtering out such images also aims to remove this type of *abstract* image from the in-class images, as the focus is to build datasets of natural images for the specified object classes.

	downloaded images			after drawing&symbolic filtering			
Class	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	false pos.
airplane (ap)	904	1659	35.27%	635	1007	38.67%	91
beaver (bv)	236	3121	7.03%	160	2195	6.79%	4
bikes (bk)	1268	1931	39.64%	983	1082	47.60%	111
boat (bt)	856	2175	28.24%	726	1354	34.90%	70
camel (cm)	594	1808	24.73%	485	1274	27.57%	46
car (cr)	1128	1042	51.98%	938	568	62.28%	92
dolphin (dp)	791	1416	35.84%	533	906	37.04%	81
elephant (ep)	937	1558	37.56%	763	1007	43.11%	11
giraffe (gf)	945	1267	42.72%	802	763	51.25%	32
guitar (gr)	1219	2035	37.46%	873	832	51.20%	248
horse (hs)	1229	1720	41.68%	975	1043	48.32%	78
kangaroo (kg)	418	1763	19.17%	329	1161	22.08%	14
motorbikes (mb)	732	953	43.44%	607	582	51.05%	86
penguin (pg)	748	1400	34.82%	447	794	36.02%	33
shark (sk)	583	1710	25.43%	413	1089	27.50%	60
tiger (tr)	379	2068	15.49%	311	1274	19.62%	17
wristwatch (ww)	941	957	49.58%	710	549	56.39%	220
zebra (zb)	483	1662	22.52%	416	987	29.65%	19
total	14391	30245	32.24%	11106	18467	37.55%	1313

Table 6.1: Image class statistics of the original downloaded images using `WebSearch`&`GoogleImages` only, and after applying the drawing&symbolic images removal filter. The last column shows the number of relevant non-drawing images that were removed (false positives).

Service	in-class	non-class	precision
`WebSearch`	8773	25252	26%
`ImageSearch`	5963	135432	4%
`GoogleImages`	4416	6766	39%

Table 6.2: Statistics by source. The statistics of downloaded images for the different retrieval techniques. `WebSearch` describes the images downloaded from all web-pages that Google returns to a query word. `GoogleImages` returns all images returned by Google Image Search. `ImageSearch` uses the images returned by Google Image Search as "seed" images and returns all images on the web-pages where those "seed" images originate from.

6.3.1 Learning the Filter

We train a radial basis function Support Vector Machine (SVM) on a hand labelled dataset (examples in Figure 6.3). After the initial training no further user interaction is required. In order to obtain this dataset, images were downloaded using `ImageSearch` with one level of recursion (*i.e.* web pages linked from "seed" web pages are also used) with queries such as "sketch" or "drawing" or "draft". The goal was to retrieve many images and then select suitable training images manually. The resulting dataset consists of approximately 1400 drawings&symbolic im-

Figure 6.3: Drawings&symbolic images. Examples of positive and negative training images.

ages, and 2000 non drawings&symbolic images. We used this relatively large number of images to be sure to capture all varieties of abstract images as well as the wide range of natural images that can be encountered.

Three simple visual only features are used: (i) a colour-histogram; (ii) a histogram of the L2-norm of the gradient; (iii) a histogram of the angles $(0 \ldots \pi)$ weighted by the L2-norm of the corresponding gradient. In all cases 1000 equally spaced bins are used. Although we did not make a comprehensive evaluation of various methods and histogram-sizes smaller and larger binning was tried and did not work as well. The motivation behind this choice of features is that drawings&symbolic images are characterised by sharp edges in certain orientations and or a distinctive colour distribution (*e.g.* only few colours in large areas). The method achieves around 90% classification accuracy on the drawings&symbolic images dataset (using two-fold cross-validation).

This classifier is applied to the entire downloaded image dataset to filter out drawing&symbolic images, before further processing. The total number of images that are removed for each class is shown in Table 6.1. In total 39% of *non-class* images are removed over all classes. The remaining images are those used in our experiments. As well as successfully removing *non-class* images, the filter also succeeds in removing an average of 54% (107) *in-class abstract* images, with a range between 38% (for giraffe, 111 images) and 75% (for beaver, 72 images). There is some loss of the desired *in-class non-abstract* images, with on average 11% (73 images) removed, though particular classes lose a relatively high percentage (24% for guitar and wristwatch). Even though this seems to be a high loss the precision of the resulting datasets is improved in all cases except for the class beaver.

6.4 Ranking on Textual Features

We now describe the re-ranking of the returned images based on text and meta-data alone. Here we follow and extend the method proposed by Frankel *et al.* (1997) in using a set of textual attributes whose presence is a strong indication of the image content.

Textual features. We use seven features from the text and HTML-tags on the web-page: *contextR, context10, filedir, filename, imagealt, imagetitle, websitetitle*, as introduced in Section 6.2.1. The features are intended to be conditionally independent, given the image content (we address this independence below). It is difficult to compare directly with the features in Frankel *et al.* (1997), since no precise definition of the features actually used is given.

Context here is defined by the HTML source, not by the rendered page since the latter depends on screen resolution and browser type and is expensive to compute. In the text processing a standard stop list [Onix] and the Porter stemmer [Porter *et al.* (2002)] are used. *Stemming* is a technique that reduces words to a "stem" which is independent of plural, singular and other inflections. *Stopping* describes the method where common words as "a", "I", "and" and the like are discarded, under the assumption that they do not contribute to the semantics of the text. In addition HTML-tags and domain specific stop words (such as "`html`" or "` `") are ignored.

We also experimented with a number of other features, such as the image MIME type (`gif`, `jpeg`, *etc.*), but found that they did not help discrimination. Table 6.3 gives an idea of the occurrence of the query word in any of the text features. It can be seen that the occurrence of the query word in the filename- or imagetitle-tag is a strong indicator for the image being in-class.

6.4.1 Image Ranking

Using these seven textual features, the goal is to re-rank the retrieved images. Each feature is treated as binary: "True" if it contains the query word (*e.g.* penguin) and "False" otherwise. Thus, the seven features define a binary feature vector for each image $\mathbf{a} = (a_1, \ldots, a_7)$. We learn a class *independent* ranker in order to re-rank the images based on those seven features; *i.e.* the ranker is not learnt or tuned for each class separately, but is learnt once and can then

	airplane			beaver			bikes		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	263	407	39.25%	107	1312	7.54%	717	717	50.00%
context10	264	349	43.07%	97	1025	8.65%	580	523	52.58%
filedir	85	104	44.97%	29	1093	2.58%	440	496	47.01%
filename	186	289	39.16%	128	420	23.36%	249	153	61.94%
imagealt	130	148	46.76%	76	246	23.60%	139	92	60.17%
imagetitle	17	26	39.53%	9	19	32.14%	20	18	52.63%
websitetitle	0	0	NaN%	63	372	14.48%	0	0	NaN%

	boat			camel			car		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	458	819	35.87%	326	786	29.32%	679	401	62.87%
context10	354	593	37.38%	296	547	35.11%	452	268	62.78%
filedir	315	480	39.62%	77	208	27.02%	408	247	62.29%
filename	279	293	48.78%	320	442	41.99%	187	121	60.71%
imagealt	123	151	44.89%	162	209	43.67%	107	62	63.31%
imagetitle	3	8	27.27%	11	12	47.83%	11	4	73.33%
websitetitle	154	188	45.03%	117	246	32.23%	0	0	NaN%

	dolphin			elephant			giraffe		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	449	656	40.63%	503	604	45.44%	467	412	53.13%
context10	367	517	41.52%	464	457	50.38%	455	337	57.45%
filedir	276	496	35.75%	161	309	34.26%	119	110	51.97%
filename	259	152	63.02%	464	290	61.54%	576	193	74.90%
imagealt	166	117	58.66%	237	191	55.37%	292	131	69.03%
imagetitle	8	7	53.33%	24	15	61.54%	25	15	62.50%
websitetitle	120	190	38.71%	242	228	51.49%	248	181	57.81%

	guitar			horse			kangaroo		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	625	533	53.97%	660	735	47.31%	173	650	21.02%
context10	502	368	57.70%	506	553	47.78%	175	565	23.65%
filedir	406	384	51.39%	444	409	52.05%	53	265	16.67%
filename	280	133	67.80%	339	253	57.26%	199	326	37.90%
imagealt	140	98	58.82%	150	111	57.47%	117	269	30.31%
imagetitle	13	5	72.22%	10	11	47.62%	24	16	60.00%
websitetitle	0	0	NaN%	203	199	50.50%	104	279	27.15%

	motorbikes			penguin			shark		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	231	246	48.43%	269	424	38.82%	259	656	28.31%
context10	242	203	54.38%	233	300	43.71%	228	530	30.08%
filedir	57	28	67.06%	85	267	24.15%	144	327	30.57%
filename	248	96	72.09%	277	191	59.19%	250	380	39.68%
imagealt	137	92	59.83%	163	103	61.28%	145	204	41.55%
imagetitle	16	10	61.54%	13	2	86.67%	11	21	34.38%
websitetitle	0	0	NaN%	123	144	46.07%	129	274	32.01%

	tiger			wristwatch			zebra		
	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
contextR	188	684	21.56%	263	133	66.41%	264	602	30.48%
context10	166	518	24.27%	241	85	73.93%	275	453	37.77%
filedir	65	292	18.21%	24	8	75.00%	67	280	19.31%
filename	215	378	36.26%	57	14	80.28%	324	387	45.57%
imagealt	97	172	36.06%	97	18	84.35%	154	218	41.40%
imagetitle	4	10	28.57%	11	5	68.75%	13	12	52.00%
websitetitle	61	242	20.13%	0	0	NaN%	138	245	36.03%

Table 6.3: Relation of **image quality** and the occurrence of the query word in textual attributes for the filtered images. Shown are the numbers for non-junk images and junk images (stemming was used). The recall is given in parentheses in percent and the precision (prec.) is given in percent as well.

be applied to any new class. The posterior probability, $P(y = \textit{in-class}|\mathbf{a})$, of the image being *in-class*, where $y \in \{$ *in-class*, *non-class*$\}$ is the class-label of an image, is estimated using five different Bayesian models and logistic regression. The images are ranked based on their posterior probability of being *in-class.*

The following presents the details of those five different Bayesian models for $P(y = \textit{in-class}|\mathbf{a})$. All these models are trained by estimating the corresponding likelihoods, *e.g.* $P(a_3|y)$, as well as the prior $P(y)$ empirically, *i.e.* counting the corresponding occurrences in the training data and using a Dirichlet prior/Laplace smoothing. During testing, these estimates are used in the form of a look-up table to retrieve the final posterior estimate $P(y|\mathbf{a})$.

For simplicity Z denotes the partition function to normalise the following posteriors to proper probability distributions:

Chow-Liu dependence tree decomposition [Chow and Liu (1968)].

$$P(y|\mathbf{a}) = \prod_{1}^{8} P(x_i|x_{m(j)})/Z \tag{6.1}$$

with $x = (a_1, \ldots, a_7, y)$ and m being a permutation of $(1, \ldots, 8)$. The Chow-Liu model approximates the full joint dependency graph as a tree by retaining the edges between variables with the highest mutual information.

Naïve Bayes model.

$$P(y|\mathbf{a}) = \prod_{1}^{7} P(a_i|y) \cdot P(y)/Z \tag{6.2}$$

Given the class-label for an image the text features are assumed to be independent. For our application this is not the case, *e.g.* filename and image alternative tag are highly correlated.

Pairwise dependencies.

$$P(y|\mathbf{a}) = \prod_{i,j=1}^{7} P(a_i, a_j|y) \cdot P(y)/Z \tag{6.3}$$

Only pairwise dependencies are modelled. This is similar to the Chow-Liu model, but less sparse.

Full joint.

$$P(y|\mathbf{a}) = P(a_1, \cdots, a_7|y) \cdot P(y)/Z \tag{6.4}$$

The full joint probability distribution is learnt. If the amount of available training data is too small the learnt model can be inaccurate.

Mixed naïve Bayes.

$$P(y|\mathbf{a}) = P(a_1, \ldots, a_4|y) \prod_{5}^{7} P(a_i|y) \cdot P(y)/Z \tag{6.5}$$

where $P(a_1, \ldots, a_4|y)$ is the joint probability of the first four textual features (*contextR, context10, filedir, filename*). This choice resulted from a comparison of several different factorisations of the likelihood. We looked at the correlation of the attributes and evaluated the performance of different factorisations on a validation set. However, as the experiments in the next section show, the performance variations between these different factorisations do not differ much.

Logistic regression. We also evaluate the performance of *logistic regression* to directly model $P(y = \textit{in-class}|\mathbf{a})$. Thus,

$$P(y|\mathbf{a}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{a}}}$$

is estimated and used directly.

6.4.2 Text Ranking Results

Images are ranked using the text based posterior $P(y|\mathbf{a})$ to provide an ordering. We assess the performance by reporting precision at various points of recall as well as average precision (see Section 3.3.4). To re-rank images for one particular class (*e.g.* penguin) we *do not* employ the groundtruth data for that class. Instead, we train the Bayes classifier or the logistic regression model using all available annotations, *except* the class we want to re-rank. This way we evaluate performance as a *completely automatic class independent* image ranker; *i.e.* for any new and unknown class, the images can be re-ranked without *ever* using labelled groundtruth knowledge of that class.

prec.	ap	bv	bk	bt	cm	cr	dp	ep	gf	gr	hs	kg	mb	pg	sk	tr	ww	zb	avg.
	Mixed Naïve Bayes																		
15%	40.6	30.0	**70.4**	52.2	49.7	61.2	70.3	70.7	82.4	63.2	57.7	53.9	67.9	68.4	**54.1**	43.8	78.0	48.4	59.1
100	39.0	31.0	76.0	52.0	47.0	**66.0**	69.0	**77.0**	**87.0**	58.0	58.0	53.0	67.0	**69.0**	**55.0**	43.0	79.0	50.0	**59.8**
avg.	41.7	31.2	58.7	49.0	48.2	66.5	59.2	**64.6**	**76.1**	65.7	57.2	41.0	**66.6**	61.5	42.6	37.6	70.6	53.0	**55.9**
	Pairs																		
15%	41.5	34.3	67.5	**54.6**	50.0	61.4	68.4	64.9	83.0	62.0	57.7	57.1	66.4	69.2	43.2	43.4	81.1	53.0	58.8
100	**43.0**	**36.0**	66.0	51.0	53.0	61.0	69.0	62.0	79.0	59.0	**61.0**	53.0	66.0	**69.0**	38.0	41.0	81.0	49.0	57.6
avg.	46.6	33.5	60.5	45.2	45.6	65.2	59.8	62.3	74.4	65.2	55.2	43.4	65.6	62.7	39.3	38.3	71.7	53.5	54.9
	Chow-Liu tree																		
15%	38.1	19.4	62.5	53.3	48.0	61.2	61.4	64.9	72.2	**68.7**	55.9	32.9	**69.5**	57.5	41.1	38.3	66.0	40.7	52.9
100	**43.0**	19.0	69.0	55.0	50.0	59.0	63.0	65.0	68.0	**70.0**	58.0	33.0	**72.0**	56.0	41.0	34.0	80.0	33.0	53.8
avg.	41.1	21.1	56.5	45.3	45.5	66.6	57.2	58.4	73.5	61.4	55.3	38.7	66.2	55.5	42.6	36.0	63.4	45.6	51.6
	Naïve Bayes																		
15%	**41.9**	**38.7**	67.5	54.4	50.0	**61.7**	70.3	64.91	81.3	62.0	57.5	57.1	66.4	**70.7**	43.2	41.1	81.1	**58.7**	**59.3**
100	**43.0**	**36.0**	66.0	51.0	**54.0**	61.0	68.0	62.0	79.0	59.0	**61.0**	53.0	66.0	**69.0**	37.0	40.0	81.0	**58.0**	58.0
avg.	**46.7**	**34.2**	60.6	45.2	45.5	65.1	59.9	62.3	74.4	65.5	55.2	**43.5**	65.6	**62.9**	39.8	38.4	71.7	52.1	54.9
	Full joint																		
15%	41.3	32.0	69.8	53.5	47.0	59.8	**71.6**	**72.6**	**85.4**	62.0	58.9	39.7	65.4	67.7	45.5	41.1	79.8	56.0	58.3
100	37.0	32.0	**77.0**	**56.0**	45.0	62.0	**71.0**	74.0	84.0	57.0	55.0	42.0	63.0	66.00	45.0	41.0	82.0	55.0	58.0
avg.	42.2	32.2	**60.6**	**49.3**	49.1	**66.6**	**62.6**	63.2	75.9	65.8	**57.6**	37.6	65.6	62.1	**44.9**	38.6	71.3	**54.1**	55.5
	Logistic Regression																		
15%	41.2	30.4	68.4	53.3	**55.9**	61.4	66.7	62.7	78.5	62.0	**59.7**	**57.8**	65.9	66.3	41.7	**45.1**	**81.8**	51.3	58.3
100	**43.0**	33.0	68.0	**56.0**	**54.0**	63.0	68.0	61.0	79.0	60.0	57.0	**54.0**	64.0	67.0	41.0	**44.0**	**83.0**	47.0	57.9
avg.	46.2	32.4	60.6	48.3	**49.5**	64.7	60.5	61.9	75.0	**66.0**	54.5	43.4	66.0	61.1	39.3	**40.6**	**72.0**	53.8	55.3

Table 6.4: Precision of textual re-ranking. The performance of the textual re-ranking for all 18 classes over different models: precision at 15% recall, precision at 100 images recall, and average precision. The precision is given as a percentage. The last column gives the average over all classes. Mixed naïve Bayes performs best and was picked for all subsequent experiments.

In the evaluation there are two possibilities for how to treat the *abstract* images: (i) *abstract* images are considered to be *in-class*, or (ii) *all abstract* images are treated as *non-class*. The case with *abstract* images treated as *non-class* (a$\not$) leads to poorer performance of the text ranker. This is expected behaviour as it can be imagined that many images might textually be related to the object-class, but of type *abstract*; all those images are considered *non-class* and therefore reduce the precision for a specific recall. We do not report the detailed results here as we choose option (i) for most experiments, *i.e.* abstract images are considered to be in-class.

Figure 6.4 gives an overview of the different methods and their performance. The mixed model (6.5) gave slightly better performance than other factorisations (for the top 100 images), and reflects the fact that the first four features are less independent from each other than the remaining three. It is used to train the visual classifier in the next section. Logistic

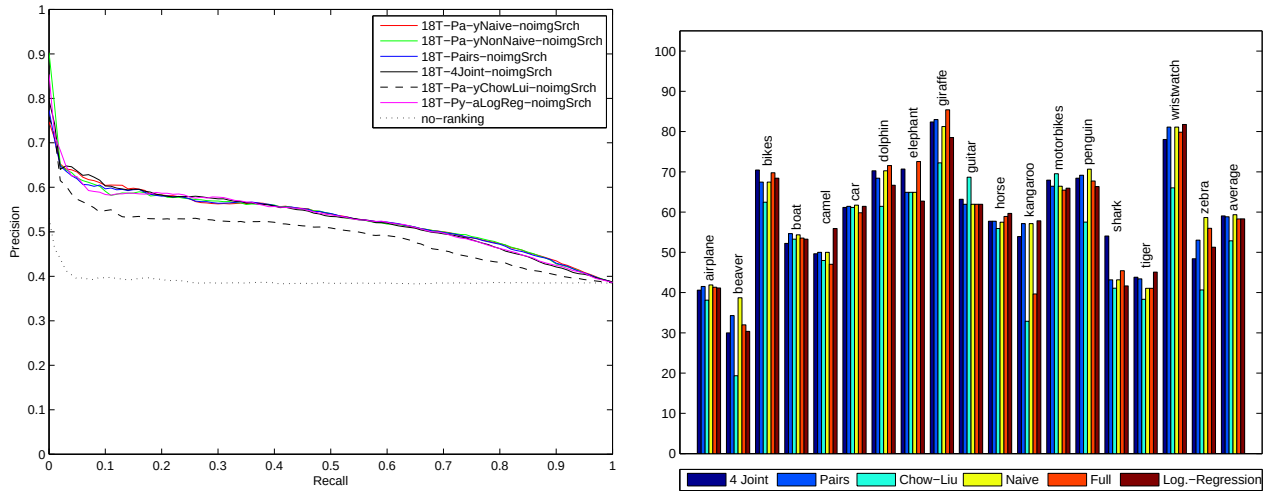


Figure 6.4: Comparison of text rankers. The average precision-recall of text rankers is shown on the left. The precision-recall curve averaged over all 18 classes for each of the described ranking methods. The performance at 15% recall for all 18 classes and the four different models (see bar chart on the right).

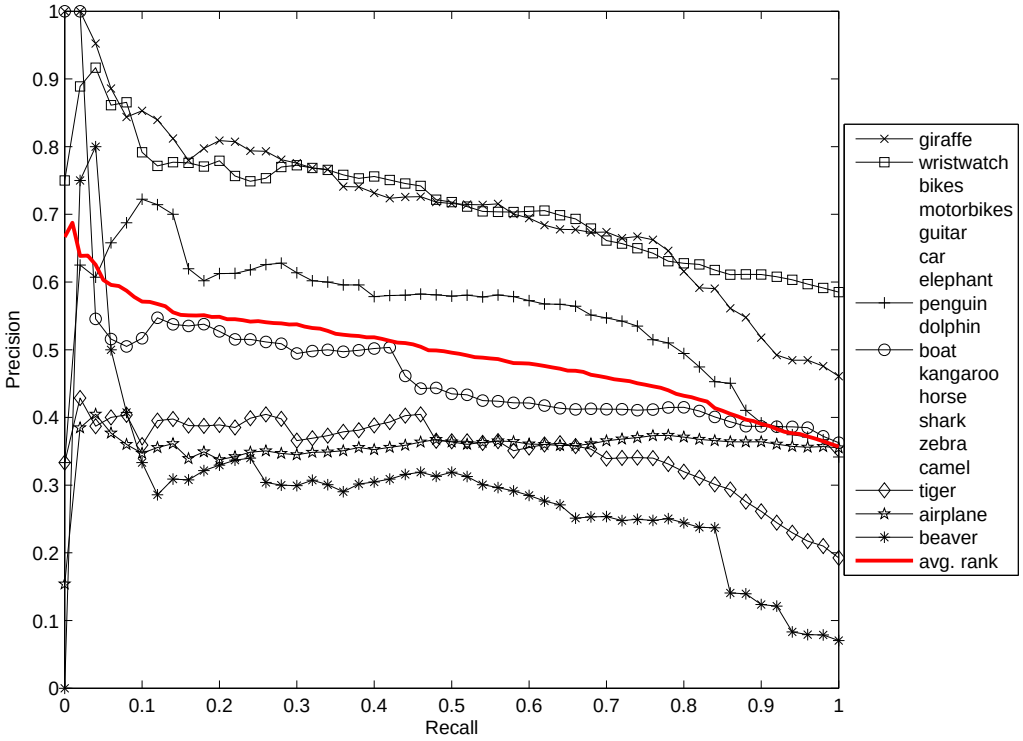


Figure 6.5: Text based re-ranking. precision of recall estimated for each class with *abstract* images considered *in-class* using the mixed naïve Bayes model. The labels are shown in decreasing order of precision at 15% recall. The recall precision curves are only shown for selected classes for clarity. The average over *all* 18 classes is also shown.

regression performed similar to the Bayes models. Only the Chow-Liu decomposition performs significantly worse. Figure 6.5 shows the precision-recall curves for selected classes using the mixed naïve model. It can clearly be seen that precision is highly increased at the lower recall levels, compared to the average precision of Table 6.1 on page 121.

A separate set of experiments was carried out to measure how the performance of the text ranker varies with the number and choice of classes used for training. Given our goal of ranking images we compare the different models by assessing precision at 15% recall. We find that the performance is almost unaltered by the choice of training classes, provided more than five classes (chosen randomly) are used for training.

Discussion As can be seen from Figure 6.5 our text re-ranker performs well on average, and significantly improves the precision up to quite a high recall level (see Figure 6.8 on page 137 to 6.11 for top ranked images). In Section 6.5 we will show that this is sufficient to train a visual classifier. For some classes the text ranker performs very well (*e.g.* wristwatch, giraffe) for others it performs rather poorly (*e.g.* airplane, beaver, camel, tiger). Visual inspection of the highly ranked "outlier" (*non-class*) images in the text ranked lists gives some explanation for these performances. Classes that perform well (wristwatch, giraffe) generally have outliers that are unrelated to each other. In contrast for the classes that perform poorly the outlier images are related and result from lack of discrimination of the query word – for example for airplanes there are images of airplane food, airports, toy airplanes, paper airplanes, airplane interiors, and advertisements with comic airplanes. Other classes suffer from the type of polysemy described in Section 6.2: for camels there are brand and cigarette related outliers; and for tiger there is the attribute problem with images of helicopters, tanks, fish (sharks), boxing, golf, stones, and butterflies. Beaver suffers from both problems – lack of discrimination and polysemy – there are many nature related outliers that do not contain beavers, and some brand related outliers. Despite these problems we are able to train a good visual classifier for most classes in the next section.

We investigated two alternative text-based classifiers, pLSA and SVM, in addition to logistic regression and the Bayes estimator finally adopted, but found they had inferior performance. For the SVM the same binary text features $\mathbf{a}$ were used. It is possible that the binary features lead to the poor performance of the SVM. For the pLSA we used *context10* and *contextR* (similar to Berg and Forsyth (2006)). Due to problems in the pLSA clustering, the problem of how to select the right topic without user interaction as in Berg and Forsyth (2006), and the question of how to use the additional binary features (*e.g.* *filename*) in a principled manner, we adopted the Bayes estimator instead. For a more detailed discussion on the pLSA ranking

also see Section 6.7.2. It is interesting to see that logistic regression, being conceptually very different, performs very similar to most of the Bayes factorisations. This might lead to the conclusion that some sort of saturation point is reached and that those *seven binary* features do not allow for more discrimination.

6.5 Ranking on Visual Features

The text re-ranking associates a posterior probability with each image as to whether it contains the query class or not. The problem we are now faced with is how to use this information to train a visual classifier which would improve the ranking further. The problem is one of training from noisy data: we need to decide which images to use as positive and negative training data and how to select a validation set in order to optimise the parameters of the classifier. We first describe the visual features used, and then how the classifier is trained.

6.5.1 Visual Features

The basics for modelling object-classes using visual features and feature descriptors were introduced in Section 2.1. We follow the approach of Fergus *et al.* (2005a) and use a variety of region detectors with a common visual vocabulary in the bag of visual-words model framework (BOW), as introduced in Section 2.1.2. All images are first re-sized to 300 pixels in width. Regions are detected using: difference of Gaussians, Multiscale-Harris [Mikolajczyk *et al.* (2004)], Kadir's saliency operator [Kadir *et al.* (2004)], and points sampled from Canny edge points. Each image region is represented as a 72 dimensional SIFT descriptor [Lowe (1999)]. A separate vocabulary consisting of 100 visual-words is learnt for each detector using k-means, and these vocabularies are then combined into a single vocabulary of 400 words. Finally, the descriptor of each region is assigned to the vocabulary.

In addition to the SIFT descriptor used in Fergus *et al.* (2005a) we add the widely used HOG descriptor [Dalal and Triggs (2005)]. It is computed over the whole image to incorporate some spatial layout into the model. Fritz and Schiele (2008) used it in a setting similar to ours. We use a cellsize of 8 pixels, a blocksize of one cell, and 9 contrast invariant gradient bins. This results in a 900 dimensional feature vector, as the images where re-sized to 80×80 pixels. The two descriptors are concatenated resulting in a 1300 dimensional feature vector per image.

6.5.2 Training the Visual Classifier

Due to the amount of noise in the training data we chose to use a SVM classifier, since it has the potential to train despite noise in the data. The following presents our setup in detail.

At this point we can select n_+ positive training images from the top of the text ranked list, or those that have a posterior probability above some threshold. A subset of these positive images will be "noisy", *i.e.* will not be *in-class*. Figure 6.4 on page 128 gives an idea of the noise stemming from the proportion of outliers. It averages at 40%, if $n_+ = 100$ (see Table 6.4 on page 127). However, we can assume that the *non-class* images in this positive set are *not* visually consistent – an assumption verified to some extent by the results in Section 6.6. The case of negative images is more favourable: we select n_- images at random from all downloaded images (*i.e.* from all 18 classes, tens of thousands of images) and the chances of any image being of a particular class is very low. We did not choose to select the n_- images from the low ranked images of the text ranker output, because the probability of finding *in-class* images there is higher than finding them in the set of *all* downloaded images. Note, we do not use the groundtruth at any stage of this training.

To implement the SVM we use the publicly available SVM$^{\text{light}}$ software by Joachims, with the option to remove inconsistent training data enabled. See Section 2.2.1 for the details of the parameters that occur in the version we use. Given two input images I_i and I_j and their corresponding normalised histograms of visual-words $\mathbf{x}$ and $\mathbf{x}'$ this implementation uses the following χ^2 radial basis function (RBF) kernel [Zhang *et al.* (2007)]: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \cdot \chi^2(\mathbf{x}, \mathbf{x}'))$, with γ being the free kernel parameter. Thus, γ, C_+ and C_- are the three parameters that can be varied. Sometimes C_+ and C_- are used to correct unbalanced training data [Morik *et al.* (1999)]. In our case, however, the SVM is very sensitive to these parameters, probably due to the huge amount of noise in the data, and the optimal value does not directly correspond to the ratio of positive to negative images. Therefore, we obtain the optimal values for these parameters by means of ten-fold cross validation.

Cross validation. We require a performance measure for the cross-validation and use precision at 15% recall. A grid search over the aforementioned three parameters is performed, training the SVM with each of the 560 parameter settings on 9/10th of the total training data

$(n_+ + n_-)$. The testing is then performed on the remaining 1/10th of the data and the quality of each parameter setting is rated based on the 15% recall on this test-set. This cross validation is carried out for each of the 10 non-overlapping chunks.

6.6 Results for Textual/Visual Image Ranking

In this section we evaluate different combinations of training and testing. Figure 6.8 to Figure 6.11 on page 140 compare text and vision and also show example high ranked images using our text+vision algorithm with BOW+HOG. First we investigate the influence of the parameter choices for fore- and background images. Then the influence of the training data noise is evaluated by running the algorithm on the manually labelled groundtruth data. We also compare to a "baseline" where images are sampled randomly, *i.e.* no text ranking is used. Last, we take a look at the influence of the filter step from Section 6.3 and a variation on the cross-validation step. We report the precision at 15% recall ± one standard deviation over five runs of cross-validation for those experiments in Table 6.5.

Influence of n_+ and n_-. We first investigate how the classification performance is affected by the choice of n_+ and n_-. Results are given in Table 6.5 for various choices of these parameters. The clear improvement brought by the visual classifier over the text based ranking for most classes is obvious. It can be seen that increasing n_- tends to improve performance. It is, however, difficult to select optimal values for n_+ and n_- since these numbers are very class dependent. Table 6.5 indicates that using more images in the background class n_- tends to improve performance but there is no real difference between using 150/1000 and 250/1000, which perform at $68.4\% \pm 1.9$ and $68.0\% \pm 2.0$ and are thus not significantly different. For each choice, five different random selections are made for the sets used in the ten-fold cross-validation, and mean and standard deviation are reported. It can be seen that HOG alone performs significantly worse than the bag of visual-words $57.9\% \pm 1.8$, but the combination of BOW and HOG improves the overall performance to $69.8\% \pm 2.1$, also compared to BOW alone $68.0\% \pm 2.0$.

prec. 15%	ap	bv	bk	bt	cm	cr	dp	ep	gf	gr	hs	kg	mb	pg	sk	tr	ww	zb	avg.	std
text	40.6	30.0	70.4	52.2	49.7	61.2	70.3	70.7	82.4	63.2	57.7	53.9	67.9	68.4	54.1	**43.8**	78.0	48.4	59.1	
t+v (250/250)	54.0	36.4	64.8	60.9	53.5	78.5	61.9	77.5	86.5	65.4	60.6	53.5	78.2	55.5	56.8	45.7	81.6	**96.8**	64.9	9.3 (2.5)
t+v (150/500)	59.	33.	**69.7**	63.	56.8	91.8	61.6	70.7	83.2	**67.3**	67.	50.0	77.4	70.4	69.3	59.6	86.3	85.5	67.9	8.3 (2.0)
t+v (250/500)	58.6	35.0	64.6	**65.1**	47.8	79.0	60.1	75.4	86.9	51.7	68.4	50.2	77.7	69.7	64.3	39.8	90.4	87.9	65.2	8.8 (2.0)
t+v (250/1000)	**63.5**	32.3	65.9	62.4	51.6	93.0	61.7	80.2	87.8	62.6	**71.2**	45.5	**84.6**	69.6	64.9	53.0	86.6	95.8	68.4	7.9 (1.9)
t+v (150/1000)	52.3	39.3	68.6	**66.2**	57.3	87.9	66.5	78.7	83.8	63.0	57.3	48.6	64.0	72.5	**82.9**	**62.6**	**93.2**	79.8	68.0	7.2 (2.0)
t+v (150/1000) C	49.8	42.7	**71.0**	**66.2**	55.1	88.8	64.7	78.0	85.7	62.3	49.2	**56.2**	67.0	72.6	69.8	58.5	88.9	86.1	67.4	8.3 (2.1)
HOG t+v	45.1	21.1	53.2	62.0	49.1	80.4	51.6	71.6	78.7	62.9	64.1	43.0	68.2	77.8	47.7	29.8	87.6	48.0	57.9	6.6 (1.8)
HOG+BOW t+v	42.8	**46.4**	70.4	60.9	54.5	**93.5**	67.7	**84.8**	88.3	62.5	68.8	55.2	72.1	78.9	80.7	57.3	89.7	81.4	69.8	8.1 (2.1)
HOG+BOW t+v C	51.3	42.3	68.2	60.3	**63.2**	91.1	**69.7**	78.7	**88.7**	66.3	70.1	53.9	76.6	**90.2**	66.1	50.2	92.5	91.7	**70.6**	7.2 (1.7)
gt (150/1000)	83.1	90.8	75.8	76.1	78.0	98.6	78.2	96.0	91.4	88.8	90.2	69.0	95.5	82.7	91.8	94.3	96.1	93.3	87.2	8.4 (2.0)
(B) (150/1000)	52.4	12.9	55.4	63.6	54.1	94.9	42.6	47.9	83.7	61.4	52.8	29.8	65.2	53.7	42.9	28.5	82.7	78.3	55.7	6.7 (1.7)

Table 6.5: Comparison of precision at 15% recall. 'text' refers to text re-ranking alone; 't+v' is text+vision re-ranking using different training ratios n_+/n_-; 'gt' is groundtruth (only positive images) training of the visual classifier; and (B) is the baseline, where the visual classifier is trained on $n_+ = 150$ images uniformly sampled from the filtered images of one class, instead of the text re-ranked images, and $n_- = 1000$ background images. The second last column (avg.) gives the average over all classes. The last column states the mean of the classwise standard deviations over five runs of cross-validation, as well as the standard deviation of the means over all classes, in parentheses.

Influence of training data noise. We next determine how much the performance is affected by the noise in the training data by training the SVM on groundtruth positive data, *i.e.* instead of selecting n_+ images from the text ranked images (HOG+BOW), we select n_+ in-class images using the groundtruth labelling (gt). We find that the text+vision system performs well for the classes where the text ranking performs sufficiently well (see Table 6.5). HOG+BOW *vs.* gt: *e.g.* car 93.5%±7.1 *vs.* 98.6%±1.1, or giraffe 88.3%±2.7 *vs.* 91.4%±4.8. If the text ranking fails the groundtruth performs, as to be expected, much better than the text ranked based training, *e.g.* for airplane, camel, kangaroo. These experiments show that the SVM based classifier is relatively insensitive to noise in the training data, as long as there is a critical mass of in-class positive images within the top n_+ text ranked images

Baseline comparison. As a baseline comparison, we investigate performance if no text re-ranking is used, but the n_+ images are sampled uniformly from the filtered images. If the text re-ranking works well, and hence provides good training data, then text+vision improves over the baseline, *e.g.* elephant 84.8% ± 3.3 *vs.* 47.9% ± 4.0, penguin 78.9% ± 13.2 *vs.* 53.7% ± 3.4, or shark 80.7% ± 4.2 *vs.* 42.9% ± 14.4 (see Table 6.5 (B) for all classes). In cases where the text

prec. 15&	ap	bv	bk	bt	cm	cr	dp	ep	gf	gr	hs	kg	mb	pg	sk	tr	ww	zb	avg.	std.
tf rf	42.8	46.4	70.4	60.9	54.5	93.5	67.7	84.8	88.3	62.5	68.8	55.2	72.1	78.9	80.7	57.3	89.7	81.4	**69.8**	8.1 (2.1)
tf ru	38.7	39.0	68.7	62.3	55.5	93.4	63.6	81.8	84.4	64.3	68.9	53.0	74.7	80.0	79.1	58.2	92.5	80.2	68.8	6.6 (1.8)
tu ru	49.9	32.1	61.4	64.1	55.1	82.0	63.9	62.2	90.1	46.6	41.6	49.3	78.8	80.6	62.0	34.7	91.4	78.2	62.4	9.2 (2.5)

Table 6.6: Comparing filtered vs. non-filtered images and baseline. $t\{f|u\}$ denotes training on filtered (unfiltered) images; $r\{f|u\}$ ranking on filtered (unfiltered) images; The first row gives the results for the whole system (HOG+BOW (t+v) in Table 6.5). The second row shows the case where *all* images were re-ranked, including the ones that were filtered out in Section 6.3. The third row shows training and ranking on unfiltered images. The last column states the mean of the classwise standard deviations over five runs of cross-validation, as well as the standard deviation of the means over all classes, in parentheses.

ranking does not perform well the baseline can even outperform text+vision. This is due to the fact, that bad text ranking can provide visually consistent training data, that does *not* show the expected class (*e.g.* for airplanes it contains many images showing: airplane food, inside airplanes/airports, taken out of the window of an airplane). However, the uniformly sampled images still consist of about 35% in-class images (Table 6.1 on page 121) and the n_- are very unlikely to contain in-class images.

Influence of the filter step. In addition to re-ranking the filtered images we applied the text+vision system to all images downloaded for one specific class, *i.e.* the drawings&symbolic images were included. It is interesting to note that the performance is comparable to the case of filtered images (Table 6.6). This means that the learnt visual model is strong enough, to remove the drawings&symbolic images during the ranking process. Thus, the filtering is only necessary to train the visual classifier and is not required to rank new images as evident from row two and three in Table 6.6 where the performance for training on filtered (tf) images is compared to training on unfiltered images (tu); both cases use unfiltered images for the ranking (ru). Training on filtered images and ranking on unfiltered images (tf ru) performs almost as good as both training and ranking on filtered images (tf rf), $68.8\% \pm 1.8$ *vs.* $69.8\% \pm 2.1$, *i.e.* using unfiltered images during testing does not affect the performance significantly. However, using unfiltered images during training and ranking (tu ru), decreases the performance significantly down to $62.4\% \pm 2.5$.

Cross validation. In order to select the appropriate parameter settings we use cross-validation as described in Section 6.5.2. Figure 6.6 and Figure 6.7 show how different parameter

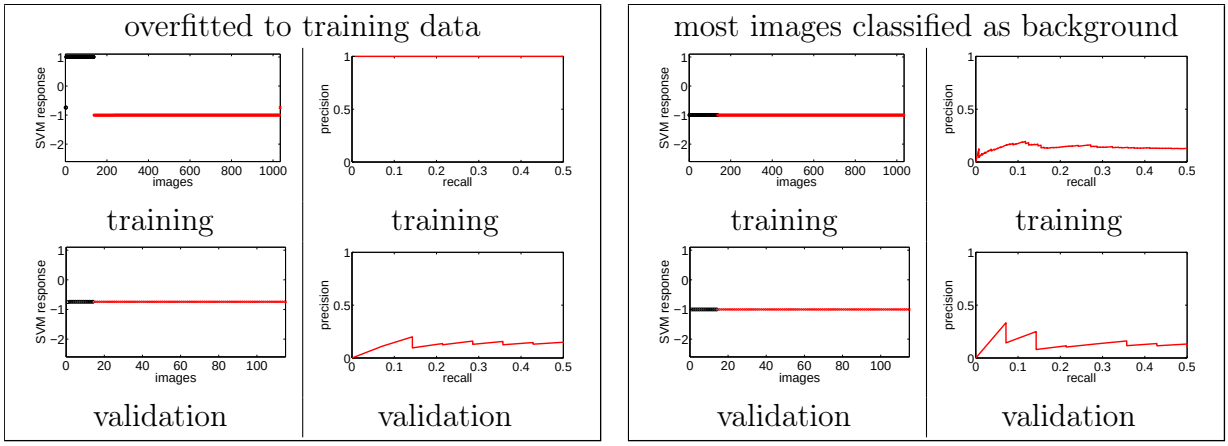


Figure 6.6: Parameter selection examples I. The left column in each box contains the SVM responses for the images and the right column the precision recall curve. The top row in each box shows the values for the training data and the bottom row for the validation data. The images are sorted by SVM response and their label, *i.e.* positive images first (black), then negative images (red). The first parameter setting (shown in the left box) *overfits* to the training data and does not perform well on the validation set. The second setting classifies most images as background and consequently does not perform well either. These examples are for penguin.

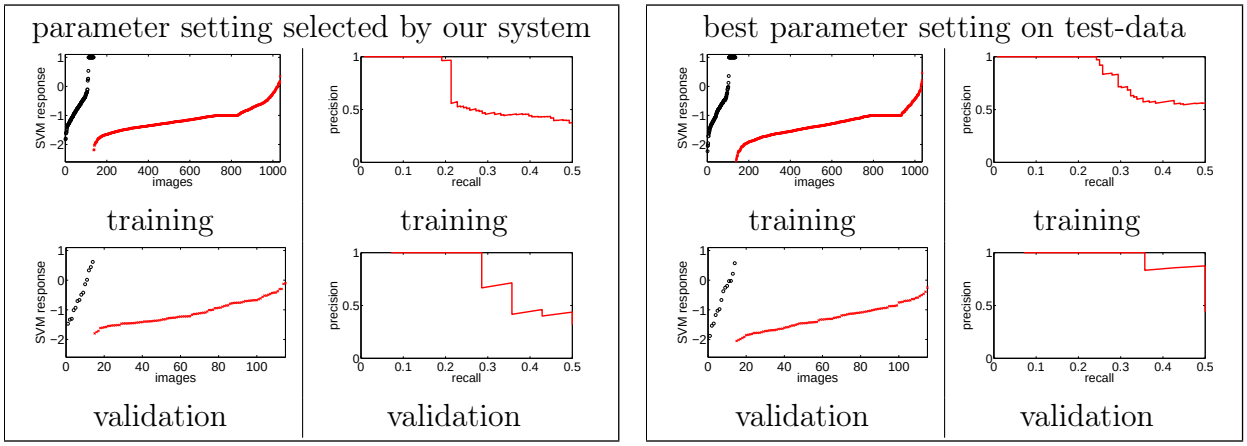


Figure 6.7: Parameter selection examples II. See caption of Figure 6.6. The first parameter setting is the one that was selected by our cross-validation method. It performs reasonably well on both the training and validation-data (test-data: 65.6%). The second setting is the one that actually performs best on the test data (94.0%). Note that the choice is made at 15% recall. Again these examples are for penguin.

settings for the SVM, result in different distributions of SVM response values for the images. Figure 6.6 shows two examples that fail: one parameter setting overfits to the training data. However, this problem is detected on the validation set since all images are classified as negative, and correspondingly the performance (precision at 15%) is bad (as shown by the precision recall plot). In the second example all images are classified as non-class/negative, training as well as validation images. This leads to bad, but detectable, performance as well. Figure 6.7 shows two parameter settings which result in good ranking. It can be seen that the precision

recall curves for both the training and validation set look similarly good. As we use precision at 15% as selection criteria it may happen that the seemingly better model is not selected.

We also investigated a slight adjustment to this method, that ignores "difficult" images. Parameter settings that classify (almost) all images as fore- or background are not useful, neither are those that overfit to the training data. We reject those parameter settings. We then use the "good" parameter settings to train and classify all images. By looking at the distribution of SVM responses (over all parameter settings) we are able to eliminate "intermediate" images, *i.e.* images that are not classified as positive or negative images in the majority of cases. We assume that those images are difficult to classify and we do not use those in our model selection step, because we cannot rely on their class-labels being correct due to the noise. This method does not give a significant improvement over all classes, but improves the performance dramatically for some classes, *e.g.* penguin 90.2%±5.0 from 78.9%±13.2. This modified version of the cross validation is denoted "C" in Table 6.5 on page 133.

6.6.1 Discussion

Training and evaluating the system with the goal to build natural image datasets, *i.e.* treating *abstract* images as *non-class* ($\not\in$) in all stages of the algorithm, as opposed to treating *abstract* images as *in-class*, gives similar performance, however slightly worse. The slight drop in performance can be explained by the fact that the text ranker inevitably returns *abstract* images which are then used as training images. That our method is applicable to both those cases is further supported by the results we retrieve on the Berg and Forsyth (2006) dataset in Section 6.7.1.

We also investigated alternative visual classifiers, topic models (see Section 6.7.2), and feature selection. For feature selection our intention was to find discriminative visual-words and then use these in a Bayesian framework. The discriminative visual-words were obtained based on the likelihood ratio of a visual-word occurring in the foreground to background images [Dorkó and Schmid (2003)]. However, probably due to the large variation in both the foreground and background images, together with the noisy training data, we were not able to match the performance of the SVM ranker.

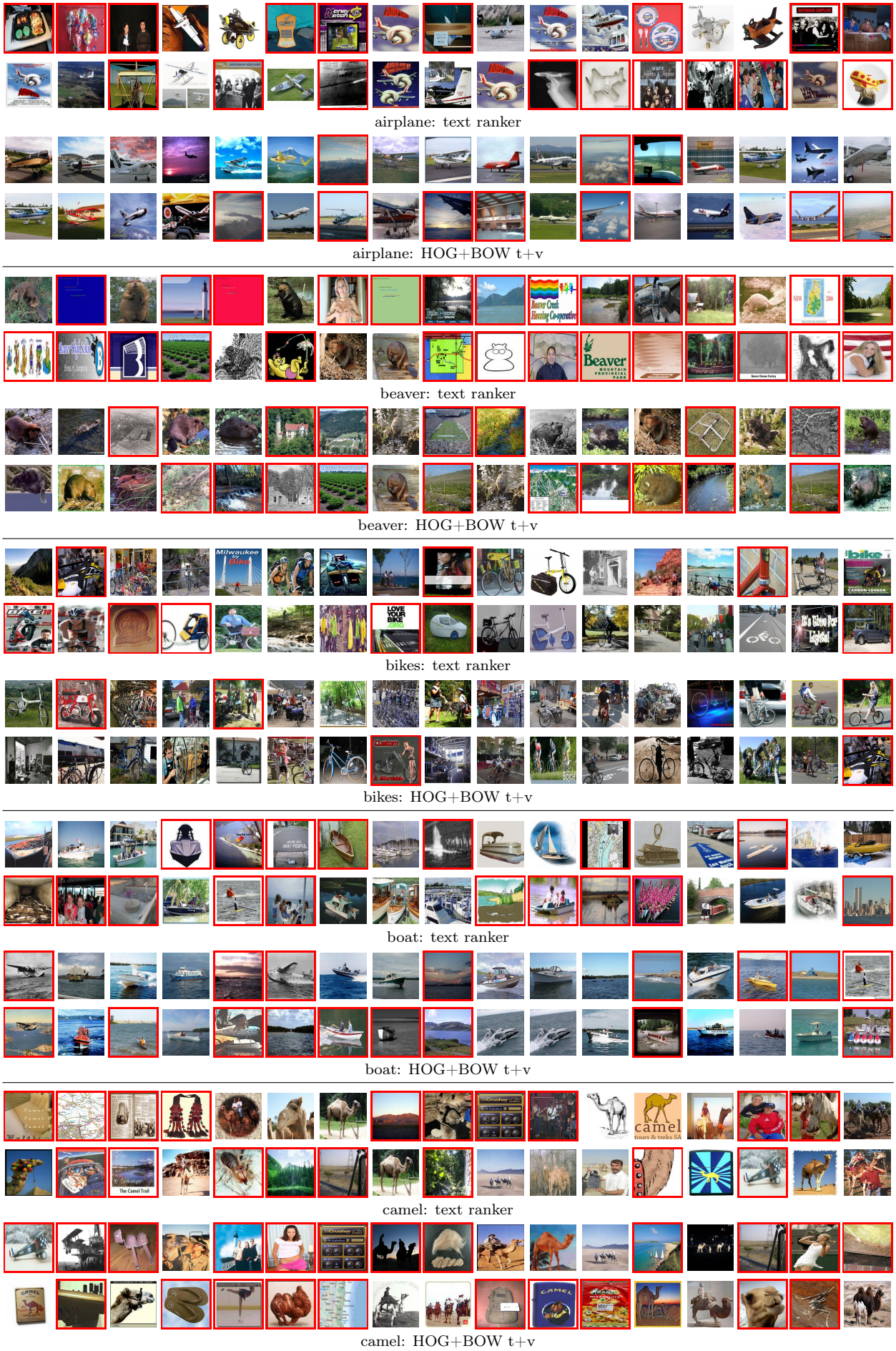


Figure 6.8: Comparing top ranked 34 images using the text ranker, and the full system. Red boxes indicate false positives.

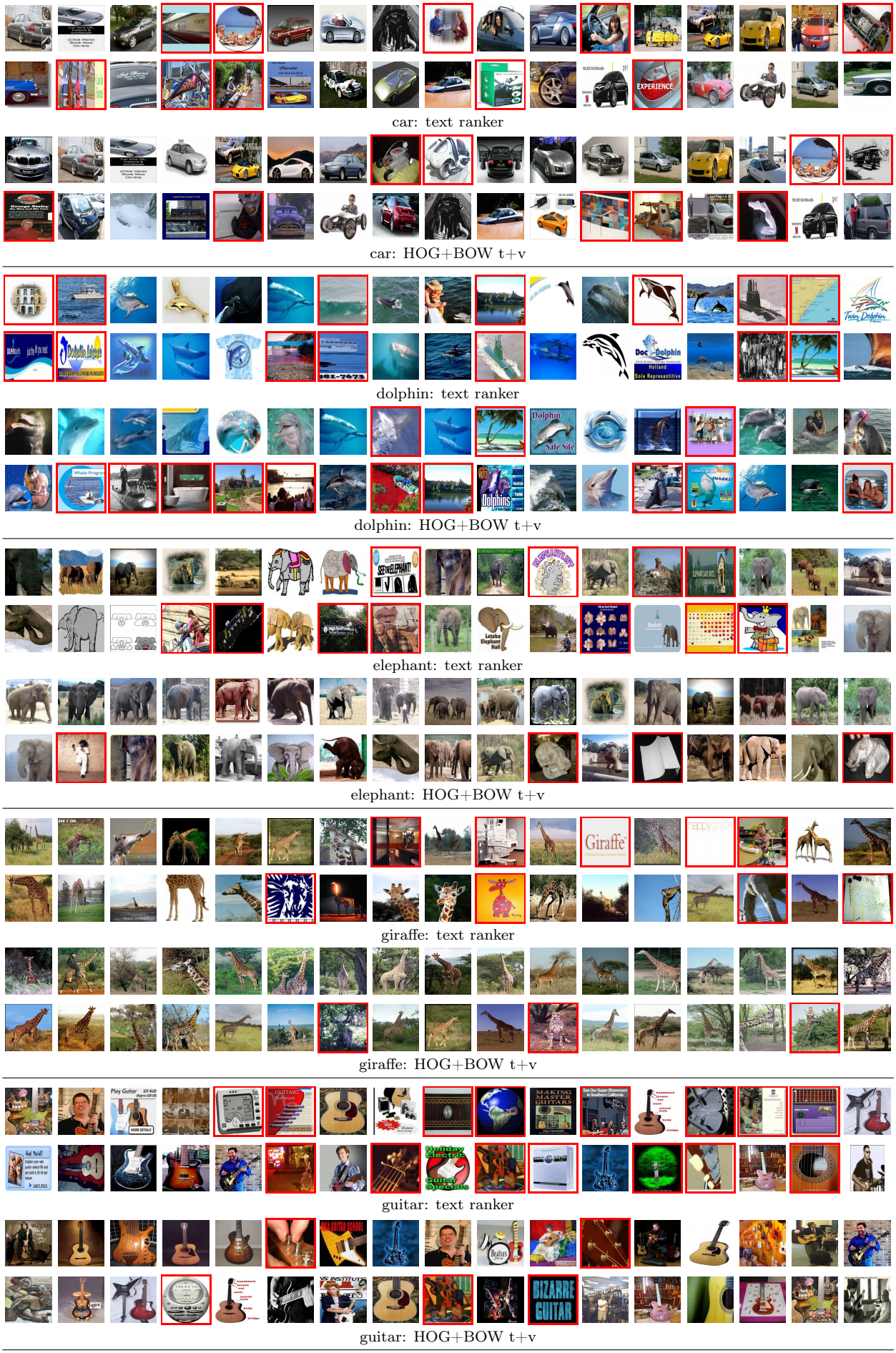


Figure 6.9: Comparing top ranked 34 images using the text ranker, and the full system. Red boxes indicate false positives.

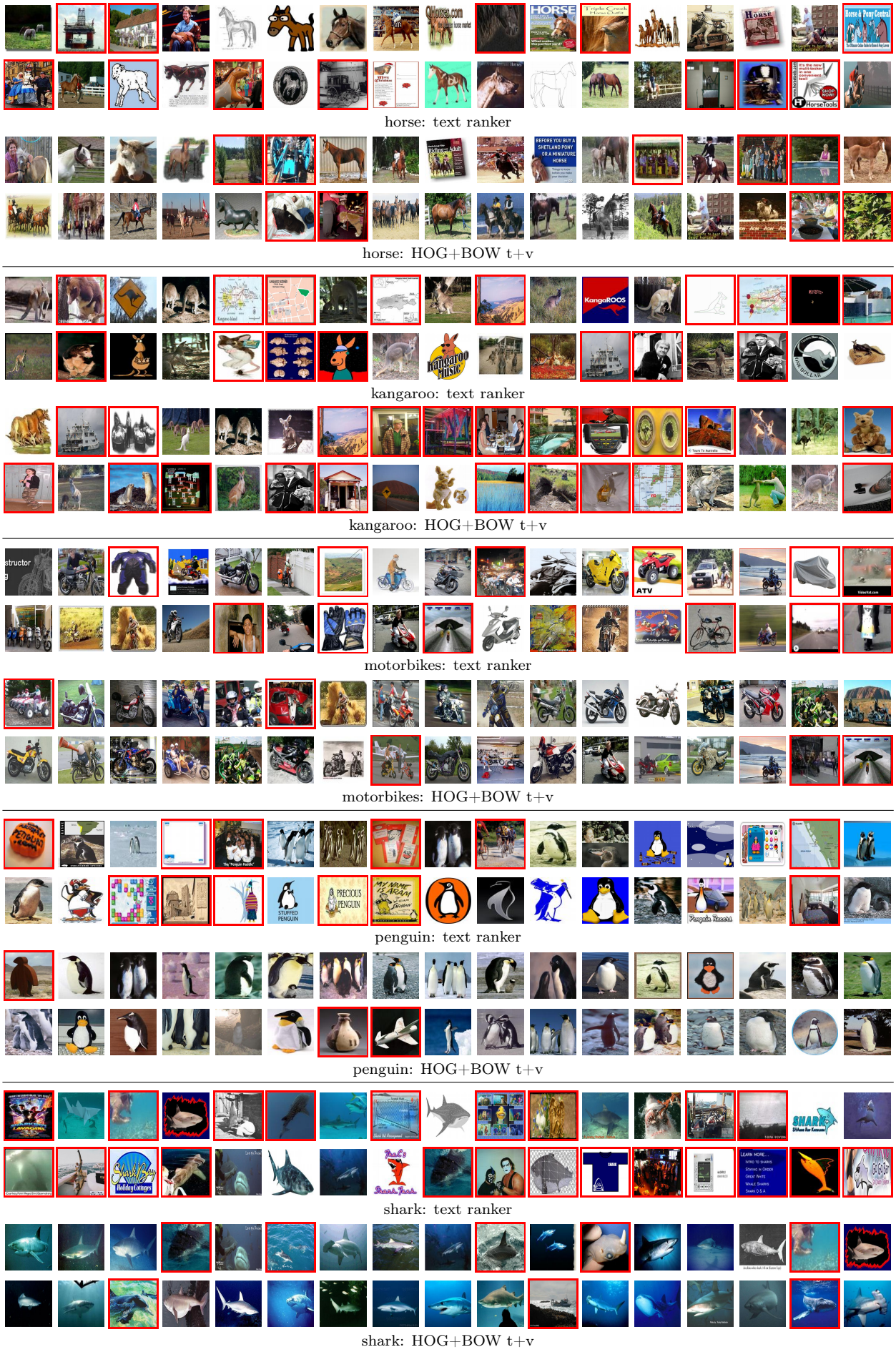


Figure 6.10: Comparing top ranked 34 images using the text ranker, and the full system. Red boxes indicate false positives.

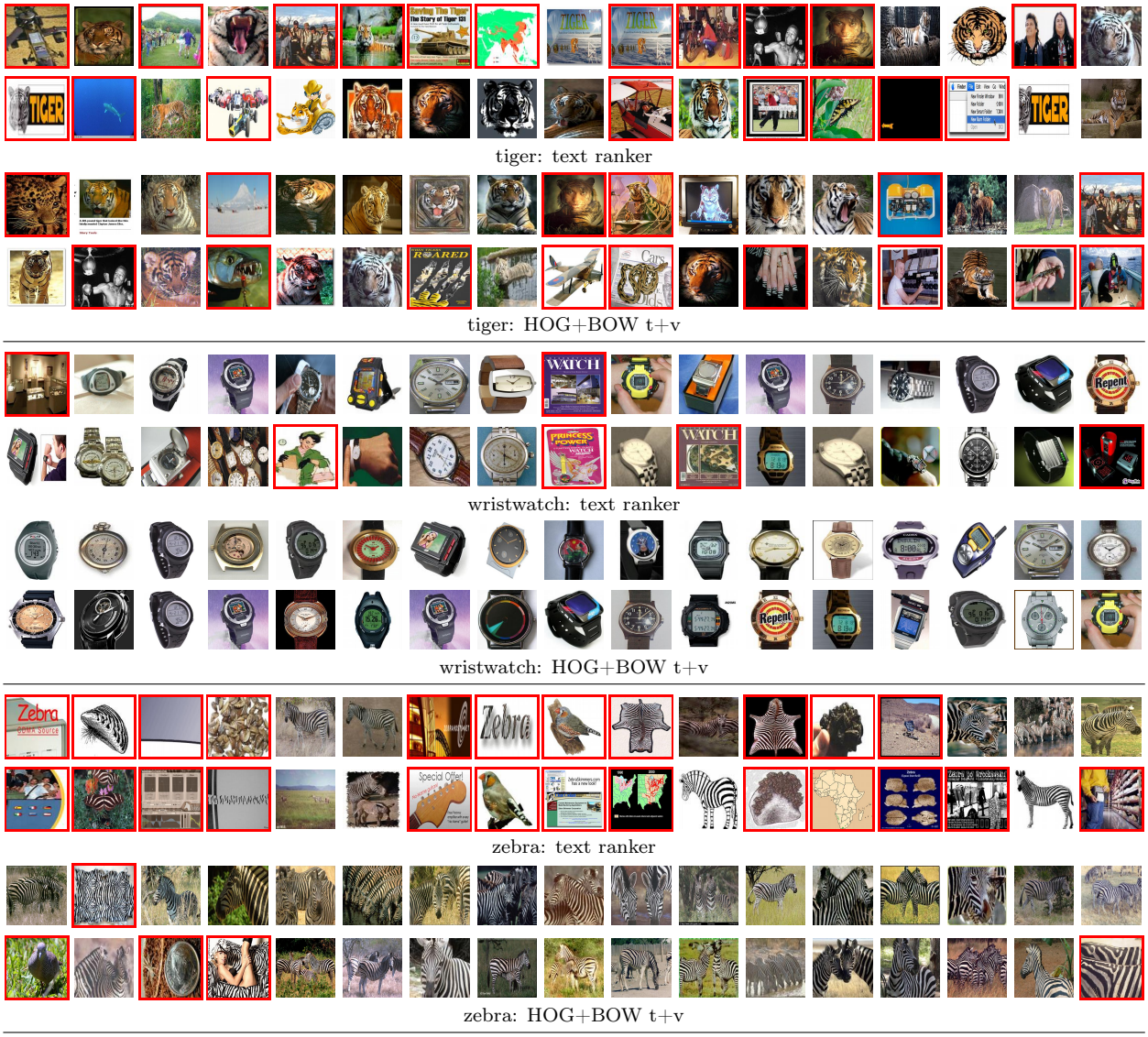


Figure 6.11: Comparing top ranked 34 images using the text ranker, and the full system. Red boxes indicate false positives.

We found that combining the vision and text ranked lists using Borda count[2] [Aslam and Montague (2001)] or similar methods gave a slight improvement on average, but results were very class dependent.

Polysemy and diffuseness, problems with no standard automatic solutions, do affect our results. However, this chapter improves our understanding of the polysemy problem in its different forms. The recent work of Saenko and Darrell (2008) provides very interesting methods for first automatic solutions to the polysemy problem by using dictionary entries and images from web-pages to learn different word senses.

[2]Borda count combines two ranked lists of images by adding the ranks of images that are common to both lists. This assigns a single number to each image and hence defines a new ranking.

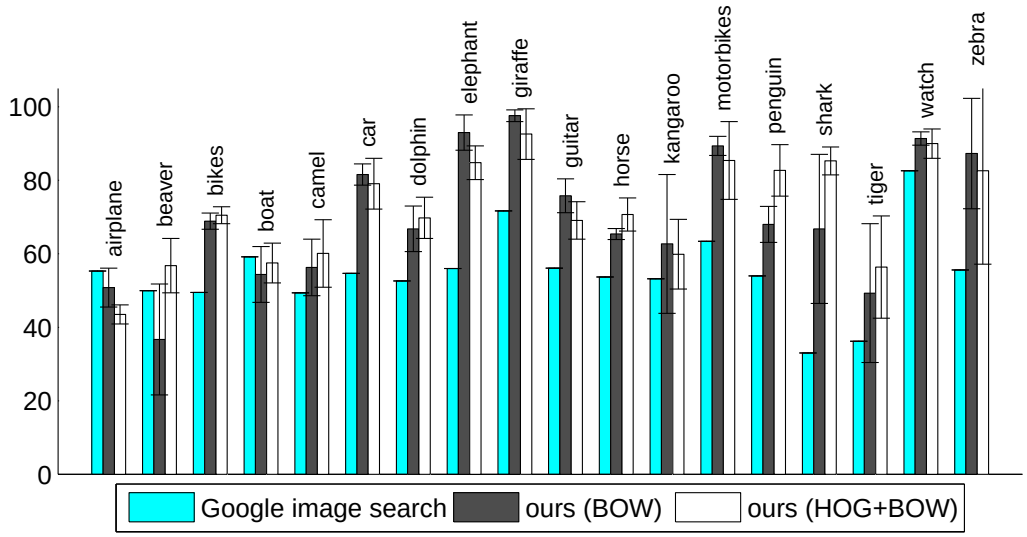


Figure 6.12: Comparison with Google Image Search. Precision at 100 images recall. Compare to highlighted versions in first column of Table 6.5 on page 133 for our algorithm.

6.7 Comparison with Other Work & Methods

This section compares our algorithm to the work of Fergus *et al.* (2005a) and Berg and Forsyth (2006) using their respective datasets. We also give a detailed evaluation of a non-discriminative visual "classifier" and compare the performance of pLSA, LDA, and HDP in Section 6.7.2.

6.7.1 Comparing with Other Work

We compare our algorithm to three other approaches and report the results for $n_+ = 150, n_- = 1000$ and HOG+BOW (see highlighted versions in first column of Table 6.5 on page 133). Again we report mean and standard deviation over five runs of ten-fold cross-validation.

Comparison with Google Image Search. Here we re-rank the images downloaded with `GoogleImages` with our fully automatic system, *i.e.* text-based training of the visual classifier followed by visual re-ranking of `GoogleImages`. Comparative results between our approach and `GoogleImages` are shown in Figure 6.12. As can be observed, our approach achieves higher average precision for 16 out of 18 classes with only airplane and boat being outperformed by the Google results. Those are cases where our text ranker does not perform that well, which increases the noise in the training data and thus explains the decreased visual performance. Images for a selected set of classes returned from both Google and our re-ranking (using the

	airplane	guitar	leopard	motorbike	wristwatch
our	58.5% ± 25.8	70.0% ± 3.9	49.6% ± 9.9	74.8% ± 7.3	98.1% ± 3.0
our (ok)	41.8% ± 18.4	31.7% ± 4.1	22.6% ± 10.2	50.5% ± 8.0	94.4% ± 3.2
Fergus *et al.* (2005a) (ok)	57%	50%	59%	71%	88%
Google (ok)	50%	30%	41%	46%	70%

Table 6.7: Comparison with Fergus *et al.* (2005a). Average precision at 15% recall with one standard deviation. The images are from Google Image Search and were provided by the author. (ok) uses the same annotation as Fergus *et al.* (2005a). The first row of the table treats Fergus' *ok* class as *in-class*, unlike Fergus *et al.* (2005a).

learnt visual classifier applied to the Google images) are shown in Figure 6.13.

Comparison with Fergus *et al.* (2005a). In this experiment we re-rank the Google images provided by Fergus *et al.* (2005a). In order to do so we train on the downloaded images. We downloaded "leopard" in addition to the previously described classes. It is difficult to directly compare the results in Fergus *et al.* (2005a) to our text+vision algorithm. Fergus *et al.* (2005a) treats *ok* images as non-class, whereas our system is not tuned to distinguish *good* from *ok* images. Due to this our system performs slightly worse than Fergus *et al.* (2005a), when measured only on *good* images. However, it still outperforms Google image search on most classes even in this case. Table 6.7 also shows (first row) the results when *ok* images from the Fergus *et al.* (2005a) data are treated as in-class. As expected the performance increases significantly for all classes.

Comparison with Berg and Forsyth (2006). Here we run our visual ranking system on the dataset [Berg (2006)] provided by Berg and Forsyth (2006). In order to do so we downloaded an additional set of six classes (alligator, ant, bear, frog, leopard, monkey) for which no manual annotation was obtained. Figure 6.14 on page 144 compares the results reported in Berg and Forsyth (2006) to the re-ranking by our visual classifier. Note that we are training on *our* set of images, which might stem from a different distribution than the Berg (2006) images, but we are testing on their test data. We compare with the "classification on test data" category of Berg and Forsyth (2006), not to their "final dataset" which includes groundtruth from their manual step. Their provided groundtruth, which treats *abstract* images as *non-class*, was used. Note that our automatic algorithm produces results comparable or superior to those of Berg and Forsyth (2006), although their algorithm requires manual intervention. Figure 6.15

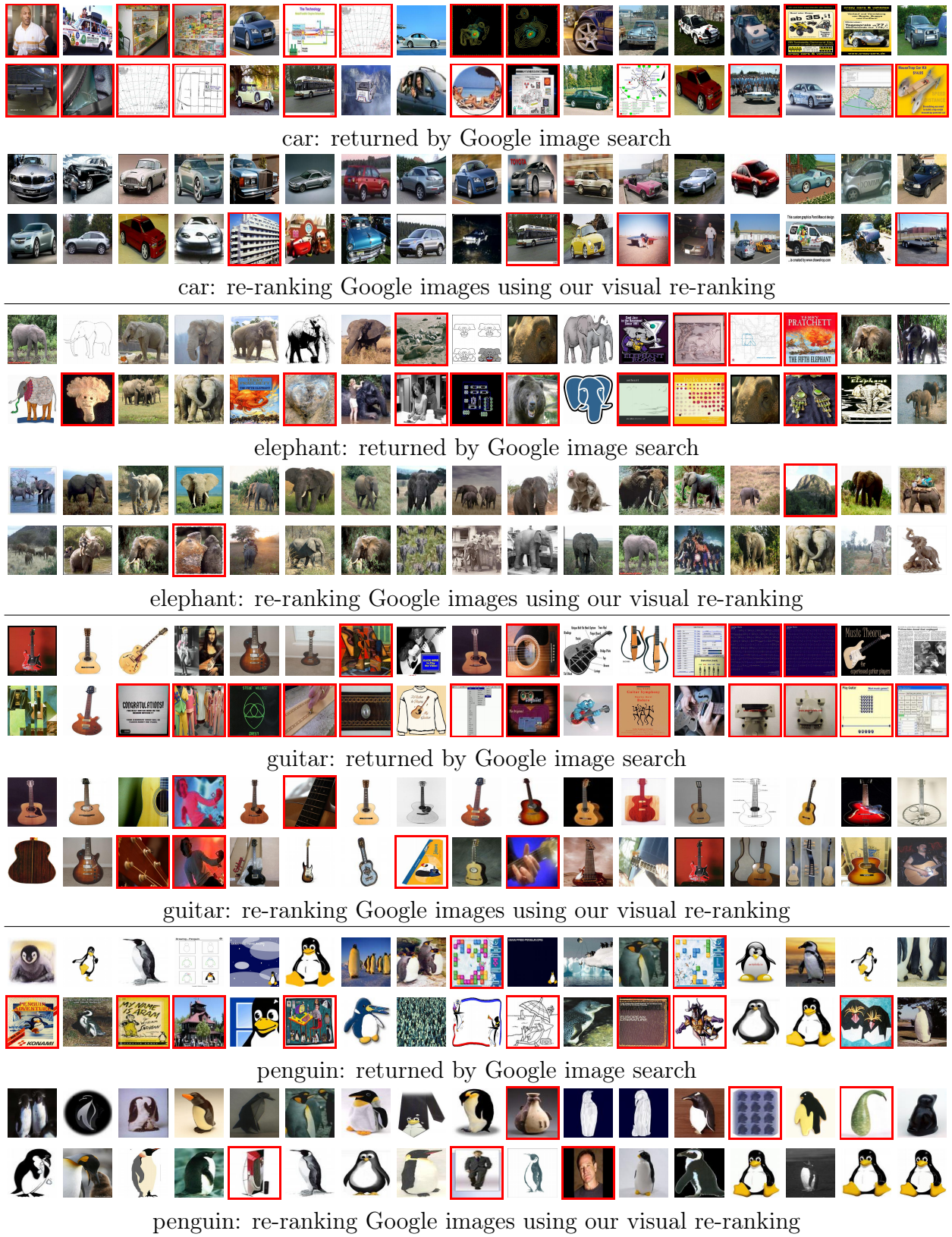


Figure 6.13: Re-ranking Google images. The 34 top-ranked penguin images returned by Google Image Search (in their original order), as well as the 34 top-ranked images after re-ranking Google images using our visual (only) ranking. Red boxes indicate false positives.

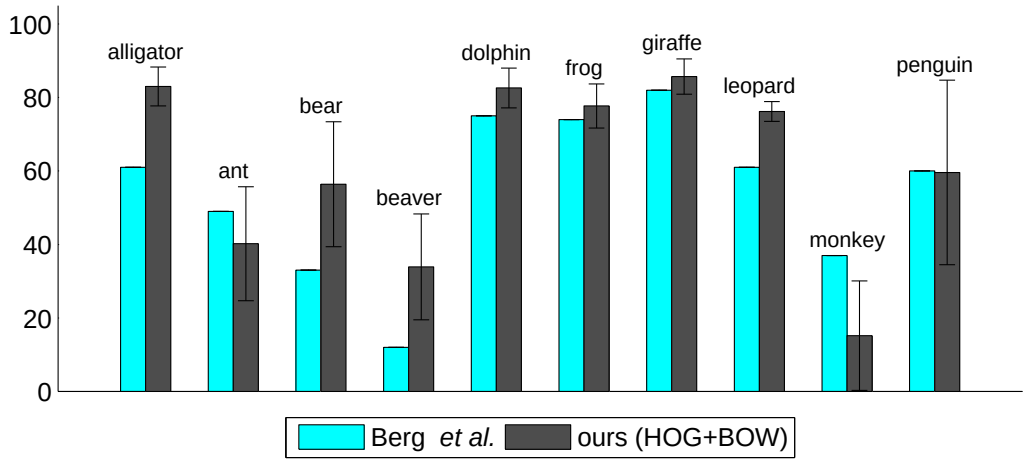


Figure 6.14: Comparison with Berg and Forsyth (2006). Precision at 100-image recall level for the 10 animal classes made available by the authors of Berg and Forsyth (2006). Note that our automatic algorithm is superior in many cases, even though the method of Berg and Forsyth (2006) involves manual intervention.

shows the top 34 images returned by our system applied to each of the classes in the Berg and Forsyth (2006) dataset. Many of the false positives returned for penguin are related to the cartoon "TUX penguin". Berg *et al.* are able to remove those kinds of images during their manual selection step. Monkey and ant are the two classes where we perform worse than Berg and Forsyth (2006). Some of the outliers in these classes are visually close to the objects, others can be explained if one considers our automatic text ranking. They are related to the polysemous problem (*e.g.* ant galaxy) or diffuseness (*e.g.* band names, brand names for all sorts of devices). Many monkey images contain text, graphs or drawings which will be picked up by the visual system and therefore spoil the performance when testing on the Berg and Forsyth (2006) data. This again can be avoided by a manual step, as in Berg and Forsyth (2006). Our filtering step is able to remove some of those images that contain text, but not all of them.

6.7.2 Topic Models

Here we compare our SVM based model to three widely used topic models (pLSA [Hofmann (2001)], LDA [Blei *et al.* (2003)], and HDP [Teh *et al.* (2004)]). The underlying idea of all three topic models is to model each image as a distribution over topics, whereby each topic is a distribution over textons. For a more detailed introduction into topic models see Section 2.2.3.

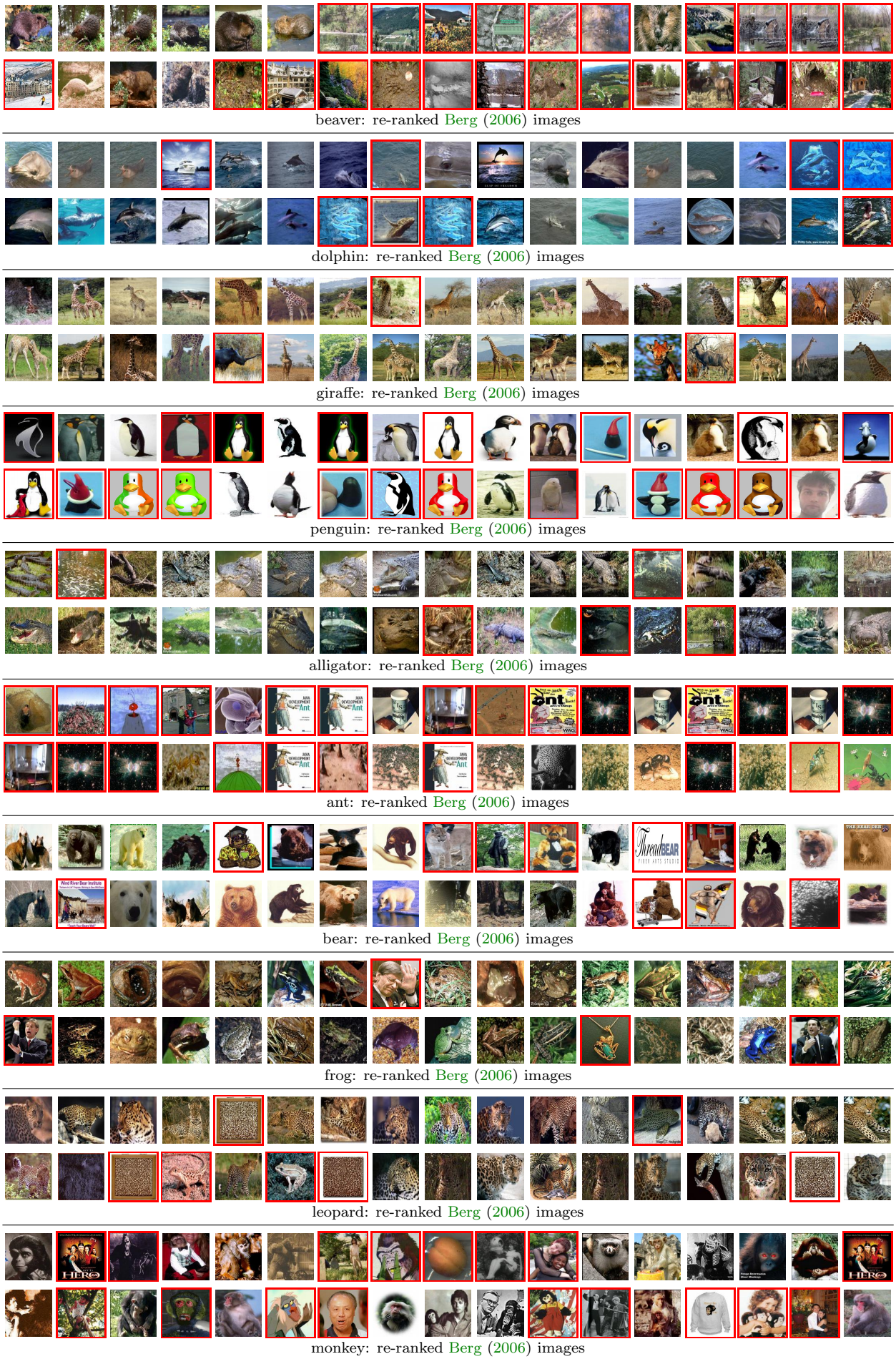


Figure 6.15: Comparison with Berg and Forsyth (2006). the top 34 images returned by our visual ranking system, when applied to the Berg and Forsyth (2006) dataset.

prec. in %	ap	bv	bk	bt	cm	cr	dp	ep	gf	gr	hs	kg	mb	pg	sk	tr	ww	zb	avg.	std
t+v (150/1000)	52.3	39.3	68.6	**66.2**	57.3	87.9	66.5	**78.7**	**83.8**	63.0	57.3	48.6	64.0	72.5	**82.9**	62.6	**93.2**	79.8	68.0	7.2 (2.0)
pLSA(5)	71.2	23.4	70.2	61.9	49.6	85.7	60.0	59.9	78.2	56.7	56.3	38.2	64.5	53.4	55.7	47.0	67.2	63.6	59.0	4.4 (1.1)
pLSA(10)	70.3	21.4	69.0	59.3	53.2	84.2	66.2	61.3	78.0	63.9	65.0	45.4	61.5	62.3	65.6	52.3	74.8	74.6	62.7	5.5 (1.4)
pLSA(50)	74.8	36.7	74.2	62.2	56.5	87.3	65.5	65.6	81.0	66.4	71.5	59.9	71.8	73.2	62.4	54.5	82.9	81.5	68.2	3.8 (0.9)
pLSA(100)	73.7	38.2	73.8	63.1	56.1	88.9	66.4	63.7	80.2	67.0	**72.3**	57.5	71.7	72.7	65.7	55.5	85.4	78.6	68.4	3.9 (0.9)
pLSA(200)	**76.1**	34.6	70.9	60.2	55.8	88.7	65.8	62.6	81.8	**67.6**	67.8	55.2	71.6	73.9	61.9	57.5	86.1	81.0	68.4	3.7 (0.9)
pLSA(500)	76.0	37.5	68.3	61.6	54.4	88.3	66.9	63.5	81.5	65.2	68.2	55.0	73.0	69.6	63.6	**64.6**	86.2	**88.4**	**68.7**	3.7 (1.0)
LDA(5)	70.6	22.4	65.9	60.9	46.0	84.2	62.0	55.8	77.9	57.3	55.7	42.7	63.7	54.2	52.9	52.9	72.5	59.8	58.7	4.5 (1.2)
LDA(10)	68.9	22.7	70.0	59.0	47.2	82.6	63.9	60.5	80.7	65.5	59.9	54.7	62.3	61.1	54.2	62.5	78.3	66.5	62.2	4.6 (1.1)
LDA(50)	72.4	37.8	74.3	64.4	56.8	**89.3**	64.1	63.8	83.6	66.7	66.6	**62.6**	69.7	68.7	50.3	60.5	86.3	82.6	67.8	3.7 (0.9)
LDA(100)	74.5	36.6	74.5	63.2	56.2	85.9	62.9	62.0	83.6	66.1	65.8	62.5	70.5	71.3	53.9	62.0	86.1	80.9	67.7	3.5 (0.9)
LDA(200)	72.9	38.9	69.7	65.1	57.1	86.7	61.5	59.4	82.9	65.4	63.8	61.3	72.2	68.9	48.4	60.8	86.0	83.8	66.9	2.8 (0.7)
LDA(500)	74.3	**42.2**	65.3	62.7	51.6	85.3	61.7	59.5	79.1	66.2	64.2	60.4	72.0	65.5	45.9	58.8	83.0	84.6	65.7	3.0 (0.7)
HDP	71.2	33.6	**76.2**	61.7	53.5	86.1	**71.7**	76.6	81.8	64.8	70.2	60.8	**72.6**	**74.3**	55.0	60.2	83.6	80.5	68.6	2.9 (0.7)

Table 6.8: SVM vs. topic models. Given are the results as inTable 6.5 on page 133 (*i.e.* 15% recall) for all object-classes. We compare the best result achieved, using BOW only, by the SVM model to the performance achieved by using topic models (pLSA, LDA, HDP) where the best topic is *selected using groundtruth*. This gives the topic models a huge advantage. Despite this, the topic models do not clearly outperform the *fully automatic* SVM model. HDP used an average of 257 topics. The last column states the mean of the classwise standard deviations over 10 runs of training the topic models, as well as the standard deviation of the means over all classes, in parentheses.

Given a set of topics and the test images ranked by their topic likelihood the question is how to select the "right" topic which induces the final ranking. This problem of selecting the right topic is difficult and there exist various approaches (see for example Fergus *et al.* (2005a)). To give an idea of the performance of the topic models we avoid this topic selection stage and select the best topic based on groundtruth. The topic that gives the best 15% recall on the images evaluated on groundtruth is selected. This gives a huge advantage to the topic models in Table 6.8, as groundtruth is used during *testing*, but still enables us to conclude that the discriminative SVM classifier is more suitable for our task of ranking images. Each of these models is trained on all images that we want to rank later on. This results in a ranking of all images for each topic. The number of topics is specified for pLSA and LDA and learnt for HDP.

Discussion. The best performing topic models are the pLSA with 500 topics $68.7\% \pm 1.0$ closely followed by the HDP $68.6\% \pm 0.7$ which uses an average of 257 topics. Our SVM based system performs similarly well with $68.0\% \pm 2.0$ *without* using groundtruth for model selection.

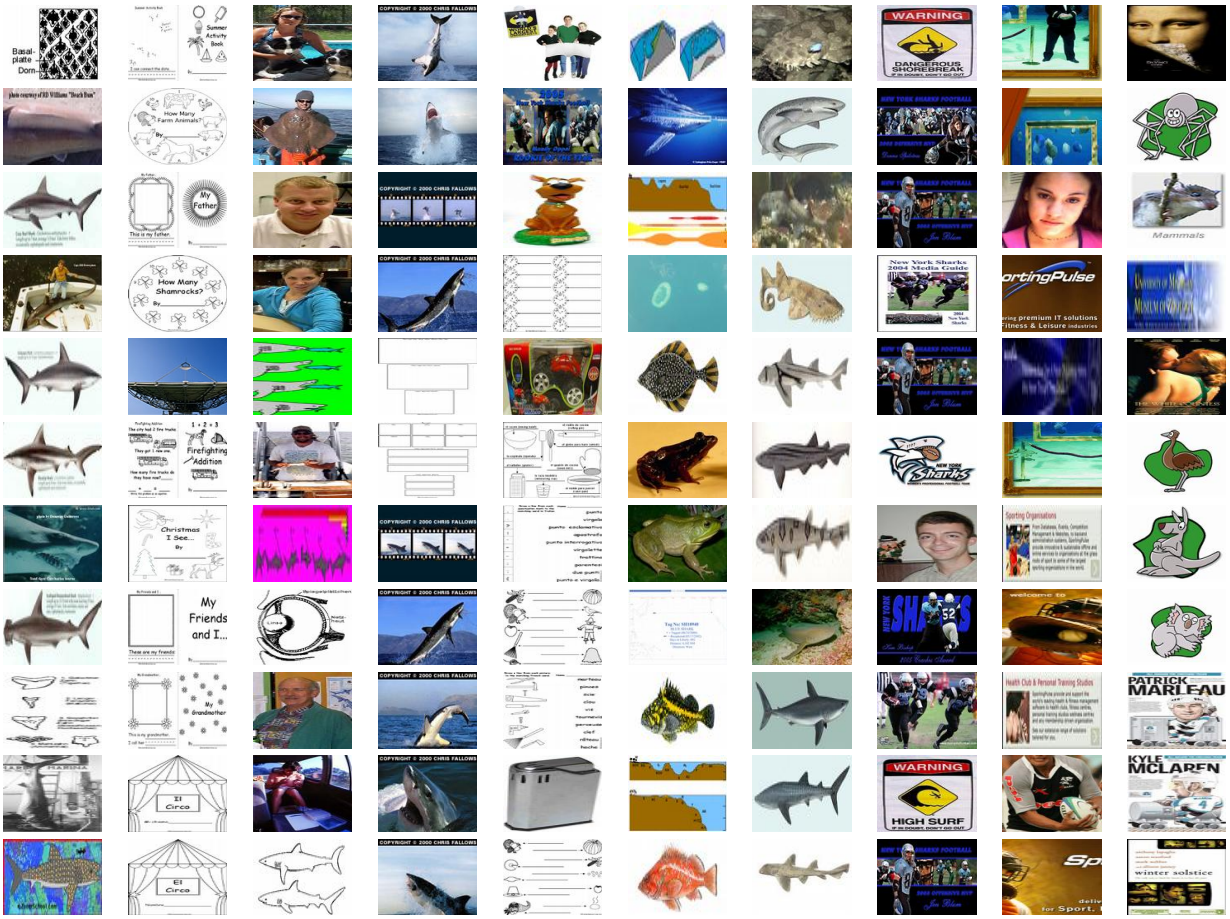
There is also no single topic model that consistently outperforms the SVM on the majority of classes. As mentioned before the results reported for the topic models use groundtruth to select the best topic. This gives an unfair advantage to topic models, but it also gives evidence to the conclusion that the SVM is the "stronger" classifier. It could be possible to combine the high ranked images from two topics, and thereby improve the ranking, although there is no straight forward method to do this. Figure 6.16(b) shows the top ranked images for a ten topic pLSA. Each column represents one topic and the images are ordered by $P(z|d)$. Some topics correspond nicely to background images and others to "shark" images. Although, it has to be noted that the topics seem to capture the overall appearance of the images rather than clustering them discriminatively based on the appearance of shark, for example.

Figure 6.16(a) shows pLSA on the text (*context* meta-data as described earlier) of the images, similar to what has been done in Berg and Forsyth (2006). The goal was to obtain some grouping of the images into junk images and images belonging to the object-class specified by the query that was used to retrieve the set of seed web pages. The context (words surrounding the image HTML-tag) of each image are considered as a document d and each document is modelled as a distribution of topics z (see Section 2.2.3 for details). The experiments revealed that using context is not sufficient to reduce the large number of junk image. Even though some sort of grouping was found, *e.g.* sport images versus the animal, in the case of shark, the amount of confusion is very high. Each column represents one topic and the images are ordered by $P(z|d)$. One reason for the poor performance might be the ratio of good to bad images (only $< 33\%$ are good images; see Table 6.2 on page 121).

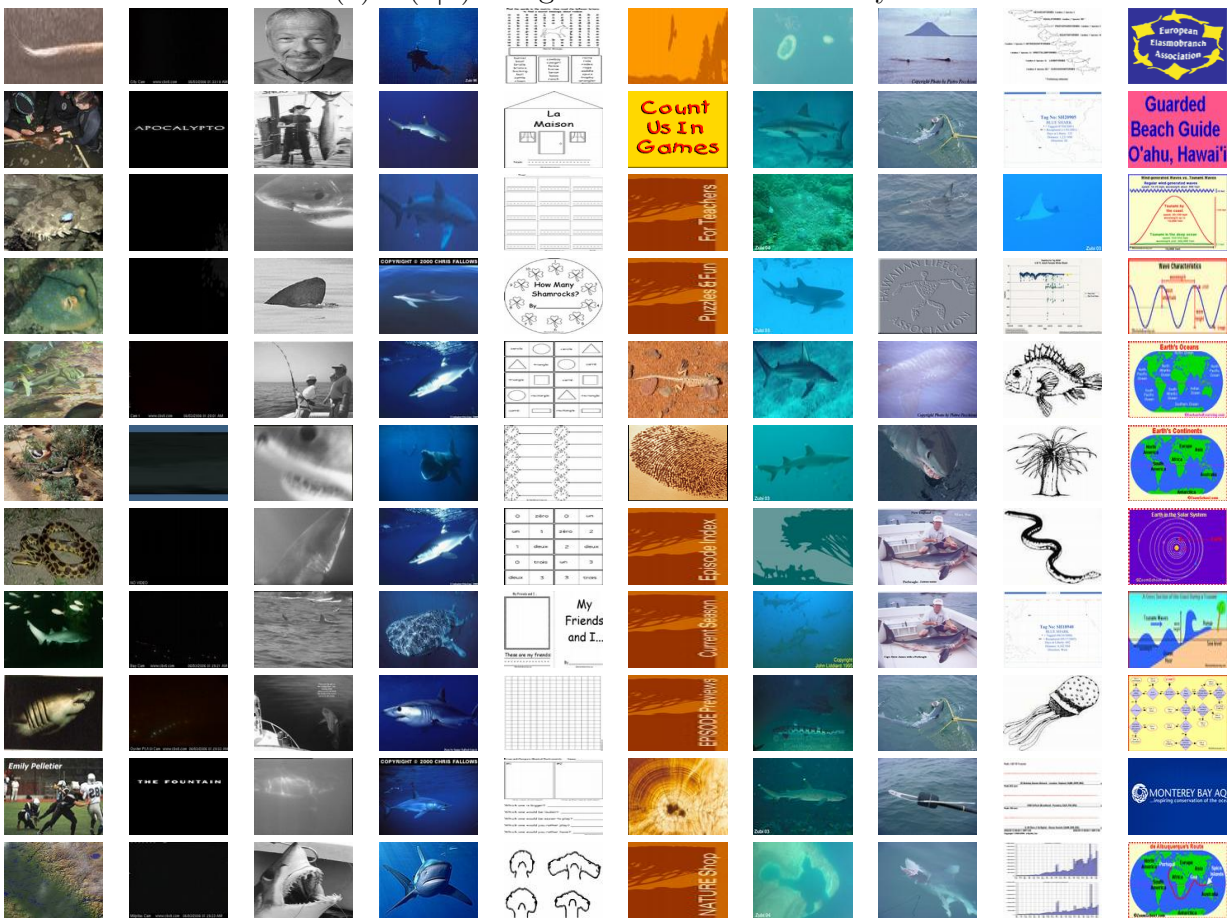
It was also tried to combine context and visual-words which led to very similar results. In the end we found our SVM based system to work better on these challenging images. Despite the great amount of noise in the data our discriminative approach performed better and avoids the difficulty to address the problem of topic selection, which was solved by manual intervention in Berg and Forsyth (2006), something we wanted to avoid.

6.8 Conclusion

This chapter has proposed an *automatic* algorithm for harvesting the web and gathering hundreds of images for a given query class. Thorough quantitative evaluation has shown that the



(a) $P(z|d)$ using the HTML context only.



(b) $P(z|d)$ using the visual-words only.

Figure 6.16: Shark images grouped depending on their pLSA topic probability.

proposed algorithm performs similarly to state of the art systems such as Fergus *et al.* (2005a), while outperforming both the widely used Google Image Search and recent techniques which rely on manual intervention [Berg and Forsyth (2006)]. Such a system can be used to improve current searches. Most importantly, however, it was proven that the information available on web-pages can be used as supervision for visual classifiers. This opens up a wide area and enables us to learn potentially hundreds of visual object-class models automatically.

Chapter 7

Conclusion

This thesis has investigated two different areas of object recognition. In the first part we looked at different methods to model local pixel context with the goal of classifying each pixel and thereby defining a *pixelwise segmentation* of the image into constituent object-class regions.

The second part focused on exploiting the vast number of images that are available on web-pages often together with a textual description in some form (filename, title, text on the web-page). We presented a fully automatic system that uses this information to find images that represent a user specified object-class well enough to learn a visual model. This visual model was then applied to all downloaded images, in order to rank them based on their visual similarity to the classes of interest.

We now show how the harvested images can be used to train segmentation models without the need for tedious manual labelling and then give an outlook for future work.

7.1 Using Harvested Images for Segmentation

In this section we give a short overview of two preliminary experiments that show how the images collected with our harvesting method can be used to train our Random Forest based segmentation system.

In Chapter 5 we used strong supervision (pixelwise segmentation) during training. In order to learn from harvested images, the training needs to succeed on weakly labelled images, *i.e.* only the objects in an image are known but not their position. Figure 7.1 shows results for this scenario on the 21-class MSRC dataset. Here we use the Random Forest classifier as described in

Figure 7.1: Weakly supervised segmentation on the 21-class MSRC dataset. These images show the quality of our preliminary results using *only* weakly supervised training data. It can be seen that "stuff" categories like road, sky, grass and tree are segmented reasonably well. Also, objects that occur together with different backgrounds, such as signs and bicycles are segmented well. The right column shows the corresponding segmentations using strong supervision as in Chapter 5.

Figure 7.2: Harvested segmentation. These results demonstrate how a segmentation model can be learnt from the images harvested from the web using our text+visual ranker from Chapter 6. These segmentations are obtained *without* any user interaction or supervision other than the specification of the object class, *i.e.* car. The failures can be explained by strong viewpoint changes or visually similar objects as the aeroplane, which is not sufficiently represented by the negative training images.

Chapter 5, but instead of using the pixelwise segmentation during training, *only* the occurrence of object classes in an image was used as supervision. For example, if bicycle and road occur in an image together, *all* pixels in that image are labelled as bicycle *and* road, for training. The results on the 21-class MSRC dataset show that, despite this weak supervision, the Random Forest classifier is able to provide reasonably good segmentations. The overall performance is 52.1% pixelwise classification accuracy compared to 71.5% with strong supervision. It needs to be pointed out, however, that the system fails to learn a segmentation model for some of the object-classes (*e.g.* cow, aeroplane) as they tend to occur in images together with the same classes (grass, sky) which in the case of weak supervision does not enable the classifier to distinguish between, *e.g.* cow and grass, as it has never seen cows in a different context.

The second experiment trains the Random Forest classifier on the top 100 images returned by our text+vision harvesting algorithm for the object-class "car". In addition, a set of 200 randomly sampled background/negative images is used. Similar to the previous experiment, all pixels in an image are either labelled as car or background. We then apply this learnt classifier to the 21-class MSRC dataset. Figure 7.2 shows some example images and the corresponding segmentation using the full Random Forest based segmentation system from Chapter 5. If the viewpoints are different, *e.g.* from above, the cars are not segmented. In general there are not many false positives in this two class problem, *i.e.* not many object are wrongly classified as cars, with the main exception of aeroplanes, which look like "cars with wings". Reflective surfaces, *e.g.* water, also tend to be confused with cars, probably due to the similarity with windscreens. This problem could be overcome by a better selection of negative images, which should contain aeroplanes and water in this case.

Both these results are quite encouraging and further demonstrate the strength of the Random Forest classifier and its robustness to label noise. Using the ideas described in the next section, it can be expected that these already strong results could be improved further.

7.2 Future Work

The following two sections present a few ideas and possibilities on how the methods described in this thesis can be extended.

Figure 7.3: Bottom-up segmentation based node-tests. (a) visualises the original idea where the weighted difference of the mean responses of two rectangular regions over the red and green channel is used in the node-test; (b) extends this idea to incorporate bottom-up segmentation as a data driven way to define the low-level features.

7.2.1 Object Segmentation

In Chapter 4 we were minimising the intra-class distance of the model histograms, based on the Maximum Likelihood Estimate in combination with the Kullback-Leibler divergence. It would also be desirable to maximise the inter-class distance when building the single-histogram class models, which would result in a more discriminative classifier. Maximising the inter-class distance or generally merging the class histograms in a discriminative way is left for future research.

In Chapter 5, we investigated another line of work that is based on the discriminative Random Forest classifier. Next, we present three directions to extend our Random Forest based object segmentation work.

Employing bottom-up context. In Chapter 4, we introduced the idea of using bottom-up segmentation in order to identify the relevant context for a specific pixel in a content driven manner. This idea could be exploited in combination with the strong Random Forest classifier of Chapter 5. Currently, our Random Forest classifier selects rectangular responses (see Figure 7.3(a)) from a "randomly" created pool, as most Random Forest based classification or segmentation algorithms that are described in the literature do. Due to the fact that the rectangular responses, in effect, encode the mean response in an image region relative to the image pixel, or the weighted difference between two mean responses in the case of two rectangles, it is imaginable that instead of using fixed sized rectangles, small segments (as in Section 4.5)

can be used to compute a weighted difference of the mean responses in the regions defined by those segments. Figure 7.3(b) illustrates this idea. Instead of the red and green rectangle that denote the response over the respective colour channels, arbitrary shapes that are induced by the bottom-up segmentation could be used to compute the response over the red and green channel. This only sketches the basics of the idea and it is not entirely clear what the best choices for the details of the implementation are. It should also be noted that there will be huge implications in terms of efficiency, as integral images cannot be used anymore. Another problem might be the fact that the bottom-up segments cannot be shifted across the images (as it is done with the rectangles in order to get the responses for the whole image). Therefore, it might be advantageous to take weighted responses of neighbouring segments. This is left for future research and could be an interesting way to build on top of the feature selection capabilities of Random Forests using data driven image specific low-level context.

Use of more global classifiers. Chapter 5 has proven that Random Forests can deliver state of the art results on the segmentation task. However, it also seems to be the case that current work hits a boundary with respect to what can be achieved using local pixel-context. Shotton *et al.* (2008), Csurka and Perronnin (2008) already investigated so-called image level priors (ILPs) or detection level priors (DLPs). The idea is to use more global classifiers to return a classification for the whole image or for the detection of objects in the image [Everingham *et al.* (2006, 2007, 2008)]. These global models can be based on the bag of visual-words model or pyramid histograms [Lazebnik *et al.* (2006)], for example. They then induce ILPs or DLPs respectively that can strengthen the more local segmentation methods, *e.g.* by adjusting the empirical class posterior returned by the Random Forest classifier. Future work would include a thorough investigation of the image and detection level priors that showed big improvements in Shotton *et al.* (2008), Csurka and Perronnin (2008). Despite their boost in performance current methods seem rather ad-hoc and are not learnt jointly together with the segmentation models which might be a worthy direction to explore. In particular, it might be interesting to look into the idea of hierarchical Random Forests, which would allow one to stay within one framework, while addressing classification in a more global manner. For example, a high-level Random Forest could be learnt using spatial pyramids or bag of visual-words, also encoding the pixel positions in the leaf nodes in addition to the empirical class posteriors. Further

low-level Random Forests can then be used to classify those areas identified by the high-level forest. The low-level forests can use the class posteriors from the high-level ones as "priors". Alternatively, object-class or region specific low-level Random Forests could be trained. Closely related to this stream of work is the incorporation of explicit context modelling, as for example in He *et al.* (2004), Kumar and Hebert (2005), Rabinovich *et al.* (2007).

Weakly supervised Random Forest based segmentation. In Chapter 5 we use strong supervision in the form of a pixelwise labelling of the training data. Our preliminary experiments in the previous section showed that a weaker form of supervision, where only a list of the object-classes occurring in an image is provided, can be used together with the Random Forest classifier. The result is a relatively context dependent visual model, which leads to deterioration for some object classes. However, it is reasonable to assume that larger amounts of training data with varying context will improve performance. Further work in this direction seems interesting and promising.

7.2.2 More Harvesting

Aside from improving on the text ranking which seems difficult without using more complex features than the binary feature vector, probably the biggest problem is the one of polysemy and diffuse classes. Adding additional information to the textual model by employing a dictionary as in Saenko and Darrell (2008) or other sources such as Wikipedia in Wang and Forsyth (2008) could be a way to improve the text ranking. One way to deal with the polysemy and diffusion problems would be to leverage multi-modal visual models to extract the different clusters of polysemous meanings, *i.e.* for tiger: Tiger Woods or the animal. It would also be interesting to divide diffuse categories described by *e.g.* the word airplane (airports, airplane interior, airplane food) into smaller visually distinctive categories. This could extend our work along the lines of Saenko and Darrell (2008). Preferably one would learn a unified model based on textual and visual attributes that represents the various senses of a word. Similar to Saenko and Darrell (2008) a sense model learnt from a dictionary can then be related to the text part of the unified model. The advantage of our system is that we can rank images for which no meta-data is available which might be more challenging for a unified model. Wang and Forsyth (2008) for example, heavily rely on the text model for its final ranking and their vision only ranking is

not particularly strong. For most of the object-classes in Berg (2006), their pure image based ranking is significantly worse than the text ranking, whereas in our case we use the pure visual object-class model with good results.

Currently we use the text ranker and the SVM in two consecutive stages. Combining the probabilistic outputs of text and SVM as in Wang and Forsyth (2008) remains an interesting addition. Using a "probabilistic extension" [Platt (1999)] for the SVM or a relevance vector machine (RVM [Tipping (2000)]) should provide descent probability estimates based on the visual model. The seven dimensional feature vector, however, currently does not deliver good estimates of the posterior probabilities. This might be due to the restricted feature space (only 128 possible feature combinations) and can be overcome with a more sophisticated text model based on a larger body of text. It is also worth investigating a more elaborate use of the text ranking in order to train the SVM. For example, the top ranked images do not need to be treated equally, but could be weighted depending on the text ranking score.

Our algorithm does not rely on the high precision of top returned images, *e.g.* from Google Image Search. Such images play a crucial role in Fergus *et al.* (2005a), Li *et al.* (2007) and Wang and Forsyth (2008) who use the Caltech dataset to improve over the visual models learnt from Flickr images. Future work could take advantage of this precision by exploiting them as a validation set to further improve on the ranking.

A large part of this thesis employed the Random Forest model and it would be interesting to see its application to the harvesting problem. In Section 7.1 we trained the Random Forest classifier on harvested data to segment new images into their constituent object regions thereby making use of the high quality images retrieved by our harvesting approach. These preliminary results are quite encouraging and can build a basis for further work in this area.

Chapter 8

Appendix

This appendix provides the derivation details for our single-histogram class models and for the mixture model of Section 4.4.1.

8.1 Single-Histogram Class Models

This section gives the derivation for the minimum of (4.8) in the Kullback-Leibler divergence case, and also for the Euclidean distance case as defined in (4.10).

8.1.1 Derivation of Single-Histogram Class Models

As the sum of convex functions is convex, and our feasibility region ($q_i \geq 0$) is also convex the Karush-Kuhn-Tucker (KKT) conditions are sufficient for a global minimum [Boyd and Vandenberghe (2006)].

Kullback-Leibler Divergence. This paragraph derives the solution of the minimisation problem using the KKT conditions.

$$E_{KL} := \sum_{j=1}^{N_c} n^j D_{KL}(\mathbf{p}^j \parallel \mathbf{q}) \quad \text{subject to} \quad \parallel \mathbf{q} \parallel_1 = 1,\ q_i \geq 0\ \forall i \quad , \tag{8.1}$$

as introduced in (4.8), can be simplified to an E'_{KL} and E''_{KL} which reach the global minimum

for the same values of $\mathbf{q}$:

$$\begin{aligned} E'_{KL} &:= -\sum_j n^j \sum_i p_i^j \log q_i \quad , \\ E''_{KL} &:= -\sum_i \sum_j n^j p_i^j \log q_i \quad . \end{aligned}$$

Using a Lagrange-Multiplier together with -E''_{KL} and $g_i = -q_i$ the following minimum $\hat{\mathbf{q}}$ is obtained, when minimising (8.1):

$$\begin{aligned} L_{KL} &= \sum_i \sum_j n^j p_i^j \log q_i + \lambda \left(1 - \sum_i q_i\right) + \sum_i \mu_i g_i \\ \overset{\frac{\delta}{\delta q_i}, KKT}{\Rightarrow} & \quad \frac{\sum_j n^j p_i^j}{q_i} - \lambda - \mu_i = 0,\ i = 1..V, \quad \sum_i q_i = 1, \quad g_i \le 0, \quad \mu_i \ge 0, \quad \mu_i g_i = 0 \\ \overset{\mu_i g_i = 0, g_i = -q_i}{\Rightarrow} & \quad \left(\frac{\sum_j n^j p_i^j}{q_i} - \lambda\right) q_i = 0,\ i = 1..V, \quad \sum_i q_i = 1, \quad g_i \le 0 \qquad (8.2) \\ \overset{\sum_i}{\Rightarrow} & \quad \sum_i \sum_j n^j p_i^j - \sum_i \lambda q_i = 0, \quad \sum_i q_i = 1, \quad g_i \le 0 \\ \Rightarrow & \quad \lambda = \sum_i \sum_j n^j p_i^j \qquad (8.3) \\ 8.2, 8.3 \Rightarrow & \quad \hat{q}_i := \frac{\sum_j n^j p_i^j}{\sum_i \sum_j n^j p_i^j} = \frac{\sum_j n^j p_i^j}{\sum_j n^j}, \quad , i = 1..V \quad . \qquad (8.4) \end{aligned}$$

It is obvious that $-g_j = \hat{q}_j \ge 0$, since in (8.1) only sums of positive numbers (histogram bins and exemplar weights) occur, for the same reason $\mu_i \ge 0$ and therefore all KKT conditions fulfilled and (8.4) is derived.

Euclidean Distance. See (4.10) for the problem definition repeated here:

$$E_{L2} := \sum_{j=1}^{N_c} n^j D_{L2}(\mathbf{p}^j, \mathbf{q}) \quad \text{subject to} \quad \| \mathbf{q} \|_1 = 1,\ q_i \ge 0\ \forall i \quad . \qquad (8.5)$$

This paragraph derives the solution to the problem of minimising E_{L2} subject to the constraints defined above:

$$
\begin{aligned}
E_{L2} &= \sum_i \sum_j n^j (p_i^j - q_i)^2 \\
\overset{\frac{\delta}{\delta q_i}}{\Rightarrow} & -\left(\sum_j 2n^j \cdot \left(p_i^j - q_i\right) \right) = 0 \quad , \forall i; \\
\Rightarrow & \sum_j n^j q_i - \sum_j n^j p_i^j = 0 \\
\Rightarrow & \hat{q}_i := \frac{\sum_j n^j p_i^j}{\sum_j n^j}, \quad , i = 1..V \quad .
\end{aligned} \tag{8.6}
$$

Apparently all constraints are fulfilled and the solution (8.6) is equivalent to (8.4).

8.1.2 Histogram Mixture Model

Kullback-Leibler Divergence. See Section 4.4.1 (4.11) for the problem definition. In order to reduce the number of subscripts we use the following equation:

$$
\mathbf{h} \overset{D}{=} \alpha \mathbf{a} + (1 - \alpha) \cdot \mathbf{b}\,, \quad \text{with } \mathbf{a} \neq \mathbf{b} \quad , \tag{8.7}
$$

where $\mathbf{a}$ and $\mathbf{b}$ define two single-histogram class models. Using the definition (4.2) for D_{KL} it follows:

$$
\begin{aligned}
F_{KL} &:= \sum_{i=1}^{V} h_i \log \left(\frac{h_i}{\alpha a_i + (1-\alpha) b_i} \right) = \sum_i h_i \log \left(\frac{h_i}{\alpha \left(a_i - b_i\right) + b_i} \right) \\
\hat{\alpha} &:= \arg\min_{\alpha} \left(F_{KL}\right) \\
&= \arg\min_{\alpha} \left(\sum_i h_i \log \left(\alpha a_i + (1-\alpha) b_i\right) \right) \\
\overset{\frac{\delta}{\delta \alpha}}{\Rightarrow} & -\frac{\delta F_{KL}}{\delta \alpha} = \sum_i h_i \frac{a_i - b_i}{\alpha a_i + (1-\alpha) b_i} = 0 \quad .
\end{aligned}
$$

Since there is no obvious general simple closed form solution, it is shown that F_{KL} is convex and thus Gauss-Newton optimisation is a good way to determine $\hat{\alpha}$. Imposing the additional constraint $0 \leq \hat{\alpha} \leq 1$ gives:

$$\frac{\delta^2 F_{KL}}{\delta^2 \alpha} = \sum_i h_i \left(\frac{a_i - b_i}{\alpha (a_i - b_i) + b_i} \right)^2 \geq 0 \quad ,$$

and therefore F_{KL} is *convex*. In fact, $\frac{\delta^2 f_{KLD}}{\delta^2 \alpha} > 0$ since the histogram bins are all positive, $\mathbf{a} \neq \mathbf{b}$ and $\mathbf{b} \neq \mathbf{0}$.

Euclidean Distance. Here we derive the closed form solution for the global minimum for the case of D_{L2}, also using (8.7):

$$\begin{aligned}
F_{L2} &:= \sum_i (h_i - (\alpha a_i + (1-\alpha) b_i))^2 \\
\hat{\alpha} &= \arg\min_\alpha (F_{KL}) \\
&= \arg\min_\alpha \left(\sum_i \left(h_i^2 - 2 \cdot h_i(\alpha(a_i - b_i) + b_i) + \alpha^2 (a_i - b_i)^2 + 2\alpha b_i (a_i - b_i) + b_i^2 \right) \right) \\
\overset{\frac{\delta}{\delta\alpha}}{\Rightarrow} & \sum_i \left(-2 \cdot h_i (a_i - b_i) + 2\alpha (a_i - b_i)^2 + 2 b_i (a_i - b_i) \right) = 0 \\
\Rightarrow & \quad 2 \sum_i (a_i - b_i) \cdot (b_i - h_i) + 2\alpha \sum_i (a_i - b_i)^2 = 0 \\
\Rightarrow & \quad \alpha = \frac{\sum_i (a_i - b_i) \cdot (h_i - b_i)}{\sum_i (a_i - b_i)^2} \quad .
\end{aligned}$$

Since $\frac{\delta^2 f_{L2}}{\delta^2 \alpha} > 0$ as well, it follows that F_{KL} is convex and therefore, in order to fulfil $0 \leq \hat{\alpha} \leq 1$, $\hat{\alpha}$ can be set to $\hat{\alpha} = \arg\min_{\alpha'} (F_{L2}(\alpha'))$ with $\alpha' \in \{0, 1, \max(0, \min(1, \alpha))\}$.

8.2 Statistics

This sections gives a rough idea of when all the hard work was done.

8.2.1 Code Statistics

Figure 8.1 and 8.2 report some statistics about my local code repository:

- Report Period: `2005-10-19 12:10:43` to `2009-04-01 15:04:24` (`1260` days)
- Total Files: `1417`

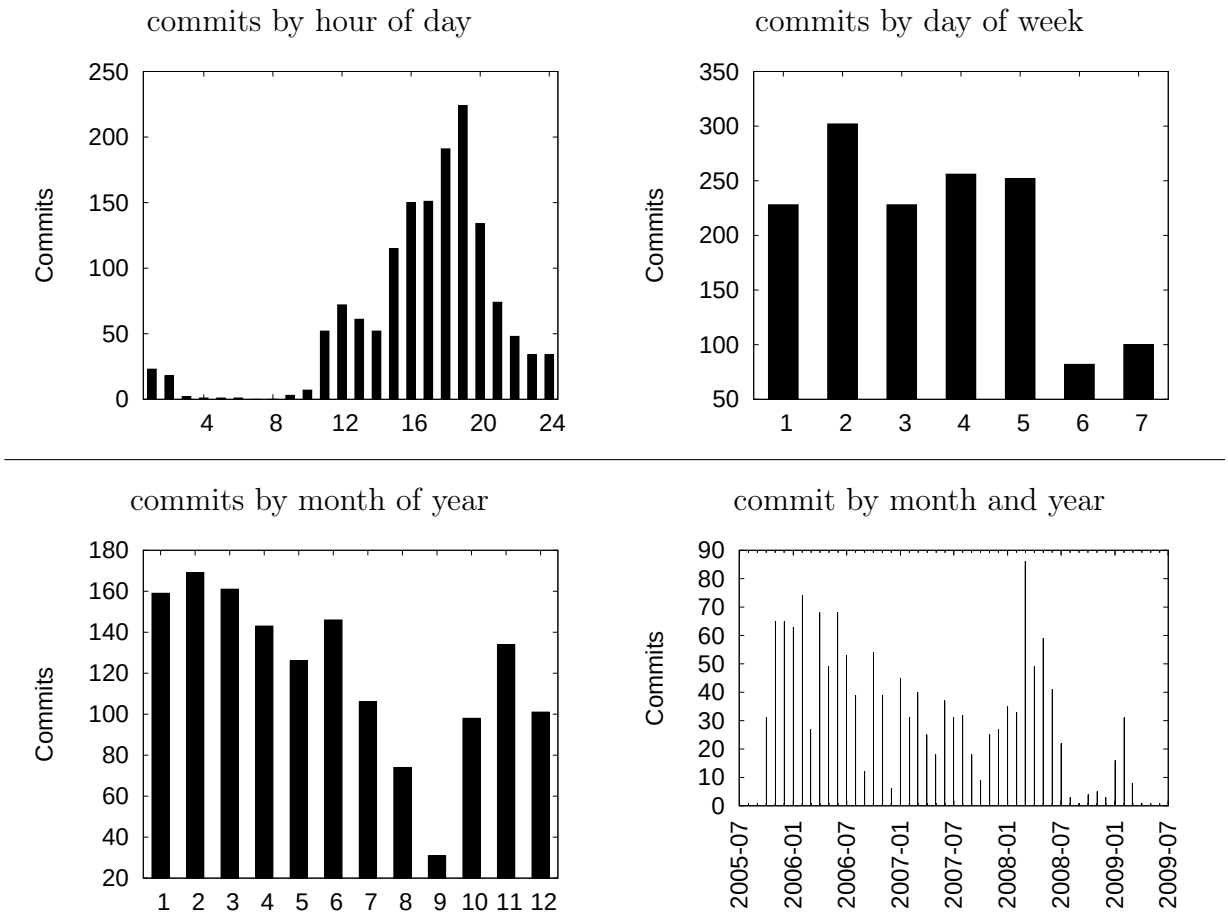
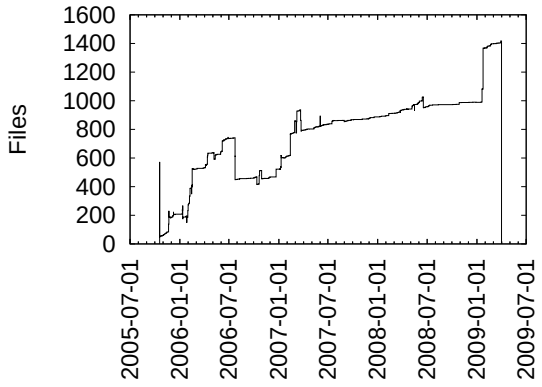


Figure 8.1: Code repository activity log. Shown are various statistics about the number of commits into my code repository.



File-type	#files (%)	#lines (%)	#lines/file
m	884 (62.39%)	95530 (15.27%)	108
cpp	31 (2.19%)	8308 (1.33%)	268
cxx	39 (2.75%)	5688 (0.91%)	145
c	23 (1.62%)	8046 (1.29%)	349
h	22 (1.55%)	2516 (0.40%)	114
hpp	6 (0.42%)	2236 (0.36%)	372
py	87 (6.14%)	61432 (9.82%)	706
sh	19 (1.34%)	825 (0.13%)	43
awk	1 (0.07%)	20 (0.00%)	20

Figure 8.2: File statistics. This shows the increase of the total number of files in the code repository on the left and the number of files as well as their size for selected file-types on the right.

- Total Lines of Code: `625565`
- Total Commits: `1448`

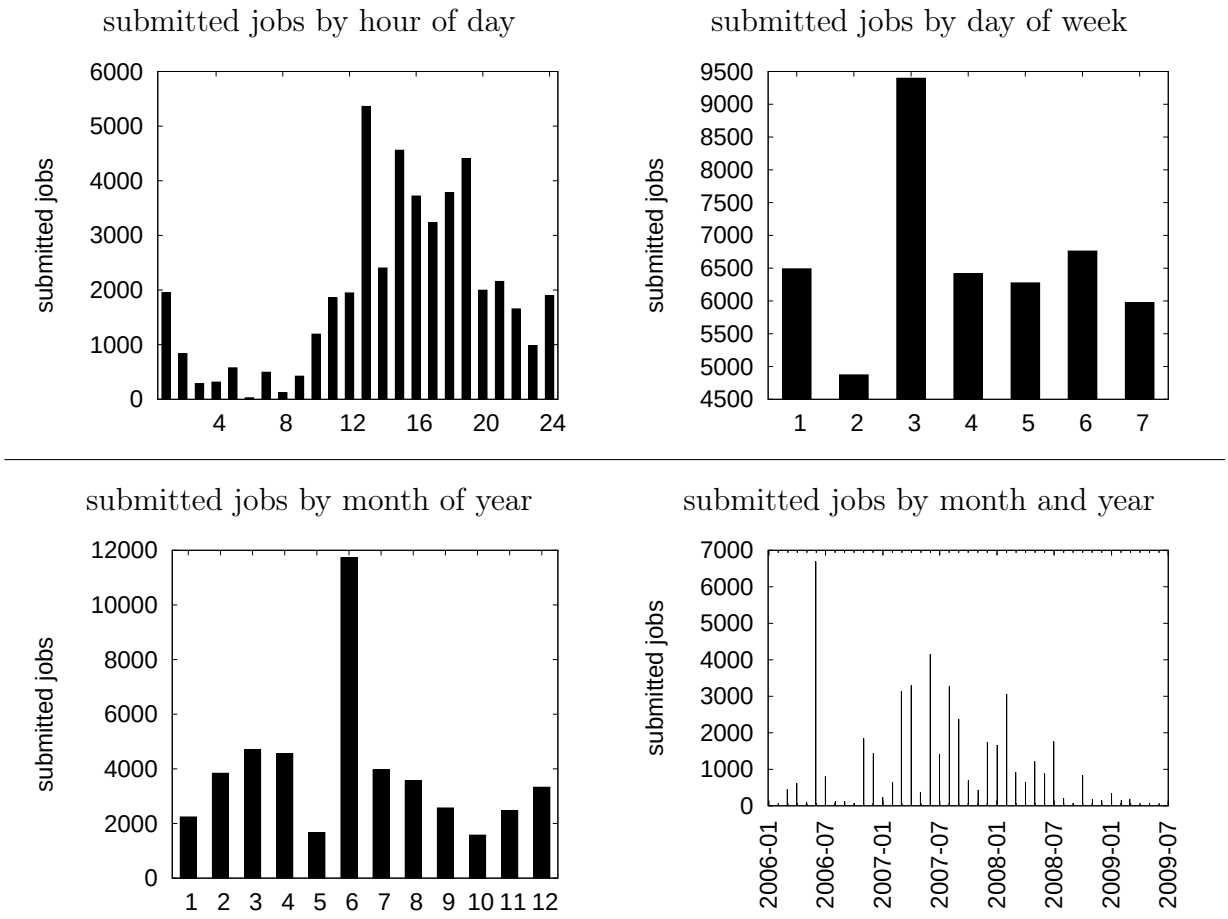


Figure 8.3: Cluster job submissions. Shown are various statistics about the number of submitted jobs to one of the clusters (titus, adam).

8.2.2 Job Statistics

Figure 8.3 gives an idea of when jobs were submitted to one of the clusters:

- Report Period: `2006-03-02 17:03:40` to `2009-04-01 14:04:14` (`1125` days)
- Total number of jobs submitted: `46190`

Bibliography

S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 113–130, 2002. 22

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997. 33, 38

S. Arya, D. M. Mount, Silverman R. Netanyahu, N. S., and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, 1998. 98

J. Aslam and M. Montague. Models for metasearch. In *Special Interest Group on Information Retrieval*, pages 276–284, New York, NY, USA, 2001. ACM Press. 140

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality and the smo algorithm. In *Proceedings of the 21th International Conference on Machine Learning, Alberta, Canada*, 2004. 32

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999. 23

K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, February 2003. 116

A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina*, pages 774–781, 2000. 14

S. Belongie and J. Malik. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24), 2002. 16

T. Berg. Animals on the web dataset, 2006. URL http://www.tamaraberg.com/animalDataset/index.html. 142, 145, 157

T. Berg, A. Berg, J. Edwards, M. Mair, R. White, Y. Teh, E. Learned-Miller, and D. Forsyth. Names and Faces in the News. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, 2004. 116

T. L. Berg and D. A. Forsyth. Animals on the web. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 11, 114, 116, 117, 129, 136, 141, 142, 144, 145, 147, 149

S. Beucher. The watershed transformation applied to image segmentation. In *Scanning Microscopy International*, pages 299–314, 1992. 86

G. Biau, L. Devroye, and G. Lugosi. Consistency of Random Forests and Other Averaging Classifiers. *Journal of Machine Learning Research*, 9:2015–2033, September 2008. 38, 39

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. 22, 30, 32, 39, 41, 43

M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, pages 2–15, Berlin, Heidelberg, 2008. Springer-Verlag. 32

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. 41, 116, 144

O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 91

E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 109–124, 2002. 48

E. Borenstein and S. Ullman. Learning to segment. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*, volume 3, pages 315–328, 2004. 48

E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Computer Vision and Pattern Recognition Workshop*, volume 4, page 46, 2004. 48

A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *Proceedings of the 9th European Conference on Computer Vision, Graz, Austria*, 2006. 25

A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the International Conference on Image and Video Retrieval*, 2007a. 29

A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007b. 33, 95, 101, 113

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2006. 158

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. 42, 44

Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, volume 2, pages 105–112, 2001. 46, 108

L. Breiman. Random forests. *ML Journal*, 45(1):5–32, 2001. 33, 36, 37, 94

C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999. 31

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. 31

M. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of the 5th European Conference on Computer Vision, Freiburg, Germany*, pages 628–641, 1998. 26

J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 13

H. A. Chipman, E. L. George, and R. E. McCulloch. Bayesian ensemble learning. In *Advances in Neural Information Processing Systems*, 2006. 39

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Information Theory*, 14, 1968. 125

B. Collins, J. Deng, K. Li, and Fei-Fei L. Towards scalable dataset construction: An active learning approach. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 116

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. 87

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, USA, 2001. 43

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 32

T. Cour, S. Yu, and J. Shi. Image Segmentation with Normalized Cuts, 2004. URL `http://www.cis.upenn.edu/~jshi/software/`. 86, 87

A. Criminisi. Microsoft research cambridge object recognition image database. version 1.0, 2004. URL `http://research.microsoft.com/en-us/um/people/antcrim/data_objrec/msrc_objcategimagedatabase_v2.zip`. 50

A. Criminisi, T. Sharp, and A. Blake. GeoS: Geodesic Image Segmentation. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 46

G. Csurka and F. Perronnin. A Simple High Performance Approach to Semantic Segmentation. In *Proceedings of the 19th British Machine Vision Conference, Leeds*, 2008. 22, 47, 58, 107, 110, 155

G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004. 23, 24, 57

O. G. Cula and K. J. Dana. Compact representation of bidirectional texture functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, pages 1041–1047, December 2001. 22

N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, volume 2, pages 886–893, 2005. 19, 45, 48, 103, 130

S. C. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information, Science*, 41(6): 391–407, 1990. 40

T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal. Incorporating On-demand Stereo for Real-Time Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 33

L. Devroye, L. Györfy, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag New York Inc., 1996. 25

T. G. Dietterich and D. Fisher. An experimental comparison of three methods for constructing ensembles of decision trees. In *Bagging, boosting, and randomization. Machine Learning*, pages 139–157, 2000. 33

G. Dorkó and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, 2003. 136

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2001. 30, 65

S. Eguchi and J. Copas. Interpreting Kullback-Leibler Divergence with the Neyman-Pearson Lemma, 2005. URL http://www.ism.ac.jp/~eguchi/pdf/KL_NP.pdf. 66

C. Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning, Washington DC, USA*, pages 147–153, 2003. 21

M. Everingham, L. Van Gool, C. K. I. Williams, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2006 (VOC2006), 2006. URL http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006. 14, 24, 44, 50, 83, 155

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007. URL http://www.pascal-network.org/challenges/VOC/voc2007. 8, 44, 50, 53, 58, 83, 108, 110, 155

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results, 2008. URL http://www.pascal-network.org/challenges/VOC/voc2008. 7, 8, 44, 50, 53, 58, 83, 155

L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, June 2005. 25

L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 1134–1141, October 2003. 26

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina*, volume 2, pages 2066–2073, 2000. 27

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004. 86, 87

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, volume 2, pages 264–271, June 2003. 26, 27

R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004. 115

R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005a. 115, 116, 119, 130, 141, 142, 146, 149, 157

R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005b. 27, 49

P.-E. Forssen and D. G. Lowe. Shape descriptors for maximally stable extremal regions. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007. 16

C. Frankel, M. J. Swain, and Athitsos V. Webseer: An image search engine for the world wide web. Technical report, University of Chicago, 1997. 123

W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991. 16

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings*, 1996. 39

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000. 44, 47

J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977. 99

M. Fritz and B. Schiele. Decomposition, discovery and detection of visual categories using topic models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 130

M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating Representative and Discriminant Models for Object Category Detection. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 28

D. M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proceedings of the International Conference on Pattern Recognition*, 1998. 17

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6): 721–741, November 1984. 42

B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A, texture classification example. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 456–463, October 2003. 86, 87

K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 29

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL `http://authors.library.caltech.edu/7694`. 7

C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988. 13

X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, 2004. 42, 45, 156

A. O. Hero, B. Ma, O. Michel, and J. D. Gorman. Alpha-Divergence for Classification, Indexing and Retrieval (Revised). Technical Report, 2002. 65

G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. Technical Report, 2000. 38

G. E. Hinton. Products of Experts. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999. 37, 38

T. K. Ho. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998. 37, 39

T. K. Ho and E. M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, page 880, Washington, DC, USA, 1996. IEEE Computer Society. 38, 39

T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 43:177–196, 2001. 25, 40, 80, 116, 144

H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 7

Y. Jin and D. Geman. Context and Hierarchy in a Probabilistic Image Model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 7

T. Joachims. SVM$^{\text{light}}$. URL `http://svmlight.joachims.org/`. 131

F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, volume 1, pages 604–610. IEEE Computer Society, 2005. 17, 21

T. Kadir and M. Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001. 15

T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004. 13, 15, 130

Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, June 2004. 16

J. T. Kent and K. V. Mardia. Spatial classification using fuzzy membership models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):659–671, 1988. 77

E. M. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. 39

E. M. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence*, pages 207–239, 1990. 38, 39

P. Kohli, M. P. Kumar, and Torr P. H. S. $\mathcal{P}^3$ & Beyond: Efficiently Solving Energies with Higher Order Cliques. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 44

P. Kohli, L. Ladický, and Torr P. H. S. Robust Higher Order Potentials for Enforcing Label Consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 44, 47

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. Technical Report, 2005. 43, 109

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 44

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proceedings of the 15th British Machine Vision Conference, Kingston*, 2004. 27

M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, volume 1, pages 18–25, 2005. 47

S. Kumar and M. Hebert. A Hierarchical Field Framework for Unified Context-Based Classification. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 45, 156

J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning, Williamstown, USA*, 2001. 42

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988. 43

N. Lawrence and B. Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In *Proceedings of the 18th International Conference on Machine Learning, Williamstown, USA*, 2001. 31

S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, volume 2, pages 319–324, June 2003. 16

S. Lazebnik, C. Schmid, and J Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 24, 29, 155

D. D. Lee and S. H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999. 40

B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *Proceedings of the 14th British Machine Vision Conference, Norwich*, volume 2, pages 264–271, 2003. 17, 21, 22, 27, 28

B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, May 2004. 27, 28, 47, 48, 49

V. Lempitsky, A. Blake, and C. Rother. Image Segmentation by Branch-and-Mincut. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 48

M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond Local Appearance: Category Recognition from Pairwise Interactions of Simple Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 28

V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 33, 35, 94, 95, 97, 103, 113

T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, June 2001. 18

T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1010–1017, Kerkyra, Greece, September 1999. 18, 22

J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 116, 157

L. J. Li, R. Socher, and L. Fei-Fei. Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 46

S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001. 42

W.-H. Lin, R. Jin, and A. Hauptmann. Web Image Retrieval Re-Ranking with Relevance Model. In *Proceedings of the IADIS International Conference WWW/Internet*, 2003. 115

T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998. 15

T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 7

D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, September 1999. 15, 16, 130

T. Malisiewicz and A. A. Efros. Recognition by association via learning per-exemplar distances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 25

R. Marée, P. Geurts, J. Piater, and L Wehenkel. Random Subwindows for Robust Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005. 33

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the 13th British Machine Vision Conference, Cardiff*, pages 384–393, 2002. 15

K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*. Springer-Verlag, 2002. 15

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, 2003. 16

K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*. Springer-Verlag, May 2004. 130

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005. 14, 15

F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests. In *Advances in Neural Information Processing Systems*, 2006. 33, 36

K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. In *Proceedings of the 16th International Conference on Machine Learning, Bled, Slovenia*, 1999. 32, 131

N. Morsillo, C. Pal, and R. Nelson. Mining the Web for Visual Concepts. International Workshop on Multimedia Data Mining, 2008. 117

D. A. Norman. The Design of Everyday Things. *Basic Books*, 2002. 2

A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11:520–527, 2007. 2

Onix. ONIX Text Retrieval Toolkit. URL http://www.lextek.com/manuals/onix/stopwords1.html. 123

A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the 9th European Conference on Computer Vision, Graz, Austria*, 2006. 17, 28, 47, 48

M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 33

C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, India*, page 555, Washington, DC, USA, 1998. IEEE Computer Society. 19

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 7

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999. 157

M. Porter, R. Boulton, and A. Macfarlane. The English (Porter2) stemming algorithm, 2002. URL http://snowball.tartarus.org/. 123

P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica, T. Tuytelaars, and L. Van Gool. Modeling scenes with local descriptors and latent aspects. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, pages 883–890, 2005. 25

J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. 33, 35

A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007. 45, 156

X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, 2003. 87

L. G. Roberts. Machine perception of three-dimensional solids. *Optical and electro-optical information processing, J. Tippet et al., Ed.*, 1965. 1

C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, volume 23, pages 309–314, New York, NY, USA, 2004. ACM Press. 9, 46, 84

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 44

B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 46

K. Saenko and T. Darrell. Unsupervised Learning of Visual Sense Models for Polysemous Words. In *Advances in Neural Information Processing Systems*, 2008. 140, 156

S. Savarese, Criminisi A., and J. Winn. Discriminative object class models of appearance and shape by correlatons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 29, 94

C. Schmid. Constructing models for content-based image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, volume 2, pages 39–45, 2001. 18

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002. 30, 31

F. Schroff, A. Criminisi, and A. Zisserman. Single-Histogram Class Models for Image Segmentation. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2006. 10

F. Schroff, A. Criminisi, and A. Zisserman. Harvesting Image Databases from the Web. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007. 11

F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *Proceedings of the 19th British Machine Vision Conference, Leeds*, 2008. 10

T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005. 18

E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, volume 1, pages 469–476, 2001. 45

T. Sharp. Implementing Decision Trees and Forests on a GPU. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 33

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. 86, 87

J. Shotton, A. Blake, and R. Cipolla. Contour-Based Learning for Object Detection. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 17, 28, 47, 48

J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of the 9th European Conference on Computer Vision, Graz, Austria*, pages 1–15, 2006. 23, 25, 47, 58, 94, 95, 97, 101, 102, 103, 107, 110, 113

J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 19, 33, 58, 94, 97, 103, 107, 108, 110, 113, 155

A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, volume 1, pages 235–241, 2003. 45

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, volume 2, pages 1470–1477, October 2003. 23, 24

J. Sivic and A. Zisserman. Video Google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 127–144. Springer, 2006. 7

J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 25, 45

A. J. Storkey and C. K. I. Williams. Image modelling with position-encoding dynamic trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 7, 46

R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, June 2008. 44

M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *Proceedings of the 10th European Conference on Computer Vision, Marseille, France*, 2008. 32

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. In *Advances in Neural Information Processing Systems*, 2004. 41, 144

M. E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems*, San Mateo, CA, 2000. Morgan Kaufmann. 157

A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):153–167, 2003. 29, 120

A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, pages 762–769, 2004. 34

I. Tsochantaridis, T. Hofmann, T. Joachims, and Altun Y. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21th International Conference on Machine Learning, Alberta, Canada*, 2004. 32

L. Van Gool, T. Moons, and D. Ungureanu. Affine / photometric invariants for planar intensity patterns. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, UK*, pages 642–651. Springer-Verlag, 1996. 16

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1999. 31

M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, October 2007. 32

M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 3, pages 255–271. Springer-Verlag, May 2002a. 22

M. Varma and A. Zisserman. Classifying materials from images: to cluster or not to cluster? In *Proceedings of the 2nd International Workshop on Texture Analysis and Synthesis, Copenhagen, Denmark*, pages 139–144, May 2002b. 22

M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin*, volume 2, pages 691–698, June 2003. 18, 25, 69

J. Verbeek and B. Triggs. Scene Segmentation with CRFs Learned from Partially Labeled Images. In *Advances in Neural Information Processing Systems*, 2008. 47, 58, 72, 107, 110

S. Vijayanarasimhan and K. Grauman. Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, 2008. 116

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, pages 511–518, 2001. 19, 44, 48, 94, 95

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967. 43

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *40th Allerotn Conference on Communication, Control and Computing, Urbana-Champaign, IL, USA*, 2002. 43

F. Wang. Fuzzy supervised classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 28:194–201, 1990. 77

G. Wang and D. Forsyth. Object image retrieval by exploiting online knowledge resources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska*, pages 1–8, 2008. 117, 156, 157

M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proceedings of the 6th European Conference on Computer Vision, Dublin, Ireland*, pages 18–32, 2000a. 26

M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina*, volume 2, pages 101–108, June 2000b. 26

J. Winn and A. Criminisi. Object Class Recognition at a Glance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 33, 97, 103

J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, pages 756–763, 2005. 29

J. Winn and J. Shotton. The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006. 29

J. Winn, Criminisi, A., and T. Minka. Object Categorization by Learned Universal Visual Dictionary. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005. 18, 21, 23, 26, 50, 57, 58, 63, 68, 69, 74, 77, 80, 103

P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based Classifiers for Bilayer Video Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis*, 2007. 9, 33, 94, 95

J. Zhang, M. Marszalek, S. Lazebnik, and Schmid C. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 2007. 16, 24, 25, 131

L. Zhu, Y. Chen, C. Lin, and A.L. Yuille. Rapid Inference on a novel AND/OR graph: Detection, Segmentation and Parsing of Articulated Deformable Objects in Cluttered Backgrounds. In *Advances in Neural Information Processing Systems*, 2007. 7, 46

Index