Detecting Domestic Objects with Ensembles of View-tuned Support Vector Machine Cascades Trained on Web Images

Marco Kortkamp

Detecting Domestic Objects with Ensembles of View-tuned Support Vector Machine Cascades Trained on Web Images

Der Technischen Fakultät der Universität Bielefeld zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

vorgelegt von

Marco Kortkamp

geboren am 28.06.1983 in Ibbenbüren.

Bielefeld, Novermber 2011

verteidigt am 12. November 2012

Gutachter:
PD Dr. Sven Wachsmuth
Prof. Dr. Ulrich Rückert

Prüfungsausschuss:
PD Dr. Sven Wachsmuth
Prof. Dr. Ulrich Rückert
Prof. Dr. Barbara Hammer
Dr. Hendrik Koesling

Ausdruck der vom Prüfungsausschuss genehmigten Fassung

gedruckt auf alterungsbeständigem Papier nach ISO 9706

Abstract

This thesis develops a system, based on Web images, for the detection of domestic objects in images of indoor home environments. Images from ten different domestic object categories (apple, bottle, bowl, cup, handbag, laptop, light switch, potted plant, shoe and toaster) are downloaded and annotated from the Web. This results in complex training sets for each category, which are divided unsupervised into sub-sets according to the extracted principal views. The principal views are also employed to learn class- and view-specific, data-tuned hierarchical tessellations of the 2D image plane. The 2D tessellations are used along with a class-independent data-tuned hierarchical tessellation of a high-dimensional descriptor space to realize a view-tuned approximate partial matching kernel. A view-tuned kernel implements a fine-to-coarse matching of Bag of Words-based object parts, while paying attention to the structure of the object and the relative positions of its parts. Both the tessellation of the image plane and the high-dimensional descriptor space are learned with a hierarchical Growing Neural Gas, the IbTreeGNG. View-tuned kernels are used efficiently with Support Vector Machines in a sliding window approach to train view-tuned experts for the different sub-sets. Finally, the outputs of various experts are fused to determine a final detection result. The proposed system shows a state-of-the-art recognition performance on the image database created, and is able to detect unseen object instances in unknown environments.

Acknowledgments

First of all, I want to thank Sven Wachsmuth for supervising my thesis and for supporting new ideas by encouraging my explorations. I also want to thank Ulrich Rückert, who kindly agreed to be my second reviewer.

In addition, I want to thank Applied Informatics for providing me with the oppotunity to pursue a Ph.D., and for being such a warm and open-minded group. A special thanks goes to all the colleagues I have had the pleasure of working closely with in the last years. Thanks, too, for all the non-technical chats and socializing activities.

Last but not least, I want to thank my family for their unfailing support during my studies and beyond. I am also grateful for the moral support of all my friends, many of whom have known me since childhood.

Contents

Abstract i

Acknowledgments iii

1 Introduction 1
1.1 Motivation 2
1.2 Objective and Contribution 4
1.3 Document Structure 6

2 Related Work 7
2.1 Category Classification 8
2.2 Category Detection 17
2.3 Datasets and Challenges 30
2.4 Relation to this Thesis 33

3 Data Acquisition and Annotation 36
3.1 Category Selection 36
3.2 Web Image Retrieval 38
3.3 Web Image Annotation 41
3.4 Results and Discussion 45

4 Principal View Extraction 50
4.1 Introduction 50
4.2 Approach 51
4.3 Results and Discussion 55

5 View-tuned Kernel 62
5.1 Introduction 62
5.2 Approximate Partial Matching with lbTreeGNGs 64
5.3 View-tuned Approximate Partial Matching 68
5.4 Results and Discussion 77

6 View-tuned Support Vector Machines 79
6.1 SVM Formulation and Optimisation 81
6.2 Efficient Feature Extraction 85
6.3 Efficient Computation of the Decision Function 87
6.4 View-tuned SVM Cascade 90
6.5 Sequential Discriminative Optimisation 94
6.6 Summary 98

7 Detection with Ensembles of View-tuned SVM Cascades 100
7.1 Introduction 100
7.2 Classifier Ensembles for Detection 103
7.3 Results & Discussion 110

8 Final Conclusion and Outlook 126

A A Limited Branching Tree Growing Neural Gas 129
A.1 Introduction 129
A.2 Approach 131
A.3 Results and Discussion 134

B Supplementary Material 144

Bibliography 174

Chapter 1

Introduction

Category Detection vs. Instance Detection

This thesis deals with the problem of *object/category detection*, where an object in a category should be located in an image (cf. [195],[167]). For instance, consider a detector for the category "cup", as shown in figure 1.1 (left). Here, as a result the cup detector provides rectangular bounding boxes, that tightly enclose the cup objects. The object detection problem differs from the related problem of *instance detection*. In instance detection, it is the goal to learn and detect a specific

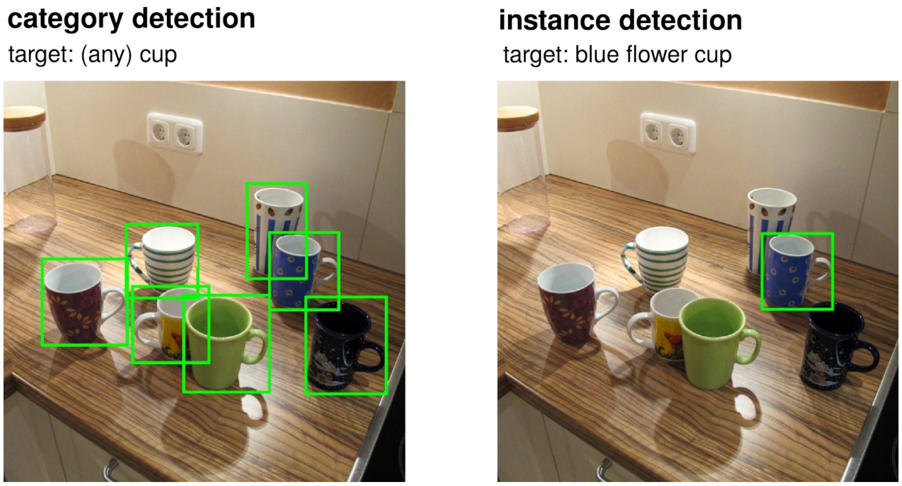


Figure 1.1: (left) Detection results of a detector for the category cup. (right) A detection result of an instance detector for the blue flower cup.

object (cf. [195],[167]), like the “blue flower cup” seen in figure 1.1 (right). This means that the instance detector should only detect the blue flower cup while a cup detector should ultimately detect any cup, no matter what the cup looks like. When I talk about categorisation, it should be understood as being directly appearance-based, because no other semantic or haptic cues are employed. Obviously, category detection is a much harder problem than instance detection. However, both problems are very interesting in the context of mobile service robots, as discussed next.

1.1 Motivation

Motivation for Detecting Domestic Objects

Service robotics is a fast growing field of research (e.g. [3]), where different kinds of ability need to be engineered in order to create systems that interact smoothly with humans and perform tasks in regular home environments. For example, those abilities subsume safe navigation (e.g. [233]), the understanding of dialog (e.g. [159]), the recognition of interaction partners (e.g. [10]) and also the recognition of domestic objects (e.g. [135]). In this context, I use the term *domestic objects* to refer to any objects which can be found in regular home environments and which are used by humans in daily living. In addition, the considered objects should be manipulatable by a robot in some way. For example, a light switch can be turned on or a bottle can be carried.

In the future, when a service robot like Biron/ ToBI (e.g. [86], [218]) is delivered to an apartment, the robot should have a basic knowledge about the world, but should also be able to learn new specific things in its novel environment. With respect to the robot’s object recognition capabilities, this means the following. On the one hand, the robot should be able to learn to detect the blue flower cup of figure 1.1, since it is specific in a new environment and most likely the robot hasn’t seen it before. This instance detection capability allows a user to delegate tasks w.r.t. a specific object instance, e.g.: “Find my blue flower cup!” On the other hand, the ability to detect objects in a category is also very useful in a range of different scenarios:

- **Category Level Tasks.** In some cases, the human does not want to tell the robot what specific object should be used in a specific task. For instance, when giving commands like "Get me a cup of coffee!" or "Clean the cups on the dining table!"
- **Inference.** Detecting an object in a category can allow the robot to infer the function or manipulation possibilities of a novel object, i.e. what can be done with the object.
- **Exploration.** When the robot is new to an environment it could try to detect objects, for instance during an home tour (e.g. [196]) or in an autonomous exploration (e.g. [226]). Subsequently, the robot could initiate a dialog with the human in order to clarify uncertain detection results, as stated next.
- **Robot Initiative.** Here, the basic idea is that the category detection system will never be perfect and produces errors. However, to deal with uncertain stimuli the robot could ask the human in selective situations to clarify things, similar to [158]. For instance, while presenting an image on the screen the robot could ask "I have seen this object today and I think it is one of your cups. Is this true?"
- **Environmental Adaption.** When a robot takes initiative and asks for clarification, it's possible that the robot found an important specific object instance. For example, the robot found the blue flower cup of figure 1.1 and it should remember this specific cup, because it is my favorite cup. In this scenario, category detection can be used in order to initiate the learning of instance detection.

From this small set of examples it can be seen that a working detection system for domestic objects is a basic building block of future research in service robotics. But an important question is on what basis should this basic detection ability for domestic objects be trained and tested?

Motivation for Training and Testing on Web Images

In this work I follow the de-facto standard for acquiring training and testing samples for category detection systems, i.e. by employing images from the Web (e.g. [45], [197], [39], [48], [204], [82], [203]). The fact that employing Web images is the most common method to acquire samples for training and testing of object detectors results from the following. First of all, it is easy to acquire a large number of images by using textual queries. Further, the objects found in Web images are very diverse w.r.t. their appearance and pose. Also the background of the objects is very diverse. Beyond that, domestic objects can often also be found in product images, where they had been photographed in front of a homogeneous background, and which may provide additional cues for learning.

Besides, I think that testing the generalisation abilities of a category detector with robots in real environments is very costly. In order to make a reasonable statement about the generalisation abilities of the system, it would be necessary to place hundreds or thousands of objects in different environments. This is usually not feasible, e.g. when testing a detector for TVs. And I think the same argument of intractability holds for the training phase of the system. It is usually not possible to present to a robot thousands of different objects (e.g. TVs) in different environments and poses. However, the good news is, by using Web images the desired diversity and quantity of training and test images can be provided at a feasible cost. In the next section, the scope of the work is specified and the contributions of the thesis are stated.

1.2 Objective and Contribution

Objective

Building an object detection system is a lot of work, e.g. compare [54]. But the integration of a detection system in the architecture and behavior of a specific robot is also a lot of work, e.g. compare [135]. Certainly, both aspects are too much to handle within a single thesis. Therefore, this thesis focuses on the first part:

> The goal of my thesis is the development of a system which is able to detect domestic objects in images on the basis of Web images.

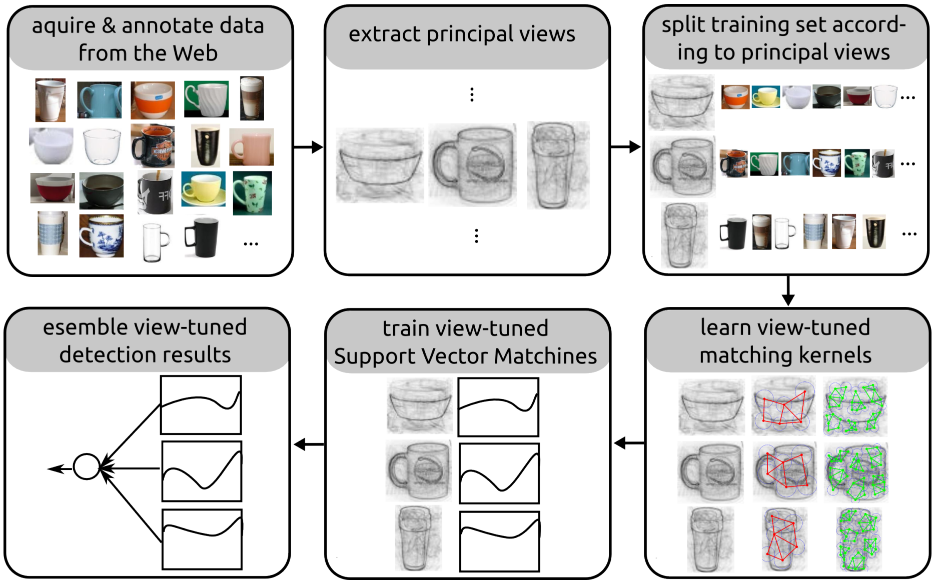

aquire & annotate data from the Web
extract principal views
split training set according to principal views
learn view-tuned matching kernels
train view-tuned Support Vector Matchines
esemble view-tuned detection results

Figure 1.2: This figure sketches the main stages of the detection system with view-tuned SVM ensembles.

In this context, I introduce the following additional constraints to the recognition system in order to specify the objective of this work. First, the detection system should not depend on any robot-specific processes or functionalities. The detector should be trained and evaluated as a self-contained component. Second, the detector should operate locally on single low-resolution (i.e. 640×480) color $2D$ images and should not require any additional context information like the 3D structure or a memory of detection results over time. These constraints make the detection system more flexible and reusable, since it avoids coupling to other components.

Contribution

The overall contribution of my thesis is the development of a novel object detection system for domestic objects, which is trained on Web images and which yields state-of-the-art results. A sketch of the different stages of the detection system is given in figure 1.2. On a more fine granular level, the contribution is given as follows. First of all, a large novel database of ten different domestic object categories is created as part of this thesis. Further, this thesis proposes an approach to split complex training sets unsupervised according to “principal views”. In addition, this work contributes a method for learning view-tuned matching kernels for the sub-sets, which realize a tuned fine-to-coarse comparison of the samples in a set. Moreover, this thesis proposes methodology for applying view-tuned kernels very efficiently in the framework of Support Vector Machines (SVMs). Also, an approach for esembling the detection results of different view-tuned experts is presented. The proposed procedures are evaluated within the different stages of the system, but also w.r.t. the total detection performance of the system. Beyond that, a robust and efficient method for constructive online one-shot learning of hierarchical vector quantisation is contributed. The method is used as a basic ingredient in several stages of the system.

1.3 Document Structure

The document is structured linearly along the different stages of the detection system, as outlined in the last section. For that purpose, the details of the hierarchical vector quantisation approach are explained self-contained in the appendix A. In chapter 2, an overview of related work in the fields of object classification and object detection is provided. Other related work is also discussed locally within some chapters. In chapter 3, a number of 10 domestic categories is selected. Then, the process of data acquisition and annotation is discussed and the results are presented. Afterwards, the method for extracting principal views and splitting training sets, as well as the results are discussed in chapter 4. Subsequently, chapter 5 explains the learning of view-tuned approximate partial matching kernel and presents the results. Further on, chapter 6 explicates the distinctive training of view-tuned SVMs. Also an efficient computation scheme for the decision function is presented in the same chapter. After that, in chapter 7 the ensembling of view-tuned experts for detecting domestic objects is explained and the overall detection system is evaluated. Finally, chapter 8 presents a final conclusion and an outlook to future work.

Chapter 2

Related Work

This chapter provides a brief overview of work related to category detection and presents some of the major directions and recent approaches. For a more comprehensive view on category recognition see [163], [167], [40], [50] and [195]. Note that I also provide directly related work in some specific sections of the thesis.

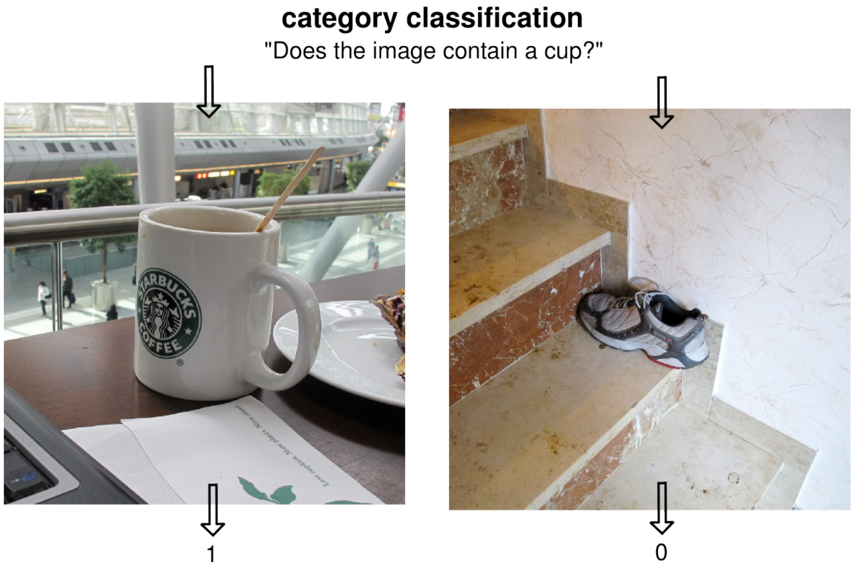


Figure 2.1: Example results of an object classification for the category “cup”.

In chapter 1 the difference between instance detection and category detection has already been explained. To recap, a cup detector localises all occurrences of cups in an image, while an instance detector only localises a specific cup, like a blue cup with flowers. In this chapter, category detection is also distinguished from *object/category classification*. A category classifier determines whether any instance of a category is present in an image or not. The classification does not provide any information about the location of the instances. For example, a cup classifier determines whether any cup is present in an image and can return a binary answer or a confidence score. Consider figure 2.1 for an example. Of course, a category classifier can be used to build a category detector, e.g. when evaluating a classifier at local image regions [110]. Also note, that category classification is quite similar to scene recognition, where the question is, whether an image shows an instance of a scene category (e.g. "Does the image show a kitchen?").

Since classification and detection are linked and detection ideas are inspired by classification techniques, the main directions in category classification are discussed first in section 2.1. Subsequently, category detection is reviewed in section 2.2. Finally, section 2.3 presents major datasets and recognition competitions in the area of category recognition.

2.1 Category Classification

This chapter presents some related category classification approaches, distinguished according to the following principles: Bag of Words (BOW), Visual Words with Spatial Location, Part-based Models and other methods. Note that the approaches presented are organised roughly under these headings. It is not intended to be fully correct, as some approaches use multiple or modified principles and do not allow a clear cut distinction.

Bag of Words (BOW)

The bag of words (BOW) technique is known from text mining (e.g. [122]), where a document is represented by a histogram of word counts over a dictionary. In doing so, the order of the words in the text gets lost, but tasks like classification can still be performed reasonably well (e.g. [12]). BOW has also widely been applied in category classification,

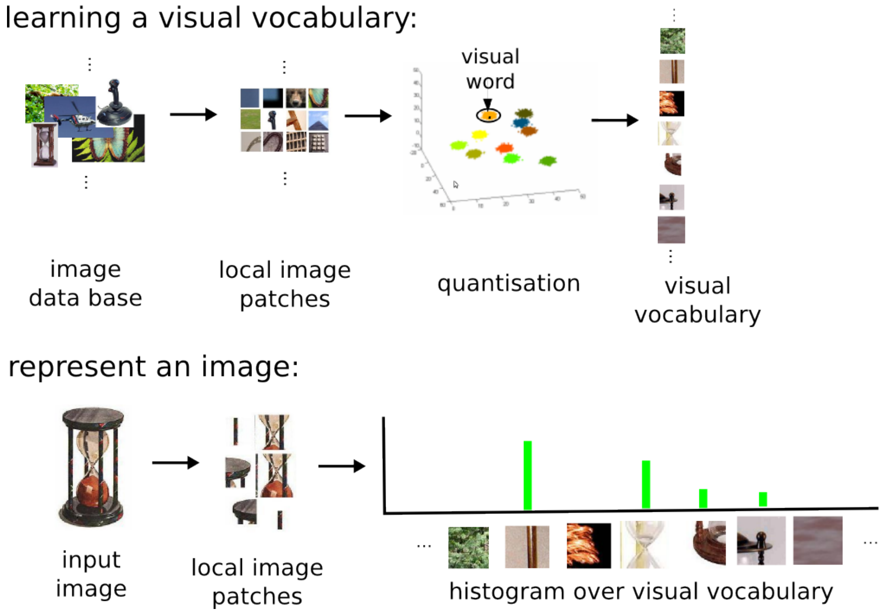


Figure 2.2: Sketch of the bag of words approach.

where an image is represented by a histogram over clustered descriptors of local image regions. See figure 2.2 for a sketch of the principle. Some approaches that use variants of this technique are given in the following.

Early versions of BOW-like approaches are mostly concerned with the recognition of texture, e.g. [120] and [112]. An early application of the mentioned text mining approach in computer vision is e.g. [190], where an object matching for searching video frames is proposed. Also using BOW representations, [36] compares SVM with naive Bayes classifiers and concludes that SVMs clearly show a better performance. When using a BOW scheme one question is how to get good codebooks. The publication [99] discusses different quantization methods to create efficient codebooks. The work of [160] discusses the idea of learning class-specific codebooks and representing an image by a set of histograms (one for each class). In another way, the authors in [225], [228], and [143] use class-specific information in the construction of visual vocabularies in order to enhance the distinctiveness of the visual words. Another direction of creating codebooks is explored in [231], where a set of vocabulary trees is incrementally learned in order to enhance indexing and recognition performances. Beyond that, the efficient use of hierarchical codebook trees [148],[104] and variants of randomised forests [161], [143], [188] are investigated, in different contexts like large instance recognition or image classification. An efficient way to compute an SVM kernel by matching unordered feature sets is the *pyramid match* as introduced in [78] and [81]. The authors propose to represent the feature sets as multi-resolution histograms and to use a weighted histogram intersection to get an efficient Mercer Kernel. This pyramid match has been employed in [79] to automatically learn category models from partially matching sets of local visual features. Various other kernels are also explored for their performance in classification tasks, e.g. [237] evaluates different kernels to compare the probability distribution of word count histograms. The work of [5] is an example, where bag of words is employed to learn a few discriminative similarity metrics from training data for image categorisation. The BOW technique is also used in classification with local neighbors: [28] proposes an improved learning of local distance functions for a better classification of categories. It is also possible to use their work for object categorisation approaches with a localised kernel, e.g. [236]. Seeing the learning of local distance functions as an approximation of the geodesic distance of a metric tensor, the paper [170] proposes a taxonomy and comparison of such learning methods. Besides, BOW is employed as part of probabilistic models: In [51], the authors propose to learn a Bayesian hierarchical model for classifying categories of natural scenes. Also a generative model is proposed in [48]. Here, the method is used to incrementally train a Bayesian models and it is evaluated on 101 object categories. In [189], the authors use two types of generative statistical model, that are known from text mining with BOW representations. They show that it is possible to discover categories in collections of images unsupervised using the methods on bags of visual features. Also hybrid learning approaches based on visual words have been proposed. For example, [94] explores the combination of generative with discriminative methods for object recognition by using Fisher Kernels. Another type of combination, i.e. the combination of visual and textual cues and slight supervision, is employed in the work of [9] to produce large visual datasets of animals from the Web. Fully automatic harvesting of image databases from the Web is also studied in [183]. Here, the proposed method makes use of visual and textual cues for classification, as well. Also using datasets from the Web, the work in [124] proposes an incremental learning scheme for a probabilistic graphical model in order to learn from the datasets with minimal human effort. In the same manner, [214] employs multiple-instance learning on bags of Web images to learn categories from weakly supervised data. Besides, [31] also employs BOW features to actively learn object categories with minimal supervision from Web image search results. In [80], the mentioned pyramid match has been extended by using a hierarchy of non-uniformly spaced bins in the feature space. The authors propose to compare sets of features of arbitrary size with a *vocabulary guided pyramid match*, where non-uniform bins allow a better approximation of the optimal correspondence. An evaluation of different BOW setups for scene category classification is studied in [227]. The authors discuss how different design choices, like the dimensionality and the weighting of visual words, affect the overall classification performance.

A large variety of feature detectors and feature descriptors can be used with the bag of words technique. For example, a common choice is the usage of the difference of Gaussians keypoint detector together with a SIFT [127] descriptor. Aside from that, other examples for keypoint detection methods are Kadir's saliency operator [100], Multiscale-Harris [139], MSER [133] and SURF [7]. For a comprehensive study of interest point detectors see [181], [137], [140] and [208]. As well, some works leave out the detection step and place the local feature on a regular grid over the image (e.g. [51]). Some additional examples for local descriptors are GLOH [138], HOG [37], PCA-SIFT [103] and spin images [111]. For a performance evaluation of different local descriptors confer [138]. Also colored versions of local features have been proposed - e.g. see [210] and [72] - or for a comparison of methods see [19] and [209]. Beyond that, researchers have studied techniques to reduce the dimensionality of the descriptors for the purpose of a more efficient matching, e.g. [96], [136] and [23]. But not only the dimensionality is also an interesting property, the informativeness of local features is also studied, e.g. in [213]. Note, that also the combination of different kinds of features is intensively studied, e.g. [76], [154].

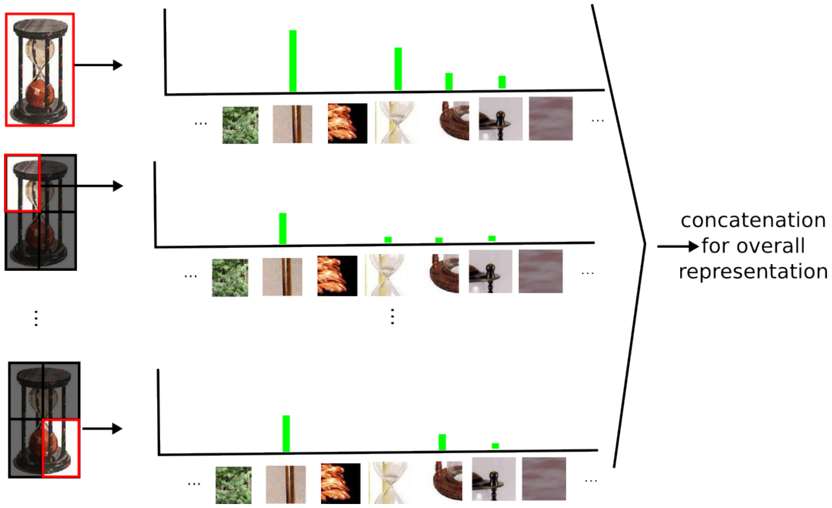


Figure 2.3: This figures sketches one possible way to add spatial information to visual words. In this way BOW histograms are computed for different image regions and fused to an overall representation.

Visual Words with Spatial Location

One problem of the BOW representation is that information about the positions of the local descriptors gets lost. There are several extensions to overcome this limitation and to add spatial information to visual words. One way to encode spatial information is by using a (hierarchical) spatial binning. Here, each bin can still be represented by a BOW, but through a concatenation of the feature vectors, the overall representation encodes information about the location of the features to a certain extent. This idea is sketched in figure 2.3. But also the direct incorporation of location information into the features or the models is possible. Some instances of these approaches are presented next.

For the purpose of scene categorisation, the authors of [113] extend the BOW method as a spatial pyramid histogram and propose a pyramid match kernel to compare images. They show, that an incorporation of the spatial information does significantly improve the results of classifiers compared to global BOW representations. Also [14] proposes to use a spatial pyramid to implement matching with a pyramid kernel. The authors learn the weights of the levels and combine shape and appearance kernels. A performance gain is demonstrated when using this spatial structure. As well as picking up the idea of [113], the authors of [13] generalise the global representation to regions of interest. Further, they propose an idea to suppress background and use random forests as a multi-way classifier, which reduces the cost of training and testing. In [230], the authors show how the spatial pyramid matching [113] can be employed as a kernel for an incremental learning SVM, applied in the context of interactive categorisation. Another way to introduce spatial information to BOW is by using a weighting scheme: The authors in [132] propose to utilise the object boundaries in order to mitigate weights for background features, while boosting the weights of object features. In doing so, they are able to estimate approximate segmentation masks and thus to perform object detection. To get a better test time complexity than typical BOW SVM systems, [232] proposes to store an inverted index lookup from visual words to the training images and perform a SVM based voting for the final classification. The authors demonstrate improvements in speed and scalability. In [192], [193] and [219], different ways of introducing spatial relations in hierarchical graphical models are explored. This realises a hierarchical model of scenes or objects with spatial parts. Another direction is to incorporate location information on the feature level, as explored in [180]. Besides, the authors of [128] proposes to represent images of a category as 2D sequences of visual words. The authors introduce a spatial mismatch kernel, that captures the spatial structure of the image for classification of natural and historical images with kernel machines.

Part-based Models

A part-based model represents objects by parts, which have a certain geometric relationship and appearance. For instance, a face can be seen as consisting of several parts: eyes, a nose, a mouse and so forth. And each of these parts has a certain appearance, but also a certain relative location - for example the nose is over the mouth and under the eyes. See figure 2.4 for a basic sketch of this type of model. The idea of recognising objects by geometrically constrained parts has a quite long history, e.g. [101], [68], [234] and [18]. Since part-based models can be

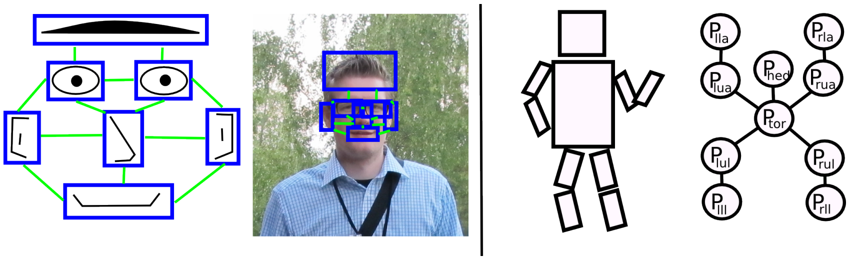


Figure 2.4: (left) Sketch of a part-based model for faces (right) Sketch of the Pictorial Structure Model [55] for human detection

used for both category classification and detection, they are discussed here. However, some additional remarks are also given in the detection section. According to the authors in [50], basic questions for part-based models are:

- How is appearance represented?
- How are geometric relationships modeled?
- How can learning and recognition be realised with such representations?

The parts themselves can be considered in a sparse or dense manner (pixels vs. regions). Using sparse parts avoids a modeling of the global variability and is more tractable than dense representations. However, while throwing away a large amount of visual information, the sparse regions must be distinctive in order to allow separation of different categories. When using a part-based model to recognise an object, the parts of the model need to be aligned to the picture. This is called the correspondence problem. In general, solving the correspondence problem can be very costly. For instance, given a model with P parts and M possible candidate regions in the image, there are P^M possible combinations for a one to one mapping. Thus, researchers have introduced various ways to reduce the complexity of learning and inference methods.

In the last decade, various probabilistic models have been proposed for realising the part-based notion. Thereby, geometric spatial priors are proposed in different topologies and result in different inference complexities. In [25] some prior topologies and their complexities are compared. The previously discussed bag of words approach can also be seen as a graphical model in which the geometry of parts is independent given the model [36], [25]. This model can be trained fast, but it doesn't pay attention to the relative positions of parts. The extreme in the other direction is given by the constellation model, which models the full joint distribution of location of parts [60].This model is able to capture a lot of information, but the learning becomes intractable quite quickly, as the number of parts goes up (exponentially growth of parameters). There are various approaches between these two extremes. The general idea is to restrict the topology of the geometric prior in order to allow more efficient learning and inference. The star shaped model [34], [62] and hierarchical models [56], [16] are examples of this strategy. All these models contain a special node, which facilitates more efficient inference (e.g. the center node of the star model). In embedded hierarchical models [16] up to hundreds of parts can be used. The geometric prior proposed in [34] has a less restricted geometry where each model component depends on the location of the k nearest neighbors. In [56] and [55] also distance transformations are used to further increase the efficiency of the probabilistic inference by avoiding an exhaustive search for the positions of parts. Later on, [57] and [54] efficiently make use of a star-graph (or 1-fan) model to discriminatively train multi-scale deformable part models (DPMs). The authors demonstrate very good results on challenging data (like the PASCAL dataset [45]) by using HOG features [37] and a latent SVM formulation with a data mining approach for finding hard negative examples. Here, a coarse global filter and high resolution part filters are used in a multi-scale pyramid with a sliding window. The spatial relationship is captured by a deformation cost for the parts. The method is among the best state-of-the-art algorithms and the fact that the authors made their implementation public renders it very suitable for comparing novel detectors against the state-of-the-art. The same method has since then been improved w.r.t. efficiency. In [53], the authors introduce the notion of cascades for DPMs and demonstrate massive speedups without altering precision or recall. Other work of [116] implicitly realises a star-shaped geometric prior by using a visual word based probabilistic voting model for the center of an object. The mentioned constellation model is originally proposed in [20]. This technique is also extended for unsupervised learning [222]. A further extension is described in [63], where appearance and shape are simultaneously learned. An extension of this constellation model was used in [61] to re-rank the results of Google's image search. Also by using Google's Image Search, [59] showed that Web images can be used directly in order to train category classifiers, here with an extension of the probabilistic latent semantic analysis (pLSA) model. Also learning categories from a small number of samples is studied with part-based models: e.g. [49] takes advantage of the knowledge gained from previously learned categories. In [8] object classification is studied in a deformable shape matching framework. Here, the correspondence problem is solved as an integer quadratic programming problem. Besides, the work of [119] encodes the relations between local feature as cliques of interconnected parts. Using this representation, the authors demonstrate the ability to localise and classify object categories. In hierarchical representations, objects not only consist of parts, but the parts can also be considered as several layers of "groupings" starting from pixels. Sophisticated hierarchical part-based models can be realised with the notion of stochastic grammars (e.g. [16], [239]). Another type of hierarchical representation is is proposed in [66] and [67]. Here, the authors propose a cross-layered compositional representation for category classification. The layered network encodes the notion of scales and allows feature sharing between classes. The approach in [151] decomposes objects unsupervised into a hierarchy of parts and learns a composed representation. The compositions allow the sharing of features and are combined in a graphical model.

Other Related Methods

There are also some methods which don't really fit under the headings used here. For instance, object classification approaches that are biologically inspired or employ neural networks for learning features. In [185] novel features are introduced which are inspired by the visual cortex. Their recognition system demonstrates good performance, even when learning from a small number of examples. Besides, [145] uses sparse features with limited receptive fields to recognise and localise objects in images. On a common image dataset, the algorithm shows competitive performance, even when detecting cars in real world scenes. Another approach is taken by [171]. Here, a gated Markov Random Field is used as a generative model of natural images and is taken as the lowest level of a Deep Belief Network (DBN) which includes several hidden layers. The method demonstrates a comparable recognition performance for facial expressions without the need for well-engineered SIFT [127] features. Also [102] proposes to learn feature extractors unsupervised in a filter and pooling framework. This method creates topographic maps of similar filters, which are pooled to an invariant result. The authors show that their features work as well as or even better than SIFT [127] in a recognition scenario.

2.2 Category Detection

This section presents an overview of related category detection work. Also in this chapter, the approaches are structured roughly under a number of headings.

Detection with Visual Words

As stated in the last section, visual words are a common technique used for classification that can also be employed for detection, e.g. by application in a *sliding window framework* [110]. The basic sliding window detection scheme is sketched in figure 2.5. Given a classifier trained on in-class and non-class images, the red window of the size $w \times h$ is moved across the image and the classifier is applied to each local image region. In doing so, an offset of dx pixels is used in the x-direction, and an offset of dy pixels is used in the y-direction. An exhaustive search uses $dx = 1 = dy$. However, to keep the computational complexity tractable, the search space is truncated by setting dx and dy to a larger number of pixels. Then the displacement can be written as a fraction of the window width and height, e.g. $dx = xs \cdot w = .3w$ and $dy = ys \cdot h = .3h$. The green window depicts a position, where the classifier assigned the label “in-class”. To be invariant to the scale

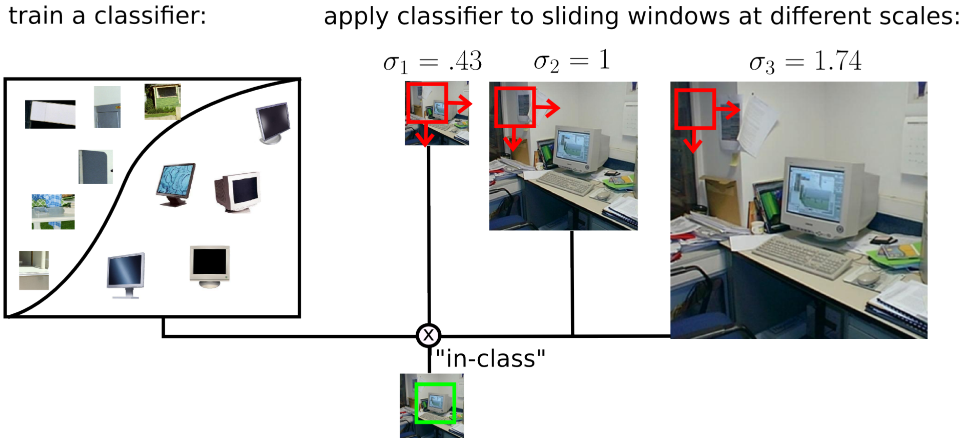


Figure 2.5: Sketch of the sliding window approach for detecting objects with classifiers.

of the object, sliding windows are normally used at different scales $\sigma = \{\sigma_1, \ldots, \sigma_N\}$ of the image. To be applicable in a sliding window approach, the classification method must be efficient, because usually a lot of windows need to be evaluated. Since various visual word based approaches were already mentioned in the last chapter, I only state a few additional examples here.

Building on the principle of boosting, the approach in [152] explores the learning of a discriminative set of descriptors for local regions (weak hypothesis) from weakly supervised data. These weak hypotheses are then combined into a final hypothesis for each class. Thereby, the authors show that the ensembling of multiple extractors and descriptors yields improved performance, without considering the relative location of features. Further, [2] represents an object using clusters of local rectangular regions, sampled from the object images. In the exemplar model learning approach of [30], objects are localised using a sliding window approach with a SVM based on a spatial pyramid kernel with visual words and edge features. The authors demonstrate that their model is scale and translation invariant.

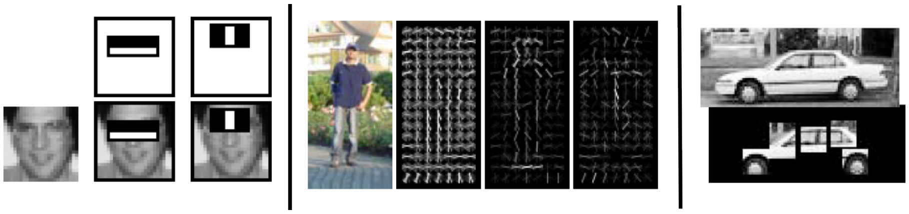


Figure 2.6: (left) This figure by Viola and Jones [215] shows the Haar wavelets chosen for face detection and which can be efficiently computed with integral images. (middle) A visualisation of [37] showing a HOG descriptor for an image and the positive and negative weights learned for human detection. (right) This figure sketches the sparse, part-based representation used for car detection (image by [2]).

Detecting Faces, Pedestrians and Cars

A lot of the early work in object detection is concerned with the problem of face detection, pedestrian detection and car detection. Often, the developed algorithmic principles or features are not restricted to a specific object class and can be adapted to other categories as well.

Early approaches in face recognition worked with good illuminated frontal face images. Later on, more sophisticated recognition setups were considered and included changes in pose, bad illumination and side-views of faces. A large number of face detection approaches have been developed over the years. In [229] a survey on earlier work is provided and in [201] and [195] some more recent work is discussed. According to the authors in [229], face detection approaches can be distinguished accoding to whether they are template-based, feature-based or appearance-based. The most exhaustive way to do face recognition is to apply a classifier at every location and scale in an image, but this is way too slow for real applications [141]. Appearance-based principles make use of this basic sliding window idea, but apply it in a more efficient manner. One way to make this principle more tractable is to use a coarser grid of positions and scales, where the classifier is applied rather than evaluating every possible location and scale. Another idea is to use cascaded structures. Here, efficient and weak classifiers are applied first and complex classifiers are only considered on face-like regions that passed the weak classifiers. The two ideas can also be combined. To be able to detect faces on multiple scales, a sub-octave image pyramid [195] can be constructed, where the scanning is performed at each scale using a fixed sized window. Examples for appearance-based methods are [70], [175] and [215]. The work of Viola and Jones [215] proposes the idea of training a chain of increasingly discriminating weak classifiers and ensembling the output to realise a complex decision boundary. In doing so, they learn distinctive features from a large pool of possible features (compare figure 2.6 (left)). Today, there are even more efficient ways for realising the widely used technique of cascades (e.g. [17]). In another direction, template-based methods are able to deal with big changes made to the pose and the facial expression. One example is the the active appearance model [32]. Often, template-based methods rely on a good initialisation close to a real face and they are not applicable for fast detection of faces. Besides, the feature-based approaches aim at finding the positions of distinctive visual features (e.g. the mouth, the nose, etc.) and try to validate the presence of a face w.r.t. geometrical constraints. Early versions of this part-based principle are e.g. [68], [234] and [101]. Later works employ eigenspaces [141], local filter jets [121], support vector machines [87] and boosting [182]. In very unconstrained situations (e.g. with bad lighting conditions), face detection can still be challenging today. However, in general the problem is solved quite well and developed methods can be found embedded in modern digital cameras. Using face detection, a camera can for instance automatically set the focus on the person being photographed.

Also, the detection of humans (pedestrians) and cars is receiving a lot of attention, e.g. [75], [142], [1] and [37]. Such methods are particularly interesting for surveillance or driver assistance systems. Today, the detection algorithms are applied in the industry, e.g. in embedded systems for driver assistance. The topics addressed in reasearch often focus on either efficiency, accuracy or precision. In [2] and [1] a visual vocabulary is used to represent informative parts of a car. Also the spatial relationships among the parts are captured and a framework for robustly detecting side views of cars is presented (see figure 2.6 (right)). A survey with a special focus on on-road vehicle detection

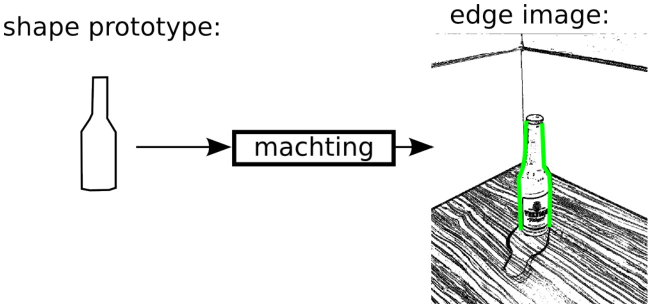


Figure 2.7: This figure depicts object detection as a shape matching problem.

is presented [235]. A well-known detection method for pedestrians is presented in [37]. As a feature the authors proposed a histogram of oriented gradients (HOG) which are classified by a SVM. HOG features are similar to SIFT features [127], as they represent the local gradient structure in an image by a histogram, but HOG is computed at a single scale using a regular overlapping grid with local normalisation. See figure 2.6 (middle) for a visualisation of the HOG feature. A survey for pedestrian detection can be found in [144]. Here, the authors conclude that methods using local receptive fields and SVMs show the best performance.

Detecting Shape

Most methods presented so far consider the shape of an object (the outer edges) in an implicit manner, e.g. the BOW representation. However, shape can also be used more explicitly as it provides a strong cue for recognition performed by humans [156]. The common problem is to robustly match a shape prototype against the edges found in an image, as sketched in figure 2.7.

According to [173], shape based models can be sorted into the following groups:

- learning codebooks of contour fragments (e.g. [153], [187])
- approximating contours by piecewise line segments (e.g. [172], [64])

- using local description of the contour at selected interest points (e.g. [8], [130])
- assigning entire edges to either foreground or background (e.g. [84],[238])

The work of [173] itself detects objects by partially matching edge contours with a single prototype. In this case, no preprocessing of the edges is needed and the notion of "connectedness" is exploited. In [153], category detectors are trained using Boosting on a visual vocabulary of reusable shape fragments and appearance. The resulting descriptors vote for the centroid of an object. The authors demonstrate robust detection results using this boundary fragment model, despite changes in scale and viewpoint. Further, [187] proposes a two-step partially supervised learning process for contour-based object detection. The method learns location sensitive classifiers via boosting and employs a discriminative set of features from a contour fragment dictionary. Besides, the authors of [172] make use of the observation that deformations often appear at high curvature points. Using these points, a curvature is decomposed to several fragments, which are matched via dynamic programming for detection in cluttered real-world scenes. The work in [130] proposes an approach that involves employing the Hough transform together with a max-margin SVM to detect objects in images. The authors demonstrate improved performance through the use of weights learned in their discriminative framework, as opposed to uniform or naive-Bayes weights. Further, [84] introduces a system for detection, classification and segmentation that employs region features. The method creates a robust bag of regions and represents regions through several cues, i.e. color, shape and texture. In doing so, also a max-margin Hough transform followed by a verification is used to provide final detection results. In another approach, [75] proposes to use Chamfer distance matching to compare edges in images to a hierarchy of templates for detecting pedestrians in real time.

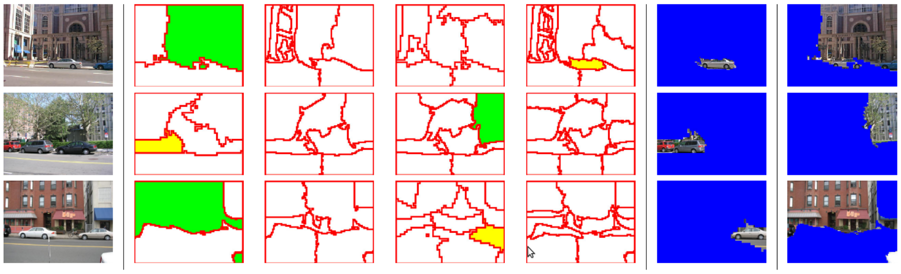


Figure 2.8: This figure of [178] shows different segmentations of an image. The green regions stand for the category "building" and the yellow regions for the category "car". The regions are discovered unsupervised.

Detection and Segmentation

In principle, the more sophisticated version of category detection is category segmentation. Here, the locations of the detected object should be specified more precisely than by a bounding box, e.g. through a binary segmentation mask which distinguishes foreground and background pixels. Hence, when doing a segmentation of an image, this can also be considered as a detection. In the following I will sketch some approaches which combine detection and segmentation.

In [115] the learning of visual vocabularies is combined with the learning of an implicit shape model that captures the relative locations of visual words. By using a voting scheme, an object hypothesis in the image is created, which leads to a segmentation mask for an object and which can again be used as an input, to truncate visual words. The authors demonstrate detection and segmentation capabilities in the presence of significant occlusion as well. Another way to automatically discover objects and their extent in images is proposed in [178]. The algorithm uses multiple segmentations as input to probabilistic document analysis in order to learn the appearance of objects. On the feature level, each region is represented as a BOW over clusters of SIFT [127] descriptors. See figure 2.8 for a sketch of this idea. A probabilistic object model (POM) for object classification, segmentation and recognition is proposed in [29]. The algorithm learns structures by combining complementary features and employs knowledge propagation. A system called Object Category Specific Markov Random Field is proposed in [107]. The authors segment images by minimising the energy function of the model in an expectation-maximisation (EM) framework. The paper demonstrates robust segmentation of animals like cows despite occlusion and intra-class variance. In [205], holistic properties of the object shape are used for the detection and segmentation of objects. To that end, the authors introduce a figure/ground segmentation method which extracts regions that have a boundary structure simlar to that of the model and that are salient. The segmentation is solved as an integer quadratic programming problem. Further, [84] provides a framework for object detection, classification and segmentation, that uses regions and does not employ a sliding window. Thereby, an image is represented by a robust bag of overlaid regions and the regions themselves are represented by multiple visual cues. Then a Hough voting procedure is employed to generate hypotheses of the object location at different scales. Subsequently, a verifying classifier and a constrained segmentation are applied.

In addition, there is literature in a sub-field concerned with “pictures and words”. These approaches are for instance considered in the context of image retrieval or segmentation. One ultimate goal is to automatically provide accurate labels for all segments of a scene. Therefore, learning algorithms try to make use of available text, like tags, in an intelligent way. For a survey on related retrieval methods see [38]. One example where segmented images and associated text are learned jointly in a graphical model is [6]. The authors study different ways to extend existing statistical methods with multi-modal information and discuss the problem of measuring the performance. A more recent example is [123], where a hierarchical generative model is proposed that performs classification, annotation and segmentation in an automatic framework. Within this framework, images are explained through a visual model and a textual model. The model robustly learns from noisy Flickr tags and shows promising performances in all three tasks.

Detection and Context

If a classifier considers not only a local part of an image for detection, but the entire image, the classification performance can be improved. This is also quite true when the recognition is performed by humans [200], and as indicated by figure 2.9. As another example, in a street scene it is much more likely to perceive a car rather than a bed. Context does change the interpretation of an object and also defines what an unexpected event is [50]. The integration of contextual information is often realised with a probabilistic graphical model.

Putting objects in perspective is explored in [92]. Here, the authors see the detection of objects in relation to their 3D visual surroundings. In this case, a set of locations and scales can be truncated from the search space, which provides benefits for detection (e.g. more efficient detection and fewer false positives). Further, [179] proposes a probabilistic model to consider the scene for detecting objects, based on GIST [150] features. Given a large image set of scenes with many labeled objects, the authors map a novel input image to a scene in the database and then use specific priors learned for this scene cluster in order to transfer knowledge from old scenes to the detection process. In [88], context is automatically learned as clustered "stuff" and is shown to improve object detection performance. Here a probabilistic method is proposed that represents the contextual relationships between "things" and "stuff". Also [169] make use of the object context in a post-process of object recognition. Thereby a conditional random field is used to incorporate the contextual labels for maximising the label agreement. In another work, [194] models categories as a distribution over 3D location and appearance features. To that end, context is considered in a transformed Dirichlet process to effectively learn the 3D object structure of offices from 2D segmentations. The work in [41] provides an empirical study of context in object detection on the PASCAL VOC 2008 dataset. This allows the comparison of context-based methods with top-scoring context-free approaches. Also different ways of using context are compared. The authors conclude that the use of context reduces detection errors and that using multiple types of context makes for better performance.

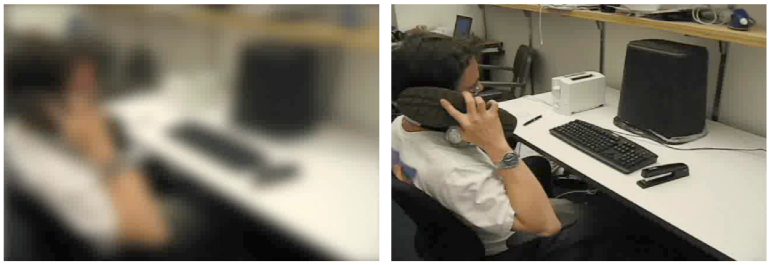


Figure 2.9: When looking at the left image, people recognise objects like a monitor, a mouse and a telephone. However, as the second picture reveals, people use context to make this inference, since the objects are to blurred too be recognised in isolation. These pictures are taken from [50].

Multi-view Detection

If a detection system should be able to recognise objects from different view-points, it becomes a basic the question how to realise this. In some of the approaches discussed so far, views are considered implicitly. The training set is supposed to be representative and is expected to contain different views of an object. Therefore after learning, the detector should be invariant to the viewpoint, to a certain extend. If the training set is labeled with viewpoints, different classifiers can be trained for different views. For instance, a face detector could be trained on frontal faces and also for left and right profiles of faces. Instead of heaving multi-view models, it is also possible to employ a full 3D model of an object for multi-view detection.

In [221], part-based detectors for human heads are trained for different viewing angles. The proposed system demonstrates an improved recognition rate by combining different orientation tuned models. Also the above-mentioned deformable part model [54] learns several components for realising a mixture model that is more invariant to the viewpoint of the object. In figure 2.10, two components of a mixture models for cars are shown. Further, in [199], the implicit shape model [115] is combined with a multi-view recognition system [65]. At first, a set of single view codebooks is trained which are connected by tracking regions across different images. Together, these codebooks can be used to vote for the location and scale of an object. The authors show that

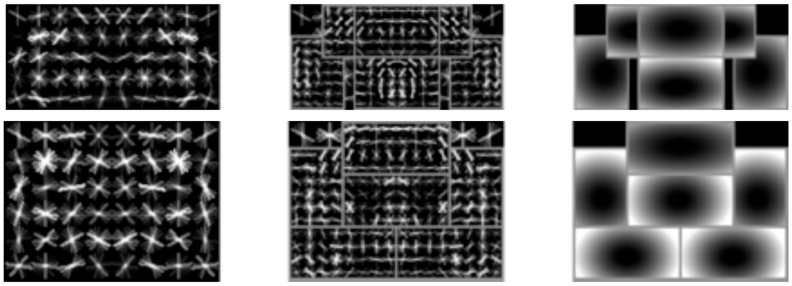


Figure 2.10: This figure by [54] shows a mixture model with two components for the category car (side view and frontal view).

their system outperforms single-view approaches. Also [184] extends the implicit shape model [115] to deal with multiple viewpoints. The approach requires a relatively small number of training samples and can detect pedestrians under different articulations and view-points. The different viewpoints or articulations are learned unsupervised by clustering silhouette images with the Chamfer distance. Besides, [93] exploits the availability of a rough 3D model of the object in an object detection and segmentation task. The paper describes a probabilistic framework to allow reasoning about occlusion, part consistency and pixel-level appearance. The method shows robust results for recognition of arbitrary views of cars in real-world scenes. In [125], mixture models are learned from synthetic 3D data in order to describe the geometry of a class. Then, the appearance and parts are learned from a set of 2D images and represented within a spatial pyramid. Both sets of information are linked and can also be used to estimate the 3D pose. The work in [95] presents a probabilistic model for mixing 3D and 2D primitives, and the learning of mixed templates to realise a viewpoint-invariant recognition. As primitives the authors propose robust stick-like elements in 2D and 3D, which are implemented by Garbor filters. The problem of object detection can also be extended by the task to simultaneously estimate the viewpoint. This problem is studied in [191], where a part-based probabilistic representation is employed to learn a 3D model from a dense multi-view representation of the viewing sphere. The method presented in the paper can also be used to generate synthetic views of an object.

The work in [85] also aims at simultaneous detection and 3D viewpoint estimation. The proposed model learns a set of holistic templates for this task and allows the use of different levels of supervision. Also, here the authors were able to demonstrate significant benefits from using their discriminative mixture-of-templates approach.

Efficient Detection

Some works in the field of object detection do not focus directly on visual features and their application in classifiers, rather they try to formulate general efficient principles for detection algorithms.

As stated before, many approaches make use of the sliding window principle in conjunction with an SVM-based classification. In an interesting work of [110] an efficient scheme for localising objects in images is proposed, called the efficient sub-window search (ESS). Thereby, a branch and bound scheme ignores large parts of the search space and converges to the optimal solution. The method can be used with a variety of kernel-based SVMs, e.g. the spatial pyramid kernel [113]. In [109], an efficient divide-and-conquer cascade for nonlinear object detection is proposed, called the efficient subwindow cascade (ESC). The mentioned ESS scheme can be directly integrated in this approach and allows very efficient detection with kernel machines. Another work of [4], also builds on the ESS scheme. The authors argue that ESS fairly often shows slow convergence, when no target object is in the image. Therefore, the paper proposes an algorithm with a better worst-case complexity, and also an approximation of that algorithm which is again much faster. In another work, [129] show an efficient way to compute the decision function for kernlised SVMs. For a set of kernels, e.g. the minimum intersection kernel, an approximate classifier with constant time and space complexity can be achieved, independent of the number of support vectors. This allows the authors to realise large improvements w.r.t. time and space complexity, while having the same classification performance. The authors of [212] generalise this idea of [129] and present fast approximate solutions for a family of additive kernels, that are often employed in computer vision. The results show significant speedups for train/test time, without altering the performance.

Going in another direction, the authors of [22] propose a system for faster object detection that is inspired by object search as performed by humans. Thereby, the authors implement a digital fovea and schedule eye fixations to maximise the information gain via stochastic optimal control. They demonstrate an increase in speed by a factor of two while reporting little loss in accuracy. Besides, [177] proposes an alternative to the sliding window approach by first segmenting an image and then selecting a small number of regions to which a classifier is applied. The segmentation is realised as an approximate solution of a directed Steiner tree optimisation problem. Other studies does not try to minimise the cost of computation, but the cost of annotation, because labeling training images is often expensive. The work of [118] proposes a way for semi-supervised boosting. The authors demonstrate an improved object detection of faces and vehicles when incorporating unlabeled samples. The efficiency of training and testing is also discussed for other specific techniques. For example, a method for fast deformable object detection was proposed very recently in [157] and realises a additional speed for cascaded DPMs [53]. The basic idea is to minimise the number of part-to-image comparisons by using multiple-resolution parts alongside a coarse-to-fine inference procedure.

Embodied Detection

There are also some approaches for embodied category detection, i.e. the category detection is performed by (mobile) robot. Category detection with mobile robots is actually a part of robot competitions today (cf. the Semantic Robot Vision Challenge (SRCV) [197] described in section 2.3). If fast enough, common recognition methods - as presented in this or the previous section - can be employed by the robots. However, due to the limited resources and computing power this is often not directly possible.

The 2007 version of Curious George [89], the winner of the SRVC challenge, uses SIFT features and a direct matching on ranked images similar to [127] to recognise objects. To focus the classification to interesting regions in the scene, a special attention system and saliency computation is employed. In a later version of the system [135], the attention system of the robot also employs structural information belonging to the 3D environment. At this juncture, the 3D information also allows the detection of pop-outs, like a bottle on a table. For learning category-level objects (e.g. frying pan), the deformable part models (DPM) [57] is employed. An integrated vision system has a closed loop to the world. This fact can explicitly be used in order to improve the quality of the detection process. For instance, the robot could move to another viewpoints and aggregate the information of multiple views, e.g. [240], [134] and [90]. In [134] and [90], category detection with a mobile robot is considered a sequential recognition problem. The authors propose a probabilistic approach to plan the next optimal viewpoints and were able to demonstrate benefits for the recognition process. In addition, new labeled data sets for evaluating multiple viewpoint recognition algorithms are provided. Other robotics scientists also explore the problem of multimodal object categorisation. For instance, [146] presents an unsupervised categorisation approach based on the probabilistic latent semantic analysis (pLSA), which employs audio-visual and haptic cues. The authors validate their approach in an experimental setup.

2.3 Datasets and Challenges

In this section, some common datasets and challenges for object recognition are presented. There has been a lot of progress w.r.t. datasets, especially in recent years, when scientists have employed Web images and had large collections of images annotated. In the process, reasonable data for benchmarking recognition systems have been created. The development of performance measures is ongoing work and incorporates critical remarks from previous databases, like [166] and [162].

- **Columbia Object Image Library** [147]. The COIL image database was established in 1996 and consists of 100 objects which are photographed in front of a black background. Each object is recorded in different poses by fully rotating a table in five-degree steps, resulting in 72 images per image.

- **ETH Database** [117]. The ETH database was created in 2003 and is composed of 8 categories. For each category there are 10 different objects. These objects are photographed in front of a homogeneous background from 41 different viewpoints, which are equally distributed over the viewing hemisphere. Also segmentation masks are available.

- **Caltech-101** [48]. The Caltech-101 database was introduced in 2003 and consists of 101 categories. For each category 40 to 800 images have been collected from the Web. The images have a resolution of approx. 300×200. Also the outlines of the objects have been annotated. The total number of images is $9,144$.

- **ESP** [217]. The ESP game was proposed in 2004 with the basic idea of getting images labeled, as part of a fun online game. It has been shown that people are willing to annotate images for free, because it is fun to play. The game results in a dataset of Web images labeled with words.

- **Amsterdam Library of Object Images** [77]. The ALOI database was contributed in 2005. It subsumes images of 1000 different objects. Each object has been placed in front of a black background, with changes made to the viewing angle (72 directions), illumination angle (24 configurations) and illumination color (12 configurations). This results in a collection of $110,250$ images. Also wide-baseline stereo images are available.

- **Caltech-256** [82]. Due to some issues with the 101 dataset (e.g. [166], [162]), the Caltech-256 dataset was provided in 2006. The dataset is composed of 256 object categories, where each category contains 80 to 827 objects. The images are much less restrictive and the objects appear in a much more variable pose and size. The overall number of images is $30,607$.

- **MIT Tiny Images** [203]. The MIT Tiny Image database was introduced in 2008 and consists of $80,000,000$ tiny (32×32) images that have been harvested from the Web by - loosely speaking - performing an image search for all words in a dictionary. The

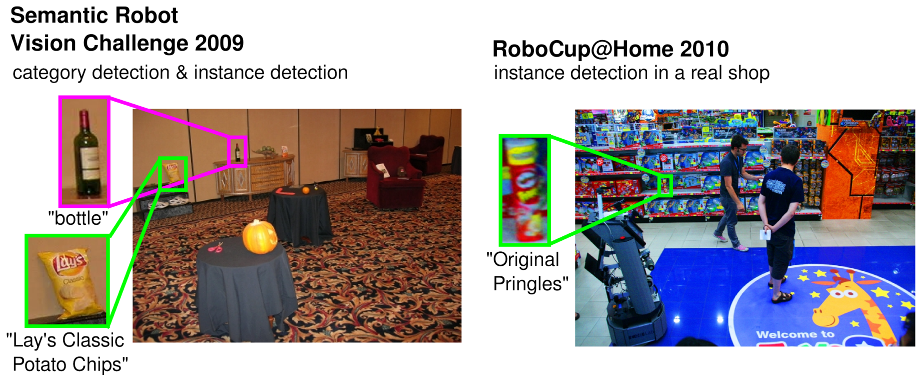


Figure 2.11: (*left*) Both categories (“bottle“) and concrete objects (“Lay’s Classic Potato Chips”) should be learned from the Web and detected in the arena of the Semantic Robot Vision Challenge 2009 [197] (image source [223]). (*right*) At the RoboCup@Home 2010 in Singapore [174], instance detection (“Original Pringles“) was also a sub-problem as part of the shopping mall task, which was carried out in a real Toys“R”us shop.

entire dataset also provides GIST features [150] for each image and has an overall size of approx. 400 GB.

- **LabelMe** [204]. The LabelMe framework and dataset was presented in 2008. The database contains more than 500 different categories with a total number of over 400, 000 closed polygon annotations. Usually large parts of the scenes have been labeled, which allows inter-object relationships to be studied.

- **ImageNet** [39] ImageNet was introduced in 2009 and is an ongoing effort to create a large visual dataset of Web images that is organised according to the WordNet [52] hierarchy. In doing so, the notion of synsets allows the mapping of multiple words or phrases to a semantic concept. Over 12, 000, 000 images have been already been collected for over 17, 000 synsets[1]. Also some bounding box annotations are available and are continuously added to the images.

---

[1] April 30, 2010: http://www.image-net.org/about-stats

- **PASCAL** [46]. Each year, the datasets of the PASCAL Challenge are released (2005–today) and provide training and testing data for category detection, category classification and segmentation. The PASCAL challenge serves as a benchmark for algorithms in the field. The 2010 version of the datasets for category detection consists of 20 classes, $10,103$ images and $23,374$ annotated bounding boxes.

Besides these major datasets for object recognition, there are also some more specialised datasets. A large database for indoor scenes was presented in 2009 by [168]. Further, there is also a rich fund of datasets for pedestrians (e.g. [155]), faces (e.g. [83]) and cars (e.g. [1]). The PASCAL challenge has already been mentioned as an important competition for category detection algorithms. However, category detection is also part of the Semantic Robot Vision Challenge [197]. Here the robots get a list of both specific items ("Gladiator DVD") and category level items ("bottle"), which they should detect in a semi-realistic environment after learning from the Web. Thus, additional challenges are e.g. the learning of object models from the Web and the navigation of the robot in order to find the objects in the environment. See figure 2.11 (left) for the setup. As already stated in the embodied detection section, Curious George [89], [135] is a leading system. It was also able to detect category-level objects in the environment. In contrast, other robotic competitions like the Robocup@Home have demonstrated that instance recognition can still be a challenging problem today, e.g. when detecting an object in a real shop under unknown lighting conditions. See figure 2.11 (right) for a Robocup setup. In the next chapter, the process of acquiring and annotating data for the domestic objects in this thesis is discussed.

2.4 Relation to this Thesis

As seen in this chapter, category recognition is a very large and highly dynamic field, where a vast variety of principles and techniques is studied. In the following, I want to provide a short overview, which basic concepts are employed and extended by the system developed in this thesis. The details are provided later on in the respective sections of the document. At the very basis, the method developed builds on the idea of Bag of Words – i.e. representing an image region by an unordered set of local features. BOW is chosen, because it has been shown to be a powerful representation with respect to its capacity and distinctiveness, e.g. when local features sets are mapped to a multi-resolution histogram. However, the loss of location information (since the feature bags are unordered) has been identified as a major drawback of the BOW approach. Static, object independent tessellations of the image plane have been proposed in the literature as one way to deal with this problem. Alongside, the usage of pyramid structures has been proposed in order to create an improved representation, that can be matched more efficiently. The work of this thesis uses these ideas as a basis and further generalises them to allow a better alignment to the underlying structure of the object. It has already been shown that aligned tessellations perform better than static ones, when employed in the domain of the feature space. The work presented here introduces a way to additionally align the tessellation in the image plane. Here, a tuning of the pyramid structure according to underlying topology of the object is proposed. Further, this tuned tessellation is realised hierarchically in order to implement a coarse-to-fine matching of local BOW-based representations. The resulting matching is called view-tuned kernel, and it matches parts of objects from coarse-to-fine by approximately matching local feature sets. To a certain extend, the proposed kernels can be seen as moving the BOW idea closer to the of idea part-based models, since the Voronoi cells are adapted to the structure of the objects, and the matching in the local Voronoi cells can be interpreted as a matching of the objects parts. This notion is then combined with the idea of decomposing complex training sets, in order to train different experts for different basic views of an object. In doing so, I propose a novel method for splitting the training set, which is unsupervised and which doesn't automatically produce mirrored model components like other approaches. A lot of the methodology that is developed subsequently in the thesis is dealing with the challenge to apply the proposed kernels efficiently in a SVM-based object detection framework. Note, that a lot of the BOW and spatial BOW related groundwork has “only“ been applied to category classification problems, and the additional challenge of object detection is often not discussed. As will be seen throughout the document, the application of the view-tuned category classifiers for object detection is not a plug-and-play process. Techniques like cascades, ensembling, or the improved computation of the decision function need to be specifically adapted to work with the proposed methods efficiently.

Chapter 3

Data Acquisition and Annotation

This chapter describes the data acquisition and annotation process. First, section 3.1 describes the set of domestic object categories that has been selected for investigation in this thesis. Then, section 3.2 presents an approach for retrieving images from the Web. Section 3.3 explains the procedure of data annotation, including the annotation format and the developed user interface. Finally, the results of the downloading and annotation step are presented and discussed in section 3.4.

3.1 Category Selection

In this section, the domestic object categories that are studied in this thesis are introduced. Overall, I decided to use 10 object classes that can be found in virtually every household today. The objects are from the following fields.

- Clothing: handbag, shoe
- Vessels: bottle, bowl, cup
- Electronic devices: laptop, light switch, toaster
- Biological items: apple, potted plant

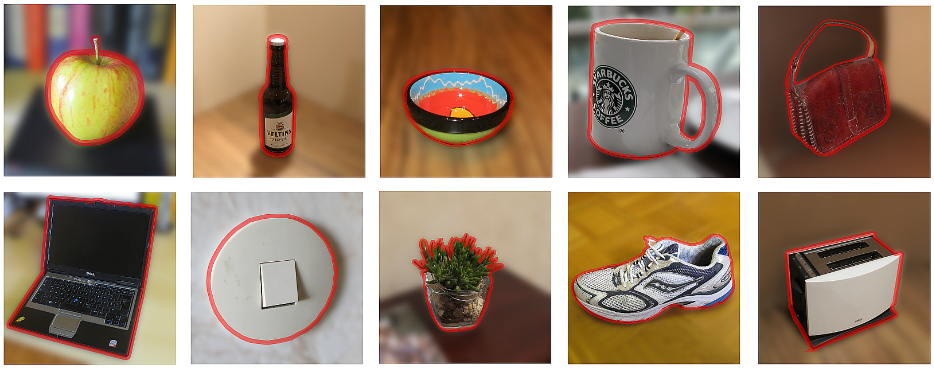


Figure 3.1: A prototypical instance of each selected domestic object category (first row: apple, bottle, bowl, cup, handbag; second row: laptop, light switch, potted plant, shoe, toaster).

Figure 3.1 shows prototypical examples for every selected category. The mentioned categories are selected for different reasons. First, the fact that humans use instances of these classes on a daily basis renders the categories interesting in the context of domestic service robots, as described in the introduction. In addition, these categories are chosen, because they are quite general and thus very challenging to recognise – for example, two different handbags could look very unsimilar. I think these general categories are much more challenging to recognise than e.g. faces. In addition, the visual appearance of the selected categories makes different demands on the abilities of the recognition system. For instance, a laptop keyboard is quite textured in comparison to an apple and the shape of a cup is quite well defined in comparison to a potted plant.

As seen in the related work section, two basic directions can be distinguished in the literature: One the one hand, there are methods trained and evaluated on datasets that incorporate many different classes, but have a relatively low number of images images per class. For example, methods using the Caltech-101 [48] or Caltech-256 database [82]. On the other hand, there are approaches that work with few categories, but with many images for every class (e.g. in the PASCAL VOC challenge [45]). In this thesis, I decided in favour of the latter approach, i.e.

using a small number of categories, while having large amount of available data. I did this, because I think the problem of learning a classifier for the selected categories is very complex and needs a large number of training examples. Beyond that, the generalisation abilities of the system can only be assessed using a reasonably large number of testing examples, as argued in the introduction. Also, the processing of a large quantity of data provides a more detailed insight into the problem and the existence of large datasets facilitates the benchmarking of different recognition engines for domestic objects.

Looking ahead, the dataset I use in this thesis should (a) contain approx. ten domestic object categories, (b) contain images from different providers (shops, image search and photo collections), (c) provide a large number of samples per class, (d) contain bounding box annotations for the objects and (e) provide an extra tag, if an object appears in a home environment (for creating realistic test sets). To the best of my knowledge, no database was able to satisfy all of these requirements when I started this thesis in 2008. Therefore, I decided to acquire and annotate a new dataset for training and testing of the detection system that I developed, and which is the subject of this thesis.

3.2 Web Image Retrieval

Today, the Web can be considered the most comprehensive collection of images from our visual world. For instance, a single website like Flickr purports to host more than $6,000,000,000$ photos.[1] And the number of Web images is continuously growing. e.g. Flickr states that the number of uploads increases yearly by 20%.[1] And as mentioned before, it has become the common paradigm in category learning to make use of these images in order to create suitable data for the training and evaluation of recognition methods. Within that context, this section explains the fundamental step of receiving Web images for the selected domestic categories. The results are discussed in section 3.4.

---

[1] April 8, 2011: http://blog.flickr.net/en/2011/08/04/6000000000/

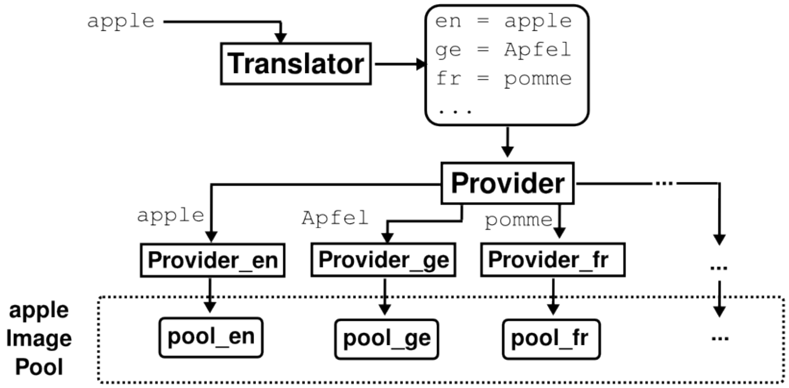


Figure 3.2: Sketch of the multi-language retrieval from a provider.

Approach

First of all, I want to distinguish three different kinds of Web image source: (1) image search engines, (2) photo portals and (3) online shops. This is done in order to differentiate the precision of the different sources in section 3.4. The images from the different sources are of a quite different nature and are merged at the end. For instance, images from photo portals are usually high resolution photographs of real world scenes, whereas images from online shops show isolated products in front of a homogeneous background. Besides, the image search engine output often contains images from photo portals and online shops, but also other images that are embedded in web pages. Thus a joined image pool will provide the most diverse training set, which usually facilitates the generalisation ability of a trained classifier.

For all kinds of stated source, text-based queries are used as an interface to retrieve images from a concrete provider. For example, the string “apple” is used as a search key to retrieve images for the category apple. In fact, all presented category labels directly serve as the query for an English language search.

For every kind of source (1), (2) and (3), a leading image provider was selected from the market, denoted as P_1, P_2 and P_3 respectively. The image search provider P_1 has the limitation of returning a maximum of $1,000$ results for a given query – as most image search providers do. Therefore, a *multi-language retrieval* is employed (e.g. [59], [31]). That means, the English search key is translated into different languages and the translated string is used for querying the image search engine. For instance, “apple” is passed to the English provider $P_1^{\mathtt{en}}$ and the translated French word “pomme“ is passed to the French provider $P_1^{\mathtt{fr}}$. The multi-language retrieval is sketched in figure 3.2. Because the providers internally use different indexes for different languages, the number of found images can be greatly extended without the risk of producing duplicated results. Translated queries are also only employed if the translated string is different from all other languages. For the provider P_1, I use a multi-language retrieval to gather a total number of $10,000$ images per category. Although, technically a maximum of $1,000$ images can be returned for a single query by provider P_1 in reality the number of gathered images is usually between 700 and 900. This is because either some of the images are not available anymore or because they are filtered (see below). The photo portal provider P_2 restricts the number of returned images to a maximum of $2,000$ images. Also here, I use a multi-language retrieval to gather a total of $8,000$ images. For the online shop provider P_3 a single language retrieval is employed, because usually multi-language translations just refer to the same underlying product and images. The number of returned images for P_3 includes everything that is available, but at most $5,000$ images. At the end, the joined image pools consist of $18,000$ to $23,000$ images per category.

When an image is retrieved by the system, a set of meta-data is also stored. The meta-data includes the following information: the associated category, the English search key, the translated search key, the language of the translated key, the image provider, the provided image URL, the rank of the result image, the retrieval date, the image filename, the size of the image, as well as the image type. The meta-data is stored in the same XML scheme that is used for annotation (described in section 3.3). For an example of a meta-data block see listing B.1 in the appendix.

After downloading, all images are converted to RGB 8-bit JPG images. If images exceed $307,200$ pixels, they are scaled to that number of pixels. In addition, images with a size smaller than 100×100 are filtered out. Images are also filtered if they are non-color images or

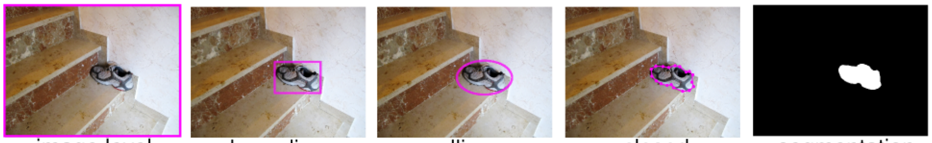


image level annotation
bounding box
ellipse
closed polygon
segmentation mask

Figure 3.3: Sketch of some prominent types of object annotation in images.

have the GIF format. For the technical realisation of the process an extendet version of the retrieval framework of [126] is used.

3.3 Web Image Annotation

In this section, the annotation of Web images is described. First, the selection of the annotation type and format is presented. Then the user interface for bounding box annotation is explained. The results of the annotation process, including the statistics on the annotation time, are presented and discussed in section 3.4.

Annotation Type and Format

Depending on the problem at hand, objects in images can be annotated with different types of annotation (e.g. cf. [195]). Some prominent examples are described as follows and are also sketched in figure 3.3.

- **Image Level Annotation.** An image level annotation states whether an instance of a category is present in the image or not. It does not include information about the location or size of the objects. This kind of annotation is usually employed in the context of classification.
- **Bounding Box.** A bounding box annotation is an axis-aligned box that tightly encloses the object of interest. The bounding box can be defined by (x, y, w, h), where (x, y) is the position of the upper left corner, w is the bounding box width and h the height of the box. Bounding boxes are often used in the context of detection.

- **Ellipse.** Similar to the bounding box, an axis-aligned ellipse can be used to enclose the object of interest.
- **Closed Polygon.** A closed polygon is a flexible way to describe the boundary shape of an arbitrary image region. It is defined by a closed chain of connected points.
- **Segmentation Mask.** A segmentation mask stores the exact pixel coordinates of foreground pixels of an object. For example, it can be represented by a binary image, where 1 denotes the foreground (white) and 0 the background (black).

By specifying a rotation angle, bounding boxes and ellipses can also be rotated in order to allow non axis aligned bounding shapes. Within this work, I decided to use axis aligned-bounding boxes for the annotation of class instances in images. This is done for the following reasons. The first and most important reason is that bounding box annotations can be produced quite quickly. In comparison, the annotation of large datasets with complex polygons, like in the LabelMe Database (e.g. [204]), would either need a huge amount of time or a lot of volunteers. A second aspect is that the learning from bounding boxes and the detection of bounding boxes is a common procedure in detection challenges, like the PASCAL VOC challenge (e.g. [45]). Another aspect is that bounding boxes are suitable for common feature extraction methods that work on rectangular image regions and allow application in a sliding window detection scheme [110].

Of course, bounding boxes can be represented in different formats, e.g. plain text, XML or binary. In this thesis, XML is used to represent the annotations of an image and the corresponding meta-data. The XML is created on the basis of a validateable XML Schema Definition (XSD). For an example of an XML annotation file confer listing B.1 in the appendix. In doing so, the XML is structurally compatible with the widely used PASCAL (e.g. [45]) annotation format, i.e. the important fields are present. XML is used because of its extensibility and because it can be parsed and converted into any other format effortlessly.

While annotating the images, i.e. labeling objects of a category with a bounding box, several aspects which are considered. Most important

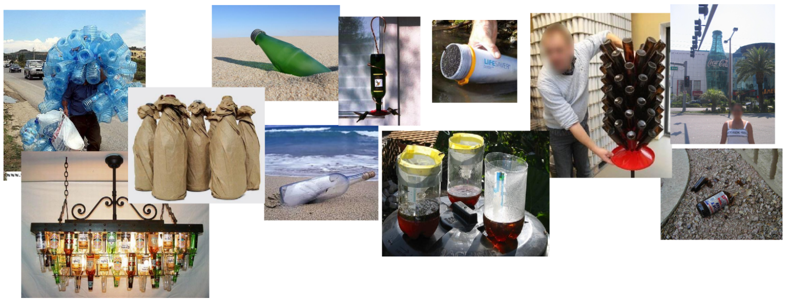


Figure 3.4: All of these images actually contain a bottle, but the annotater does not label them, because the bottles are in a very untypical configuration w.r.t. a service robot.

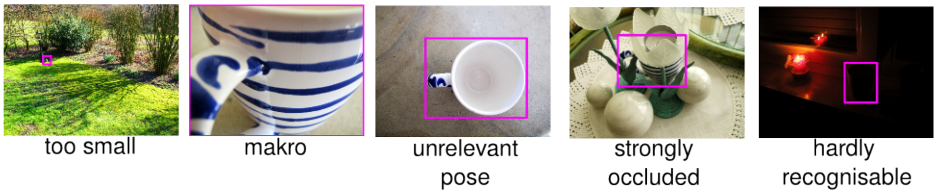


Figure 3.5: Samples of class instances that are ignored during annotation.

is the fact that the annotation is restricted to samples that appear in a typical configuration w.r.t. a service robot like BIRON (e.g. [86], [218]). To make this clear, figure 3.4 shows some images with untypical configurations of the category bottle. In the context of service robots, I focus on upright standing objects and thus can ignore very different examples. To a certain extent, the knowledge of the application domain makes life easier by posing a more specific objective than in the arbitrary and ill-posed "detect any bottle" scenario. Further, the annotaters "ignore" some samples entirely, by not labeling them. For instance, getting a makro perspective or a top-down view of a cup is quite unlikely, while 90-degree side views are very likely to be perceived by the robot. In other situations samples are ignored as well: if a sample is very small, if major parts are occluded or if the object is very hard to recognise (e.g. in case of low contrast or bad lighting). Visual examples for the situations mentioned are provided in figure 3.5.

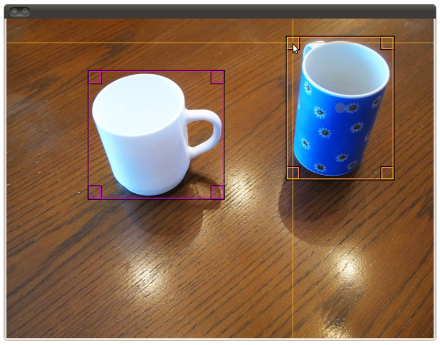


Figure 3.6: User Interface for bounding box annotation.

Beyond that, images that show samples in a regular home environment are labeled with an additional `@HOME` tag. Only images tagged with `@HOME` are used later on for the evaluation of the detection system. This is important, since it allows the composition of a suitable test set with samples from the domain of application, i.e. with samples from real environments.

User Interface

This chapter describes the user interface that has been developed for the purpose of bounding box annotation in this thesis. A main goal within this context is to facilitate the fast creation of accurate bounding boxes. Later on, in section 3.4, both the results of the annotation process as well as the time required will be presented and discussed.

The user interface employed for bounding box annotation is shown in figure 3.6. As is clear, it does not contain any buttons. It is designed for use with one hand on the keyboard and one hand on the mouse. The mouse is used to create add, delete, resize and move bounding boxes. The keyboard is used for going through the images sequentially, either with or without automatically saving the XML file. Also ignoring an image is done via keyboard, e.g. when pornographic or offensive content is displayed. Ignored images are excluded from the statistics and from further processing. When switching through the images sequentially with the keyboard, the corresponding bounding boxes are displayed as defined in the automatically loaded XML file. Thus, when no category instance is present in the image, it can be annotated very quickly by simply switching to the next image. One of the most important features of the interface is that, when configuring the program at the beginning of an annotation session, a bounding box pool and the associated category must be specified. This category label is then used during the process of annotation as a label for all bounding boxes that are created. This means that, the error-prone and time-consuming step of typing in labels is completely avoided. In addition, since only one category is annotated during a single session, the user can focus solely on this task and doesn't need to switch between categories mentally.

3.4 Results and Discussion

The multi-language retrieval and the annotation were performed as stated in the last sections. The results from these steps are shown in figure 3.7 for each of the domestic object categories and are discussed in the following. The chart (a) shows the number of retrieved images. For the providers P_1 and P_2, the number of retrieved images was set to $10,000$ and $8,000$ respectively, to allow a fair comparison of their precision. For the provider P_3, a single language retrieval was used, to avoid retrieving duplicated product images. This allowed as many images as possible to be acquired for the provider P_3, with the maximum set at $5,000$ images. The results show that in an online shopping portal with a very broad range of products, the number of retrievable images can vary quite a lot. For instance, there are many more images returned for the keywords shoe, cup and handbag than there are for light switch, apple and potted plant. This is not surprising, since shops with a broad range of products are not specialised for items like light switches that sell less often. This result means that if many product images were needed for a large range of domestic object categories, it would be necessary to implement additional interfaces to other specialised online shops (e.g. a special light switch store), as well. On average approx. 2,500 images were retrieved from the provider P_3 for the categories specified. Further, the plot (b) shows the annotation time in minutes for each provider and category. For the annotation, the previously presented user interface (UI) and procedure has been employed. Labeling

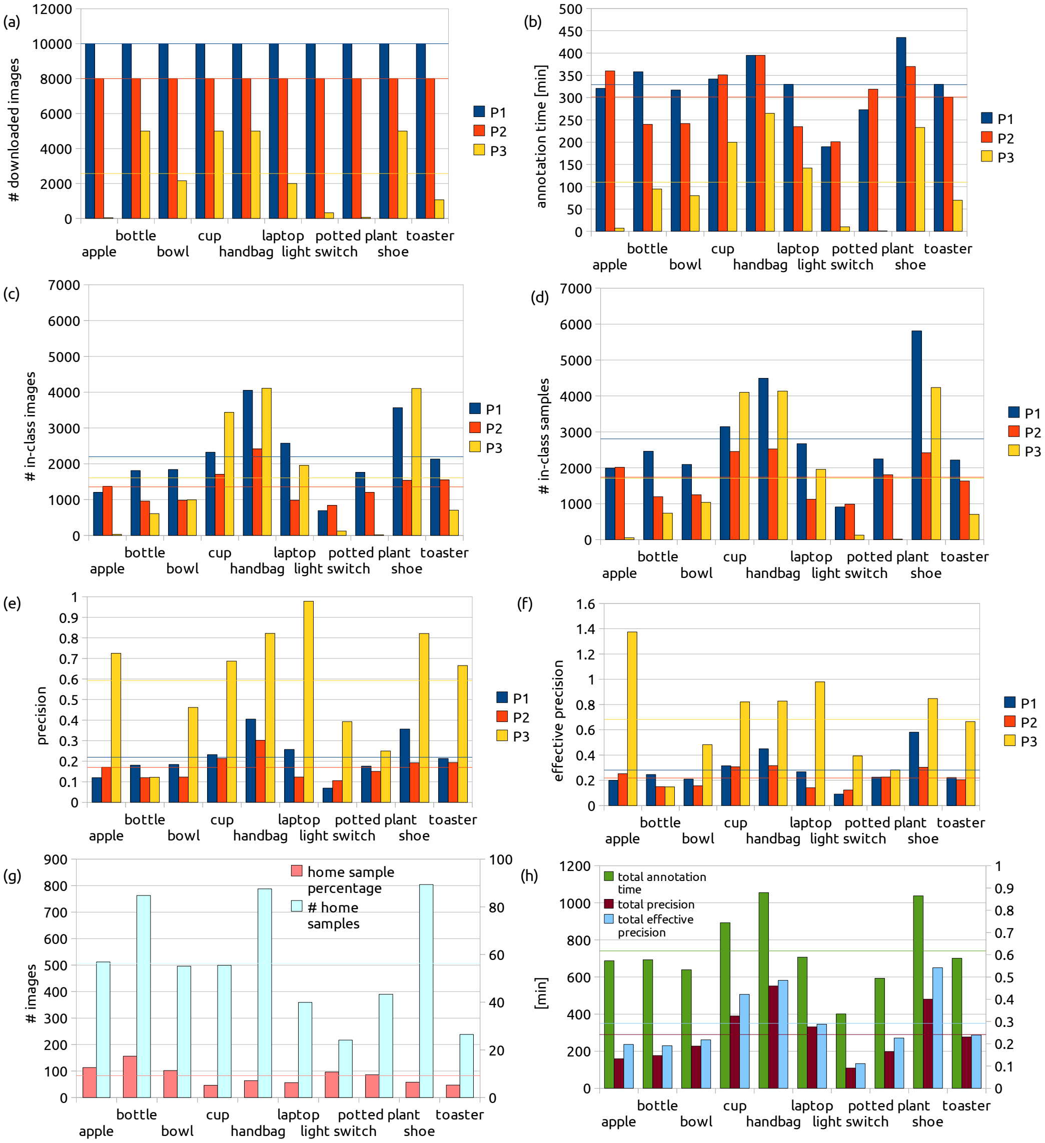


Figure 3.7: Results from the downloading and annotation step for each category. The colored lines depict the mean values.

overall	
# downloaded images	205644
# in-class images	51621
# in-class samples	62593
precision	.251
effective precision	.304
annotation time	7408 [min]

Table 3.1: The overall results from the downloading and annotation process.

10,000 images for provider P_1 took approx. 330 minutes on average and annotating 8,000 images for provider P_2 took approx. 300 minutes on average. The labeling for P_3 took approx. 110 minutes on average. Of course, the annotation time is directly connected to the number of processed images and to the number contained in-class samples, as presented next. In plot (c), the number of in-class images is displayed. As is obvious, for some categories it is much easier to retrieve in-class image than for others. Again, this is connected to the "popularity" of the items. On average 2,200 in-class images were found for provider P_1. For provider P_2 and P_3, approx. 1,400 and 1,600 in-class images were found on average. Figure (d) shows the number of in-class samples that were annotated for each category and provider. The number of in-class samples is always equal to or higher than the number of in-class images. At least one sample is found in an in-class image, but some images also contain more than one category instance. Note that there are also some images that show in principle a massive number of in-class samples in one image. For instance, images showing a large table full of apples or shoes. However, for those images only a few good samples were usually annotated, because these scenes are often very cluttered and the objects were quite occluded. Also, if an image contains 100 examples of the same cup, only a few instances at different poses are annotated, because this specific instance should not acquire excessive influence in the learning process. Comparing the averages of the number of in-class images and in-class samples, P_1 contains approx. 1.3 samples per in-class images, P_2 contains approx. 1.2 and P_3 has approx. 1.13 samples per image. In chart (e), the precision is given for each provider and category. The precision is computed as follows

$$precision = \frac{\#\, number\ of\ in-class\, images}{\#\, downloaded\, images}$$

and describes how many retrieved images actually contained (at least one) object instance. As can be seen, the direct downloading from the shopping Website provider P_3 produces many fewer false positives than the other providers P_1 and P_2. Further, the observation from before, that images for some categories can be retrieved more easily, is also directly reflected in the precision scores (compare handbag vs. light switch). On average, the provider P_1 has a precision of approx. .21, P_2 and P_3 have a precision of approx. .17 and .6. The figure (f) presents the effective precision for the categories and providers. The effective precision is a measure for a category that relates the number of in-class samples to the number of downloaded images and is computed as follows:

$$effective\ \ precision = \frac{\#\, number\ of\ in-class\, samples}{\#\, downloaded\, images}.$$

Thus, the effective precision is higher than the precision when multiple object samples are found in the download images. The average effective precision for the providers P_1, P_2 and P_3 is approx. .29, .21 and .7 respectively. The plot (g) shows the number of home samples for each category, i.e. how many images have been tagged with the `@HOME` tag after merging the images from all providers. In addition, the "home sample percentage“ (hsp) is plotted, which is computed also on the merged image sets as follows:

$$hsp = \left[\frac{\#\, number\ of\ home\, samples}{\#\, number\ of\ in-class\, samples}\right] \cdot 100.$$

The hsp relates the number of home samples to the total number of samples for a category. Remember, the `@HOME` tag was introduced to be able to create test sets with images from the domain of application. However, as visible here, both the number of home samples (minimum=217, maximum=804) and the hsps (minimum=5.1, maximum=12.5) are surprisingly low. That means that the vast majority of samples found in the images does not consist of photographs of the domestic objects taken at 1-3 meters' distance in real home environments. In other words, it can be said that getting test data from the Web for the purpose of testing detectors that should work in regular home environments is quite hard. For instance, you can get many images of laptops in front of a homogeneous background from (shopping) Websites, but the number of people who upload a usable picture of a laptop on their desk is quite smal. In fact, since the number of test samples is so small, I decided to use all `@HOME` images solely for the purpose of testing. If more home images had been available I would have divided the sets for for training and testing. So, the challenge and question at this point is, how good detection systems can get, when training on samples that do not come directly from the domain of application. As I will later show, it is possible to come quite far with this kind of training data. Finally, the chart (g) merges all providers and shows the total annotation time, the total precision and the total effective precision for each category. As is clear, the average total effective precision and the average total precision are quite similar, what means that in the overall image pool the majority of images contain only one category instance. The average annotation time for a class-associated image pool is approx. 750 minutes. An additional summary of the results can be found in table 3.1, where the numbers are merged again globally over the categories. The total number of downloaded images is 205644 and where 62593 in-class samples were found and annotated in 51621 in-class images. The overall precision is .251 and the overall effective precision is .304. The total annotation time is 7408 minutes, i.e. approx. 123 hours of work.

Overall, using the proposed procedure for acquisition and annotation of Web images from the three different sources, it is possible to create large and diverse sets of bounding box-annotated samples at a relatively low cost. The next chapters explain how the resulting sets of samples can be used to train a system that is able to detect novel object instances in unknown domestic environments.

Chapter 4

Principal View Extraction

This chapter describes a method for the extraction of *principal views* from a set of in-class images and for decomposing a training set according to those principal views. Figure 4.1 sketches the overall setup. Section 4.1 introduces the problem and goals, and section 4.2 presents the developed approach for view extraction and training set decomposition. Finally, section 4.3 states and discusses the results for the 10 domestic object categories of this thesis.

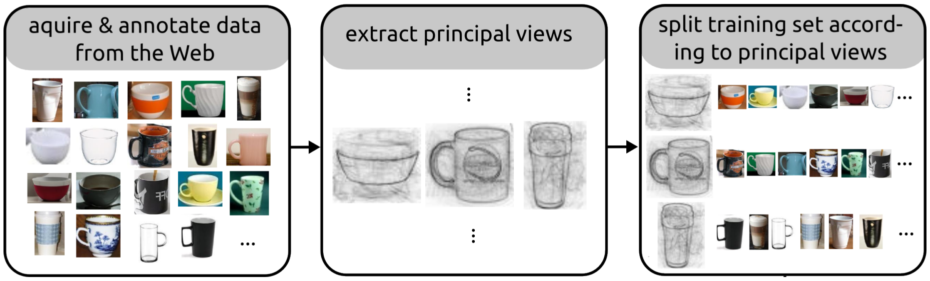

aquire & annotate data from the Web
extract principal views
split training set according to principal views

Figure 4.1: Sketch of the principal view extraction and training-set splitting.

4.1 Introduction

In the previous step, the images were downloaded from the Web and class instances were labeled with bounding boxes. Cutting out the annotated image regions results in a set of images for each category. This means that the resulting sample images usually do not have the same size or width-height-ratio. Taking this set of arbitrary-sized images, the goals of the view extraction are as follows. A number of principal views should be extracted unsupervised (since the views are not labeled) from the training set, and the training set should be decomposed into several sets. Further, the principal views should be visualizable (e.g. as an image) to allow interpretation by humans. In addition, the principal views should statistically capture the appearance of common shape and texture parts, in order to allow a training of the view-tuned kernels [105] in the subsequent processing step. After that, a view-tuned SVM is trained using the learned kernel for each of the extracted principal views. Finally, all view-tuned classification experts are fused into a robust detector for a category. In the process, the unsupervised view-extraction aims at breaking down very complex learning problems by splitting the training sets into several easier subsets and by training an expert for each set. This "divide-and-conquer" practice involving local experts has been employed and studied by various authors in the Machine Learning community (cf. [108] for a survey). For instance, [24] uses a self-organising feature map (SOM) to first partition the input space, and then train and optimise local SVM experts for each partition. The authors demonstrate a significant improvement in the generalisation performance, when using ensembled SVMs for time-series forecasting. Further, this basic idea of multiple experts has also been adopted by other authors for multi-view category detection, as presented in chapter 2. It has been shown that considering different view models does provide better performances for category detection, e.g. [221], [54], [199] and [184]. Since no 3D information or labeled views are given in the framework of this thesis, the view extraction method here must work solely with the 2D image samples.

4.2 Approach

This section explains the details of the approach used in the principal view extraction and the training-set decomposition. A visual survey of the method can be found in figure 4.2.

1. **Mirroring.** As a first step, all images in the dataset are mirrored

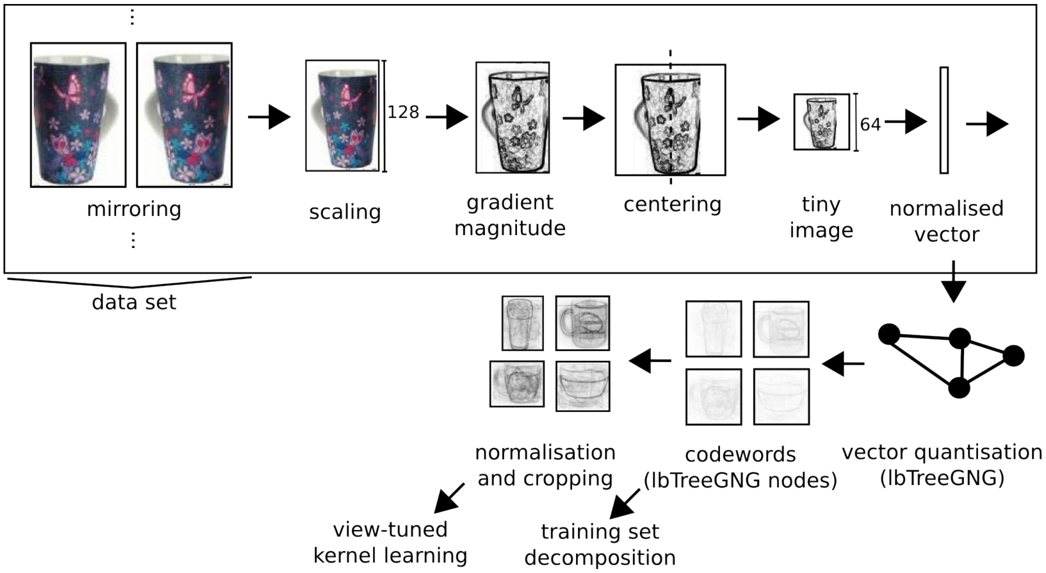

mirroring
scaling
128
gradient magnitude
centering
tiny image
64
normalised vector
data set
vector quantisation (lbTreeGNG)
codewords (lbTreeGNG nodes)
normalisation and cropping
view-tuned kernel learning
training set decomposition

Figure 4.2: This figure shows the basic steps of the principal view learning pipeline.

along the x-direction. This is very useful for enlarging the training set with real-world samples from different poses and doubles the number of training samples. A similar step is also performed in [54]. Note that, although only the mirroring is applied, it would also be possible to generate more training data with other transformations like small rotations, as e.g. proposed in [176].

2. **Scaling.** Because the target resolution of the detection algorithm is 640 × 480, and the objects of interest are quite small and expected to be distant from the camera, I choose to use a sliding window of width w and height h, such that $max(w, h) = 128$. Accordingly, the sample images are scaled proportionally such that the longest side is equal to 128 pixels.

3. **Gradient Magnitude.** In this step, the gradient magnitude image is used as a feature to capture the shape and texture information of the objects. The gradient magnitude is chosen as a feature, because in some publications averaged gradient magnitude images of in-class images reasonably capture the average shape and tex-

ture of similar objects (e.g. in [37], [14] and [105]). The gradient magnitude image is computed with the Sobel operator [97].

4. **Centering.** The gradient magnitude image is now centered on a 128×128 image. This is done in order to be able to get mathematical vectors of the same size for images of different sizes.

5. **Tiny Image.** In this step, the gradient magnitude images are tinyfied, i.e. scaled to 64×64 pixels. This is done in order to pool the gradient magnitude image information and to make it more invariant to small shifts. Beyond that, it has been shown that tiny images of objects can serve as holistic features and capture quite a large amount of the visual information (e.g. [203], [202]).

6. **Normalised Vectors.** The tinyfied gradient magnitude images are converted to a column-vector of the dimensions $64 \times 64 = 4096$. By normalising the magnitudes to the range $[0, 1]$, the final vectors are in $[0, 1]^{4096}$.

7. **Vector Quantisation.** Here, a method for vector quantissation is employed to create a small set of principal views. In doing so, the limited branching tree Growing Neural Gas (lbTreeGNG) [104] is employed. The lbTreeGNG is an extension of the Growing Neural Gas [73] and it is used because it allows efficient one-shot online quantisation of vectors sampled from an unknown probability distribution. Moreover, the lbTreeGNG is used because the learned codewords can be interpreted as supporting points of the learned principal manifolds. For that reason, I refer to the codewords as "principal views". The lbTreeGNG captures the topological structure of the input space by using a Hebbian learning scheme. Strictly speaking, it is not needed to pre-define the number of codewords for the lbTreeGNG. Given an error threshold, the network stops learning at a certain point and thus avoids overfitting. The details of the lbTreeGNG method are presented in the appendix A. The lbTreeGNG is used with the default parameters and $b = INF$.

8. **Decompose Training Set.** Given a trained codebook $\{c_0, \dots, c_{K-1}\}$

from the last step, the training set can now be decomposed as follows. For each image of the training set, a normalised vector ξ is computed as mentioned before. The vector is assigned to its nearest neighbor s, i.e.

$$s = \underset{i=0,\ldots,K-1}{\operatorname{argmin}} \parallel \xi - c_i \parallel_2$$

and the image sample is added to the set of the principal view s. With a probability of $p = .2$, the vector ξ is also added to the set of the second nearest codeword t, i.e.

$$t = \underset{i=0,\ldots,K-1,i\neq s}{\operatorname{argmin}} \parallel \xi - c_i \parallel_2 .$$

The second assignment is useful for softening the partitioning of the feature space and for allowing some degree of sample overlap between the experts. The decomposition of the feature space is not expected to be perfect on a semantical level anyway. A right-handed cup may be assigned to a codeword that represents mostly left handed cups. However, this is not a problem at all, because the learned matching kernel in the next section only realises a view-tuning. That means samples from the same view can be matched very well, but also samples from different views can be matched reasonably well.

9. **Normalisation & Cropping.** In order to provide the input for the next processing step, i.e. the learning of view-tuned kernels, a normalisation and cropping operation is performed. The learned codewords are re-shaped to 2D images and normalised, such that the minimum and maximum values are mapped to integers in the range $[0, 255]$. The cropping operation considers the statistics of columns and rows. Starting in column 0 and column 128, the number of foreground pixels is counted in each column. A pixel is considered as foreground if the its value is $< \theta_1$. If the number of foreground pixels is not big enough $> \theta_2$, then the columns are removed and the new columns are 1 and 127. The same procedure is applied in the y-direction by considering rows. As a result, white and very light gray regions around the centered

average object gets removed. Empirically, the parameters have been set to the following values: $\theta_1 = 205$ and $\theta_2 = 15$. Note, that the gradient magnitude image is stored inverted here.

4.3 Results and Discussion

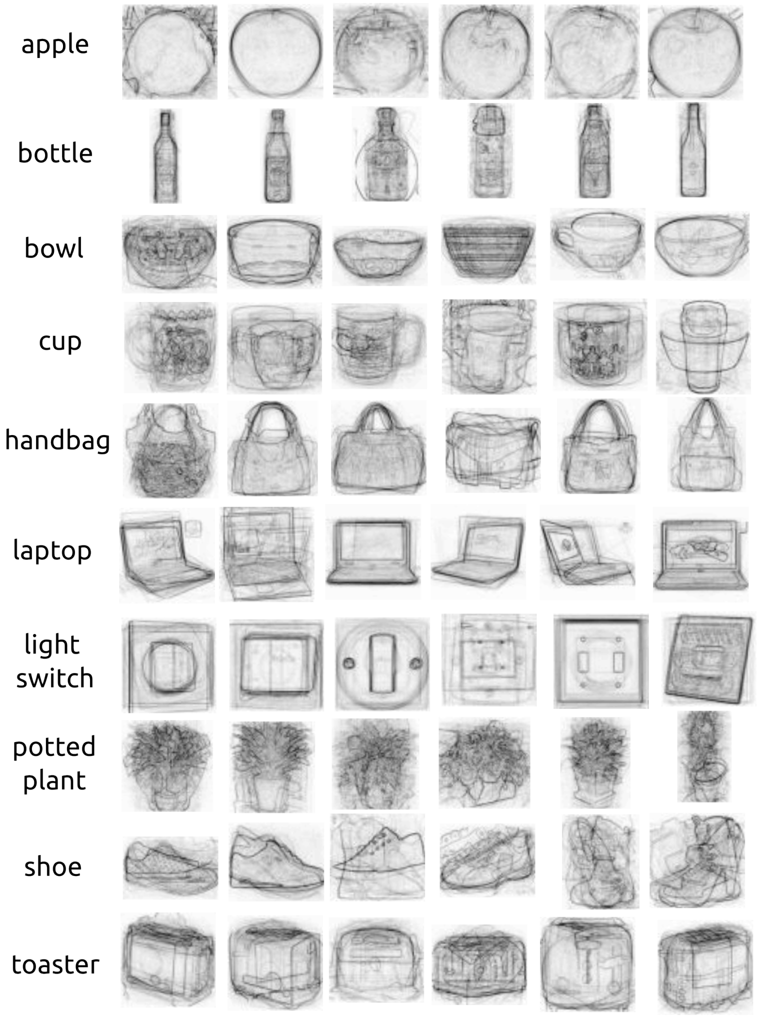


Figure 4.3: Learning six principal views for each category.

In this section the results from the principal view extraction and the

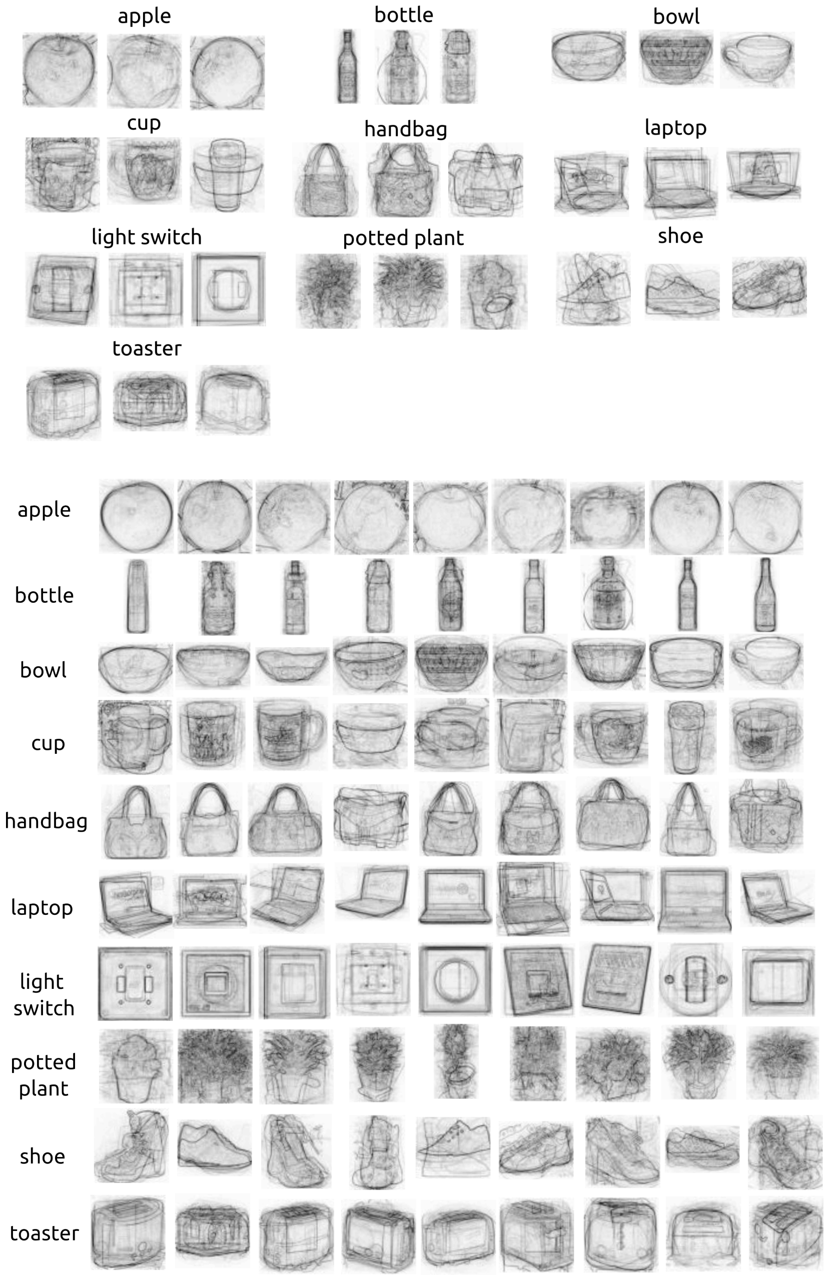


Figure 4.4: Learning three and nine principal views for each category.

training-set splitting are presented and discussed. The most critical parameter at this step in the pipeline is the parameter *m* of the lbTreeGNG, which directly affects the number of learned codewords or views. To allow a comparison between the models of the different categories and to lay common ground for the comparison with related work, *m* is chosen such that the number of codewords becomes equal to three, six and nine. The results for the extraction of six views is shown in figure 4.3, and the results for three and nine extracted views can be seen in figure 4.4. Note that the effect of the number of principal views on the overall system performance will be discussed in the evaluation in section 7 of the overall detection system.

As expected, all codewords of a category show a locally averaged instance of that category. For some categories, using three views produces quite clear principal views, e.g. for apple and bottle. This indicates that the elements of these training sets have a quite similar appearance and shape. However, for other classes the three extracted principal views are quite cluttered, e.g. for potted plant or cup. This indicates a more variable appearance and shape in the training samples. Going from three extracted views to six views, it can be observed that the clutter becomes effaced in some classes and the underlying views are revealed as the mixed clusters are decomposed (e.g. see laptop or light switch). For classes like apple, the codewords do get clearer and look reasonable, but no really new view is revealed compared with the previous views. Thus, using many views may not always be necessary when the samples in the training set have a similar appearance and shape. Because the method proposed for view extraction in this thesis allows the learned codewords to be rendered from the feature space as images, it is possible to gain an impression of the quality and reasonableness of the learned codebook. But not only principal views are revealed by the vector quantisation approach. It also seems to be possible to distinguish basic latent types of object, i.e. different types of handbag, bowl or show. This can be beneficial, since it also allows a tuning towards different basic object types. In general, when going from six to nine views, it can be seen that these tuning towards different types of object does occur more frequently. Although not shown here by example, this trend continues when using even more codewords. In this thesis, the codewords are thought of as representing the different views and the view-tuned SVM models are thought of as representing the different types. Therefore no larger codebooks are considered here, although the lbTreeGNG seems interesting for discovering latent types or sub-classes in a training set.

The decomposition of the complex training sets according to the extracted principal views is also an interesting thing to look at. Here, two major questions are addressed: First, what does a concrete example of a decomposition look like? Second, how are the training sets distributed over the different views? Figure 4.5 answers the first question, by showing six extracted principal views for the category cup alongside with 60 assigned training samples. The samples often match the viewing direction of the principal views, but samples from other viewing directions are sometimes assigned to the principal views. However, as stated before, this is not a problem, because the kernel that is introduced in the next chapter is able to match a left-handed cup with a right-handed model. In fact, the proposed method tries to mix up the models to a certain extent by assigning a sample with the probability .2 to the second winning principal view. The basic notion here is that the optimisation of the SVM will figure out what a good positive and negative sample is (given the kernel), if the training set is not mixed up too much. It can also be seen that a single viewing direction can be split (compare the 3rd and the 5th view in figure 4.5), while a mixed codeword remains undecomposed (compare 6th view). This is obviously a side effect from using an unsupervised method, as no high level semantic knowledge is available to support the decomposition. Although it has not been explored in this thesis, it may be interesting to think about some amount of user interaction here, since the extracted codewords are designed to be interpretable by humans. For instance, one could think about letting a human select semantically reasonable principal views. The second question that has been mentioned is, how are the training samples distributed over the principal views quantitatively. To allow a comparison of sets of different sizes, the relative ratio of the assigned samples is stored for each principal view. For example, if there are 1000 samples in total and a view gets 100 samples assigned, the relative ratio of the assigned samples is $\frac{100}{1000} = 0.1$. Since the views do

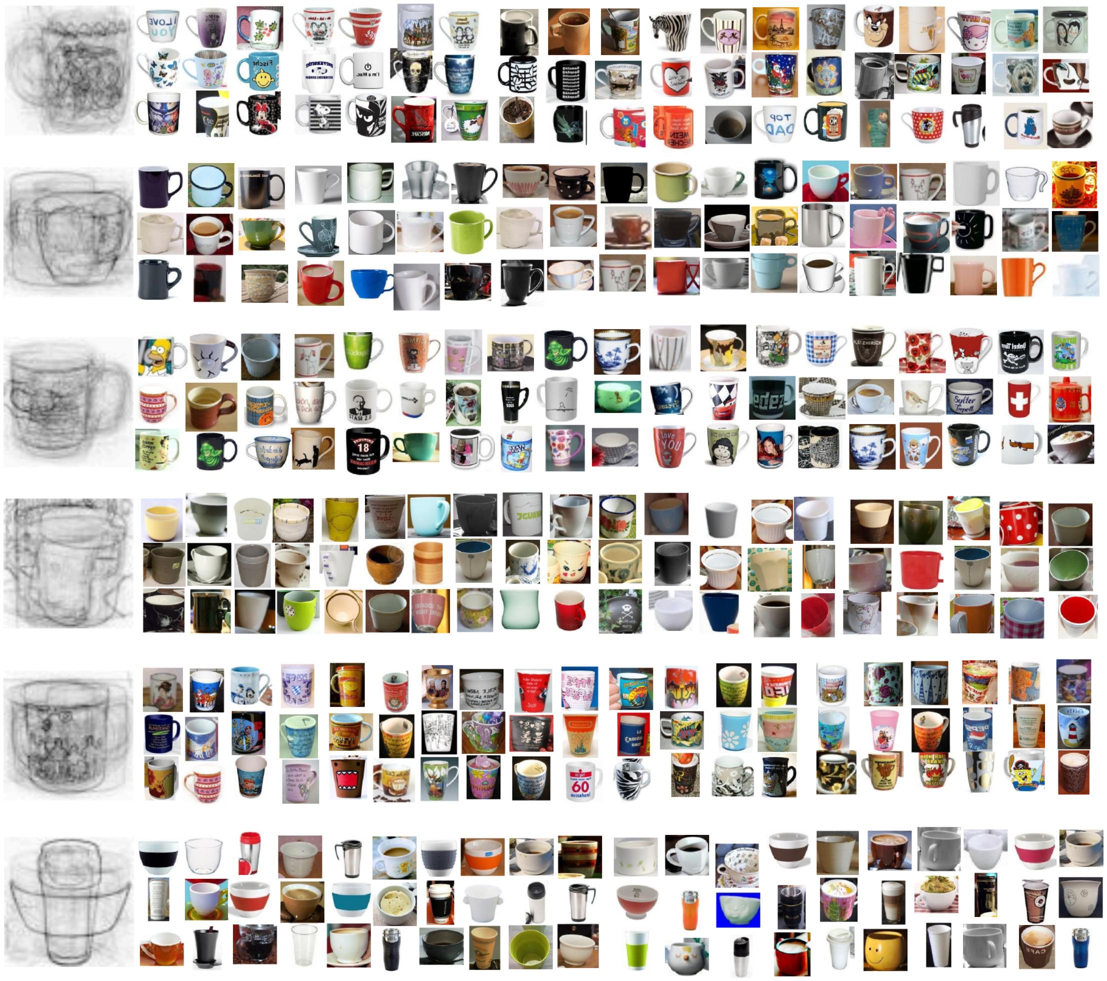


Figure 4.5: This figure shows the six extracted principal views for the category cup alongside with 60 assined training samples (randomly picked from all assigned samples).

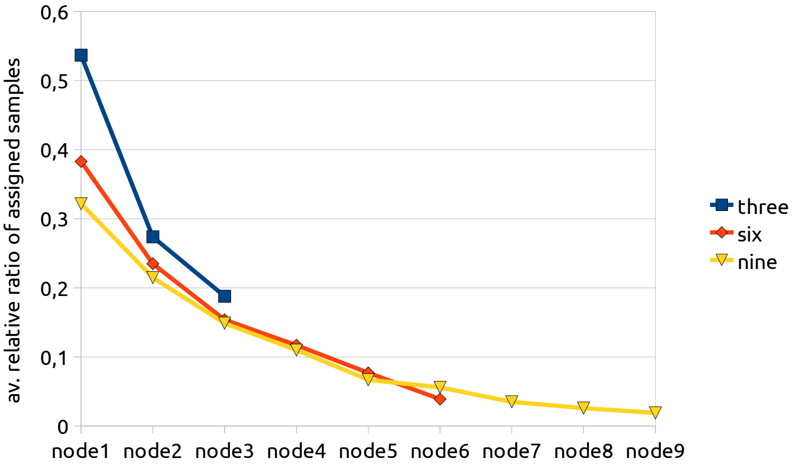


Figure 4.6: This figure shows the relative ratio of the assigned samples for six extracted principal views (explanation in the text).

not have a particular order, they need to be ordered to be comparable among different classes. This is done by sorting the views according to the relative ratio of the assigned samples in decreasing order. For a model with six principal views, node1 is used to refer to the view with the highest relative count and node6 refers to the view with the lowest relative count. With the bins in a particular order, the relative ratios of node1, node2, etc. can be compared among the different classes. For a model with three, six and nine views, the average ralative ratio of assigned samples is plotted in figure 4.6. Interestingly, on average the relative ratios seems to be similar to an exponential decay function. That means means that a fraction of the views embraces a majority of the training samples. I think this distribution is an effect of using photographs from the Web, because it is known that people prefer to take photos of objects from certain canonical perspectives (cf. [156]). Further, the chosen approach of ignoring irrelevant perspectives, like macros, may increase this effect. Although in this work full ensembles are considered, this observation may be useful for the selection and truncation of ensembles (similar to choosing the first principal components in principal-component analysis). In addition to the rendered principal views, this type of plot does provide a hint on how many nodes may be reasonable to choose for a specific category and training set. For many classes, using more than six views seems only to lead to the creation of experts with quite small positive training sets. And it is not computationally expensive to explore different parameters for m at this stage: On a modern Intel Core i7 processor with 3.2 GHz the lbTreeGNG and training set splitting takes a few seconds. In the next section I will explain how the extracted principal views can be used in order to learn view-tuned matching kernels for the different categories.

Chapter 5

View-tuned Kernel

This chapter presents a method for learning view-tuned approximate partial matching kernel from hierarchical Growing Neural Gases. For the sake of brevity, these kernels are called *view-tuned kernels* within this thesis. Major parts of this chapter have also been published in [105]. Section 5.1 provides the basic intuition behind view-tuned kernels. Subsequently, section 5.2 formally introduces the optimal partial matching and empirically shows that lbTreeGNGs can be used robustly to approximate this matching. Section 5.3 introduces the view-tuned approximate partial matching and empirically shows that it improves the separability of the classes. Finally, section 5.4 employs the extracted principal views of the last chapter and discusses the kernel learning results for the 10 domestic object categories. For an introduction to the limited branching tree Growing Neural Gas (lbTreeGNG) please see chapter A in the appendix.

5.1 Introduction

As already mentioned in the related work chapter, local visual features are often employed to compute the similarity between images or image regions. In doing so, an image can be represented by a set of d-dimensional real descriptors, e.g. SURF descriptors [7], where each vector robustly represents the distribution of gradients in a local image patch. Thus, comparing two images can be considered as the problem of comparing to vector sets of unequal cardinality. The optimal partial

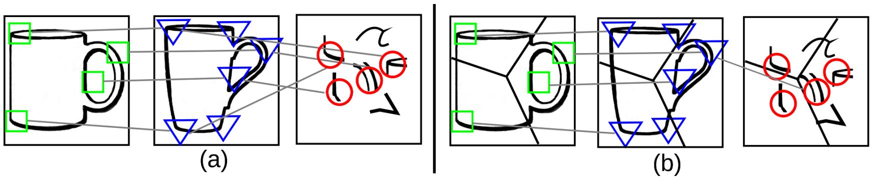


Figure 5.1: (a) Three different images are shown: two cups and one “abstract” image. The cup in the middle is compared to the other two images. The green squares, blue triangles and red circles depict some local image regions for which visual features are computed. The gray lines sketch the alignment of the feature sets with the minimal cost. (b) The basic idea of the work is to learn a (hierarchical) decomposition of the image plane (indicated by black lines) and to match features only locally in the Voronoi cells. In doing so, the cup and the abstract image get more dissimilar, because only one of the global minimal alignments is preserved when taking the positions of the local features into account.

matching (detailed in section 5.2) aligns the elements of the smaller set to the elements of the other set, such that the pairwise distances are minimised. The basic notion is sketched in figure 5.1 (a). As shown by [80], the optimal partial matching cost between two sets of local visual features is often a good similarity measure between the images. However, since the feature sets are unordered – like in the BOW approach – the sets' similarity can be high despite a dissimilar composition of the parts, as seen in figure 5.1 (a) when comparing the cup and the abstract image. In this thesis, a hierarchical Growing Neural Gas, namely the IbTreeGNG [104] is used as an efficient Mercer kernel that approximates the optimal partial matching cost. Further, this unordered set matching idea is extended and a kernel that matches sets of features in a more structured way is introduced. To that end, an additional hierarchical codebook (IbTreeGNG) is trained in the $2D$ image plane to build a view-tuned matching kernel that takes the relative positions of the features into account. The idea is illustrated in figure 5.1 (b).

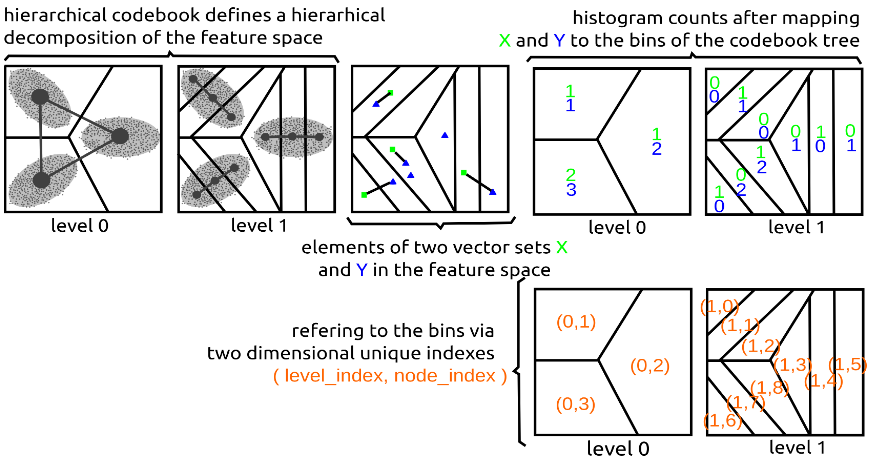

hierarchical codebook defines a hierarchical decomposition of the feature space
histogram counts after mapping X and Y to the bins of the codebook tree
level 0
level 1
level 0
level 1
elements of two vector sets X and Y in the feature space
refering to the bins via two dimensional unique indexes (level_index, node_index)
(0,1)
(0,2)
(0,3)
(1,0)
(1,1)
(1,2)
(1,3)
(1,4)
(1,5)
(1,6)
(1,7)
(1,8)
level 0
level 1

Figure 5.2: This figure sketches a hierarchical codebook tree (IbTreeGNG) in a feature space $\mathcal{F}$ with a branching factor $k = 3$ and a depth $L = 2$. Further, two feature sets X and Y are painted in the feature space and the histogram counts for the tree bins are shown. Finally, a two-dimensional indexing scheme for the nodes is presented.

5.2 Approximate Partial Matching with IbTreeGNGs

First, the optimal partial matching and the original vocabulary-guided pyramid match are introduced, as presented in [80]. Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ denote two sets of local visual features extracted from two different images or image regions. In this connection, let be $m \leq n$ and all $x_i, y_j \in \mathcal{F} = \mathbb{R}^d$. A *partial matching* is defined as $\mathcal{M}(X, Y; \pi) = \{(x_1, y_{\pi_1}), \ldots, (x_m, y_{\pi_m})\}$, where $\pi \colon \{1, \ldots, m\} \longrightarrow \{1, \ldots, n\}$ is an injective mapping of the indexes. The *cost of a partial matching* is defined as $\mathcal{C}(\mathcal{M}(X, Y; \pi)) = \sum_{x_i \in X} \|x_i - y_{\pi_i}\|_2$. An *optimal partial match* π^* is given by the assignment with the minimal matching cost, i.e. $\pi^* = argmin_{\pi}\ \mathcal{C}(\mathcal{M}(X, Y; \pi))$. The cost of the optimal partial matching can be used to measure the similarity between two sets and it defines an alignment of the sets' elements. The computation of the optimal partial matching is an instance of the assignment problem and can be solved optimally with a modified version of the Hungarian method [106] in $O(n^3)$. Before the vocabulary-guided kernel is stated, which approximates the optimal partial matching, it is important to discuss how a vector set can be represented by a histogram over a hierarchical codebook tree. Given a hierarchical codebook in a high dimensional feature space $\mathcal{F}$ that was trained on a representative training set. Figure 5.2 illustrates a codebook with $L = 2$ levels and a tree branching factor of $k = 3$. One way to assign indexes to the nodes of a tree is by using unique two-dimensional indexes (i, j), where i is the level of the node and j is a unique node number in the level i. This indexing scheme is also sketched in figure 5.2. Note that, to enhance the understandability, there are some changes in the notation of [105]. When a feature $x \in X$ is encoded by the tree, it is passed down the tree. This means, starting at level 0, the node with the nearest reference vector is found and then the vector is passed to its successor. Here, again the nearest node is found and the input vector moves to its successor. This is repeated until a leaf node is reached. Thus each input vector moves along a path of nearest nodes from level 0 to level $L - 1$. For each node the input vector passes, the histogram count of the respective bin is increased by 1. Examples for encoding two sets can be seen in figure 5.2.

In [80], the codebook is created via hierarchal k-means and the (original) vocabulary-guided pyramid match is defined as $\mathcal{C}(\Psi(X), \Psi(Y)) =$

$$\sum_{i=0}^{L-1} \sum_{j=1}^{k^{i+1}} w_{ij} [\overbrace{\underbrace{min(n_{ij}(X), n_{ij}(Y))}_{\text{\# matches in bin i,j}} - \underbrace{\sum_{h=1}^{k} min(c_h(n_{ij}(X)), c_h(n_{ij}(Y)))}_{\text{\# matches in bin i,j's children}}}^{\text{\# new matches in bin j at level i}}]. \tag{5.1}$$

Thereby, $\Psi(X) = [H_0(X), \ldots, H_{L-1}(X)]$ is a vector of histograms, where each histogram $H_i(X) = [< p, n, d >_1, \ldots, < p, n, d >_{k^{i+1}}]$ is a k^{i+1} dimensional histogram encoding X in the level i of the codebook tree. Here, p is denotes the bin index, n denotes the histogram count and d the maximal distance to the center. The n_{ij} refers to the j-th histogram count in $H_i(X)$. Further, $c_h(n_{ij})$ refers to the count of the child node of the node associated to n_{ij}. The authors in [80] show that $\mathcal{C}(\Psi(X), \Psi(Y))$ yields an efficient approximation of the optimal partial matching score with linear time complexity w.r.t. n. The authors suggest a proof showing that $\mathcal{C}(\Psi(X), \Psi(Y))$ is a Mercer kernel, which makes it usable in kernel-based classification methods like SVMs. However, in my view the argument made in [80] for $\mathcal{C}(\Psi(X), \Psi(Y))$ being a kernel does not hold. The authors rewrite the original $\mathcal{C}(.,.)$ equation as

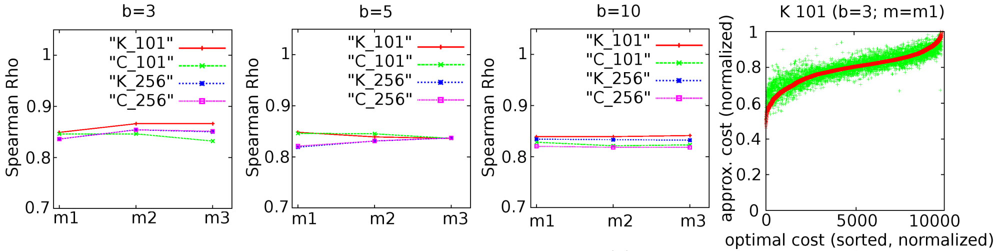
$$\mathcal{C}(\Psi(X), \Psi(Y)) = \sum_{i=0}^{L-1} \sum_{j=1}^{k^{i+1}} (w_{ij} - p_{ij}) min(n_{ij}(X), n_{ij}(Y)) \tag{5.2}$$

without formally stating what p_{ij} is. But it can be inferred that p_{ij} is the following

$$p_{ij} = \frac{w_{ij} \sum_{h=1}^{k} min(c_h(n_{ij}(X)), c_h(n_{ij}(Y)))}{min(n_{ij}(X), n_{ij}(Y))} \tag{5.3}$$

to be equal to the original equation. The authors now argue that $\mathcal{C}(.,.)$ is a kernel for $w_{ij} \geq p_{ij}$, because the minimum intersection is a Mercer kernel [149] and kernels are closed under summation and scaling with a positive constant [186]. However, $(w_{ij} - p_{ij})$ can not be treated as a constant, because it be can seen in the above equation (5.3) that the term p_{ij} *does* depend on the input of the kernel and thus can not be constant, i.e. $p_{ij} = p_{ij}(\Psi(X), \Psi(Y))$. As I did not find a correction for the proof, I propose a simplified version that drops the subtraction of the second part in equation (5.1), which is what causes the problem. In fact, the subtraction only cleans the histogram counts by removing matches that have already been found in deeper levels of the tree. Having a Mercer kernel is important for the optimisation of the SVM, because the optimal point of the convex optimisation problem may not be found if the Gram matrix is *not* positive semi-definite and symmetric.

In this thesis, a limited branching Tree Growing Neural Gas (lbTreeGNG) [104] with a branching factor k and a depth of L is used to learn hierarchical codebooks. As known from the last chapter, the lbTreeGNG is an efficient hierarchical method for one-shot online vector quantisation, that keeps track of the topological structure of the input space and avoids over fitting. Figure 5.2 sketches a lbTreeGNG learned on the elliptic input distributions. Based on the histogram encoded feature sets $\Psi(X)$ and $\Psi(Y)$, the *(simplified) vocabulary-guided pyramid*



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

Figure 5.3: (a-c) Approximation quality of the original $\mathcal{C}(.,.)$ [80] and the simplified version $\mathcal{K}(.,.)$ w.r.t. the optimal partial matching under different parameter settings (for details see text). (d) Optimal costs versus approximated costs.

match kernel is defined as a weighted minimum intersection:

$$\mathcal{K}(\Psi(X), \Psi(Y)) = \sum_{i=0}^{L-1} \sum_{j=1}^{k^{i+1}} w_{ij} min(n_{ij}(X), n_{ij}(Y)) \qquad (5.4)$$

Here, w_{ij} is a node dependent constant weighting factor with $w_{ij} > 0 \forall i, j$. Similar to [80], the weights are set to $\frac{1}{d_{ij}}$, where d_{ij} is learned for each node j in the level i of the lbTreeGNG as the average distance of samples in the bin. $\mathcal{K}(\Psi(X), \Psi(Y))$ is a Mercer kernel, because the minimum intersection is a Mercer kernel [149] and kernels are closed under summation and scaling with a positive constant [186].

Experiments

The experiments investigate two things: First, I compare the simplified kernel $\mathcal{K}(.,.)$ and the originally matching $\mathcal{C}(.,.)$ with the optimal matching cost. Second, using the standard parameters of the lbTreeGNG as stated in [104], I show how strongly the matching performance of $\mathcal{K}(.,.)$ depends on the choice of the residual lbTreeGNG parameters m, b and the training set. The experiments are done on real image data from the ETH-80 database [117] as proposed in [80]. For each of 100 images, 256 128-dimensional SURF descriptors are computed on a grid. Then the resulting 100 descriptor sets are compared, which produces $10,000$ matching costs. As a baseline, the cost of the optimal assignment π^*

is computed (using GLPK [224]). As in [80], the quality of an approximation is assessed by the Spearman Rho, when comparing the optimal ranks to the approximative ranks. The Spearman Rho is defined as

$$\rho = 1 - \frac{6 \sum_{i=1}^{N} (i - \hat{r}(i))^2}{N(N^2 - 1))} \tag{5.5}$$

where i is the optimal rank and $\hat{r}(i)$ is the corresponding rank in the approximation. The Spearman Rho of $\mathcal{K}(.,.)$ is compared with the original $\mathcal{C}(.,.)$ for different parameters $m \in \{m_1 = .15, m_2 = .015, m_3 = .0015\}$, $b \in \{3, 5, 10\}$, and $S \in \{101, 256\}$. Here, S denotes the training set, which is used to train the IbTreeGNG and 101 refers to the CalTech 101 database [48] (9144 images) and 256 refers to the CalTech 256 database [82] (30607 images).

Results

The results are summarised in figure 5.3. As can be seen, the introduced simplification is not problematic, because the approximation performance of $\mathcal{K}(.,.)$ and $\mathcal{C}(.,.)$ is very close. In addition, the results suggest that the performance is very robust to the concrete choice of the parameters, because there are no large differences w.r.t. the Spearman Rho. The fact that the Spearman Rho in the experiments is approx. .05 smaller than in [80] (with global weights) may be a consequence either of using SURF instead of SIFT features, or of not training the codebook on the test images.

5.3 View-tuned Approximate Partial Matching

This section introduces the view-tuned vocabulary-guided pyramid match kernel. The kernel employs two hierarchical codebooks (IbTreeGNGs): one class-independent GNG in the d-dimensional feature space $\mathcal{F}$ (as explained in the last section) and one class- and view-specific GNG in the 2D image plane. To learn the $2D$ IbTreeGNG, points are sampled from the unit image plane $\xi \in [0, 1]^2$ according to a distribution $P(\xi)$. Thereby, $P(\xi)$ should have high values for locations where relevant object parts occur, w.r.t. a specific feature type and view-point. At this

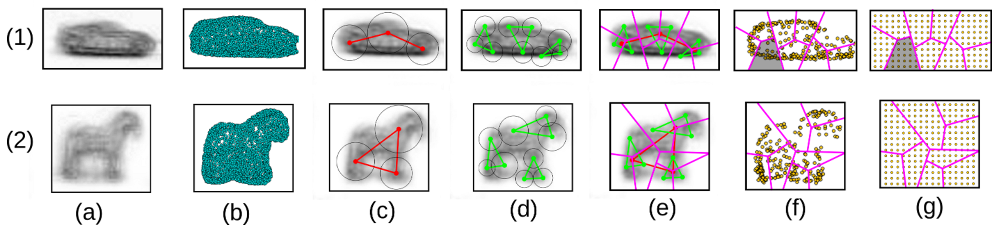


Figure 5.4: Two examples of view-tuned kernels for the categories car and horse (explanation given in the text).

point, the principal views from the last chapter come into play. By design, an extracted principal view can be employed as a $P(\xi)$ distribution, when sampling 2D pixel locations

$$\xi = (\frac{x}{w}, \frac{y}{h}) \sim \left[\frac{255 - g(x, y)}{255wh}\right] \tag{5.6}$$

for a principal view of width w and height h. Here, x and y are the pixel locations and $g(x, y)$ is the inverted gray-scale value at the pixel coordinates of the normalised averaged gradient magnitude image. In doing so, pixels from darker regions are sampled more often than pixels from brighter regions. Remember, that the results for the domestic objects categories are presented in section 5.4. Here, I want to stay at the image data from ETH-80 database, because it has been used by [80] and has already been employed for the evaluation in the last section. Figure 5.4 column (a) shows two principal views of cars and horses from the ETH-80 database. Column (b) illustrates points sampled according to those distributions and the coordinates are used as 2D input vectors to train the 2D IbTreeGNG. Further, column (c) and (d) show the first and the second level of the trained IbTreeGNGs with a branching factor $b = 3$. The thin circles indicate the average point distance estimation, as known from the weighting in $\mathcal{K}(.,.)$. Subsequently, column (e) presents the hierarchal Voronoi tessellation induced by the 2D codebook trees. Finally, column (f) shows key point locations that are also sampled according to $P(\xi)$ and which are associated with the class-specific 2D IbTreeGNGs. When comparing images for a specific class, e.g. cars, the associated key point set is used and at each key point location a descriptor is computed. An alternative way is to define key point locations on a static grid, as sketched in figure 5.4 column (g). When comparing two descriptor sets, the learned 2D tessellation is employed as follows to incorporate the relative locations of the local features.

Given a trained hierarchical codebook (IbTreeGNG) in 2D, which is associated with a class and view ω, and with a depth R and a branching factor b. This IbTreeGNG is associated with the above-mentioned key points set and each key point can be assigned to a hierarchical bin (as indicated in figure 5.4 column (f) and (g)). Following the same principle as in the last section, each node q in the level p of the 2D tree can be associated with a histogram $\Psi_{pq}(X^{\omega}_{pq})$ over the tree in feature space, where X^{ω}_{pq} is the set of descriptors from the key points that belong to bin (p, q) in the 2D IbTreeGNG of class ω. In other words, each local feature set in the 2D Voronoi cells (descriptors computed for the yellow dots in the gray region in figure 5.4 (f.1)) is represented by a histogram over a the codebook tree in the feature space. Again, this single histograms can be concatenated into one large vector, i.e.

$$\Xi^{\omega}(X) = \left[\Psi_{0,1}(X^{\omega}_{0,1}), \ldots, \Psi_{R-1,b^R}(X^{\omega}_{R-1,b^R})\right]. \tag{5.7}$$

By using this notation, the *view-tuned vocabulary-guided pyramid match kernel* is defined as

$$\mathcal{T}(\Xi^{\omega}(X), \Xi^{\omega}(Y)) = \sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \tilde{w}_{pq} \, \mathcal{K}\big(\Psi_{pq}(X^{\omega}_{pq}), \Psi_{pq}(Y^{\omega}_{pq})\big). \tag{5.8}$$

In this equation, $\tilde{w}_{pq}$ is a node-dependent constant weighting factor with $\tilde{w}_{pq} > 0 \forall p, q$. As before, the weights are set to $\frac{1}{\tilde{d}_{pq}}$, where $\tilde{d}_{pq}$ is learned for each node of the 2D IbTreeGNG as the average distance of samples in the bin q in level p. Since $\mathcal{K}(.,.)$ is a Mercer kernel, the same argument a made for (5.4) can be used to prove that (5.8) is a kernel as well, i.e. kernels are closed under summation and scaling with a positive constant [186].

The proposed kernel $\mathcal{T}(.,.)$ implements a tuned fine-to-coarse matching of local image parts for similar views of instances of a class ω. The kernel locally matches BOW like representations with an approximate partial matching, while paying attention to the structure of the object and the relative positions of the BOW-based parts. Due to the hierarchal form of the matching in $2D$, the proposed method is intended to be robust to small deformations and rotations. In addition, the minimum intersection is intended to be robust to clutter and occlusion, to a certain extent. Due to the fact that the presented matching is a Mercer kernel, it is directly suitable for kernel-based machine learning techniques, like the Support Vector Machines (e.g. [186]). Beyond, the matching kernel is theoretically independent of the particular type of the local feature, i.e. color-, appearance- or shape-based local descriptors could be used.

The space complexities are as follows: $O(b^R)$ for a saving a pyramid structure in $2D$, $O(k^L)$ for a saving pyramid structure in $\mathcal{F}$, $O(nL)$ for storing a sparse histogram $\Psi(.)$, and $O(RnL)$ for storing a $\Xi^\omega(.)$. The time complexities w.r.t. distance computations are as follows: $O(nL)$ for creating a histogram $\Psi(.)$ for $\mathcal{K}(.,.)$ and $O(2RnL)$ for creating both tree embeddings $\Xi^\omega(X)$ and $\Xi^\omega(Y)$ for $\mathcal{T}(.,.)$. Thus, the time complexity of the proposed kernel is still linear w.r.t. n and allows efficient matching of images or regions.

Experiments

The experiments presented here are an extended version of the experiments in [105]. The basic idea is to evaluate a learned kernel by its ability to separate the classes, as has been employed by other authors to find optimal parameters for learned kernels (e.g. cf. [220], [198]). In this notion, the optimal kernel parameters maximised the class separability in the induced feature space. Here, the class separability is formally measured via averages of intra- and inter-class sample distances. Intuitively, a good kernel would be given if all class elements are very similar to each other and at the same time very dissimilar to the elements of all other classes. This can be formalised as follows.

Let there be a K classes $\Omega = \{\omega_1, \dots, \omega_K\}$ and one set of samples for each class $Z = [Z(\omega_1), \dots, Z(\omega_K)]$. The *average in-class distance* for a class ω_i is given by

$$D_k(\omega_i, Z) = \frac{1}{|Z(\omega_i)|^2} \sum_{x \in Z(\omega_i)} \sum_{y \in Z(\omega_i)} \tilde{k}(x, y) \qquad (5.9)$$

where $|\cdot|$ denotes the cardinality of a set and $\tilde{k}(.,.)$ is a normalised[1] version of a kernel $k(.,.)$ such that $\tilde{k}(.,.) \in [0,1]$. The *average between-class distance* between a class ω_i and a class ω_j is given by

$$D_k(\omega_i, \omega_j, Z) = \frac{1}{|Z(\omega_i)||Z(\omega_j)|} \sum_{x \in Z(\omega_i)} \sum_{y \in Z(\omega_j)} \tilde{k}(x, y). \tag{5.10}$$

Further, the average between-class distances can be averaged over the classes by

$$\tilde{D}_k(\omega_i, Z) = \frac{1}{K-1} \sum_{j=1, j \neq i}^{K} D_k(\omega_i, \omega_j, Z). \tag{5.11}$$

The *separability score* for a kernel $k(.,.)$ can now be written as

$$S_k(Z) = \frac{1}{K} \sum_{i=1}^{K} \| D_k(\omega_i, Z) - \tilde{D}_k(\omega_i, Z) \| \quad \in [0,1], \tag{5.12}$$

where $\|.\|$ denotes the absolute value. As stated before, the best kernel (or kernel parameters) from a set of learned kernels is the one that maximises the separability score.

In the conducted experiments different view-tuned kernels should be compared, and an obvious question is, what are good values for the branching factor b and the depth R of the 2D codebook tree. Therefore, the learned view-tuned kernels $\mathcal{T}(.,.)$ with different values for $b \in \{3, 4, 5, 6\}$ and for $R \in \{2, 3, 4\}$ are compared in the experiments to find an optimal global setting. The other lbTreeGNG parameters are set to the default [104]. Using more than 4 levels in the presented setup does not make sense from a theoretical point of view, because b^R with $R > 4$ (and $b > 3$) will result in more Voronoi-cells than key points, which then results in no matches in deeper levels. If no matches (or very few) are found in deep levels of the tree, the corresponding matching score for that level will be very small and only marginally affect the overall score. Further, two types of local visual feature are compared: 128-dimensional SURF features [7] and 72-dimensional HOG features

---

[1] In the case of a weighted minimum intersection over a fixed number of key points, the normalisation can e.g. be achieved by dividing by the highest possible intersection value.

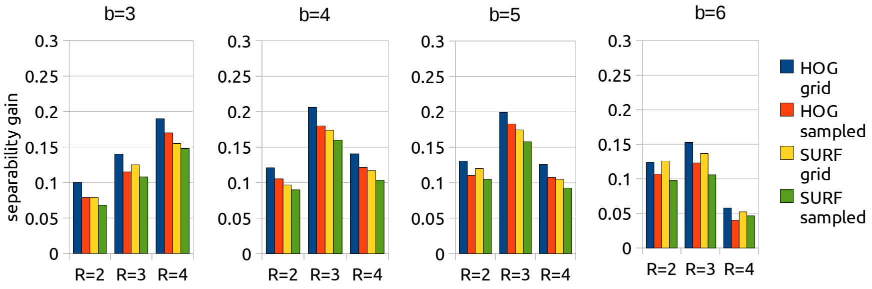


Figure 5.5: This figure shows the gain of the separability score for the ETH-80 classes under view-tuned kernels with different parameter settings compared with the baseline, i.e. the simplified vocabulary-guided pyramid match.

[37]. This is a special HOG feature, that uses 3 × 3 blocks and 8 gradient directions (for a more through explaination see figure B.1 in the appendix). Basically, both types of feature encode the gradient structure around a key point location. However, the major difference is that SURF descriptors are rotation invariant and HOG descriptors are not. Moreover, the experiment compares sampled key point locations (as seen in figure 5.4 column (f)) to key point locations defined on a static grid (cf. figure 5.4 column (g)). Here, for a 128 × 128 pixel image up to $30 \times 30 = 900$ key points locations are employed. However, since the principal view images are cropped, the actual number of key points can be lower. As a baseline, the separability score of the simplified vocabulary guided pyramid match kernel $\mathcal{K}(.,.)$ is used. For each class in the in the ETH-80 database, the side views (090 images) of the objects are used to compose the respective training sets and to learn principal views. Then the separability score is computed under a set of parameters and compared to the baseline score. That means, all HOG scores are compared with the baseline $\mathcal{K}_{\mathtt{HOG}}(.,.)$ and the SURF scores are compared with $\mathcal{K}_{\mathtt{SURF}}(.,.)$. As discussed in the last section, the lbTreeGNGs in the feature space $\mathcal{F}$ are learned on the respective descriptors from the CalTech 101 database [48] with default lbTreeGNG parameters [104], and a branching factor of 10, as well as an error parameters of .015.

Results

Figure 5.5 shows the absolute gain of the separability score compared to the baseline scores, when using view-tuned kernels under different parameters. The first interesting aspect that can be observed is that – independent of the type of feature or key point generation method – all view-tuned kernels (with levels $R \in \{2, 3, 4\}$) provide a clear gain in the class discrimination compared with the simplified vocabulary guided pyramid match $\mathcal{K}(.,.)$. These results are in line with the observations of various other authors, which state that using information about the spatial structure makes it possible to significantly enhance bag-based kernels. Another aspect is that in the majority of runs, the HOG features work better than the SURF features. A possible explanation is, that in the scenario of a view-tuned kernel the additional rotation invariance removes information from the representation, that is actually beneficial for matching objects under similar views. Further, the experiments show that in the view-tuned matching scenario the usage of key points defined on a static grid usually works slightly better than using sampled key point locations. One possible explanation for this is that the grid-based representations allow a better encoding of the objects' shape, texture and background, since each location in the images is covered and key points can not get too dense. Taking a look at the performance w.r.t. the number of tree levels R, it is clear that using too few levels is not beneficial. Also for $b > 3$ it's obvious that that too many levels do not enhance the performance either. The best performance can be observed with $R = 3$. One explanation of this may be as follows. On the one hand, having too few levels creates large bags of features, which representations have a large amount of flexibility. On the other hand, having too small Voronoi-cells results in tiny feature sets which can not be matched good in the deepest level of the tree. Thus, with R set to a value "in the middle", the feature sets of the Voronoi cells are reasonably large w.r.t. flexibility and comparability. Another interesting parameter is the branching factor b of the 2D IbTreeGNG. Note, that the branching factor directly influences the size of the Voronoi cells in the different levels. That means that the higher b is set, the smaller the Voronoi cells become. As stated before, if the Voronoi cells get too small the corresponding key points sets get too small and are hard to match. Again,Voronoi cells that are too large can also degrade performance, as larger unordered BOW sets are matched. Also here, good choices of the parameter – in conjunction with the depth of the codebook tree – seem to be in the middle of the tested spectrum, i.e. $b = 4, 5$. Finally, among different good working parameter settings, the best global parameter settings are $[b = 4; R = 3;$ `HOG_grid`$]$. Therefore, these parameters will be used for the training of view-tuned kernels and the matching in the subsequent chapters.

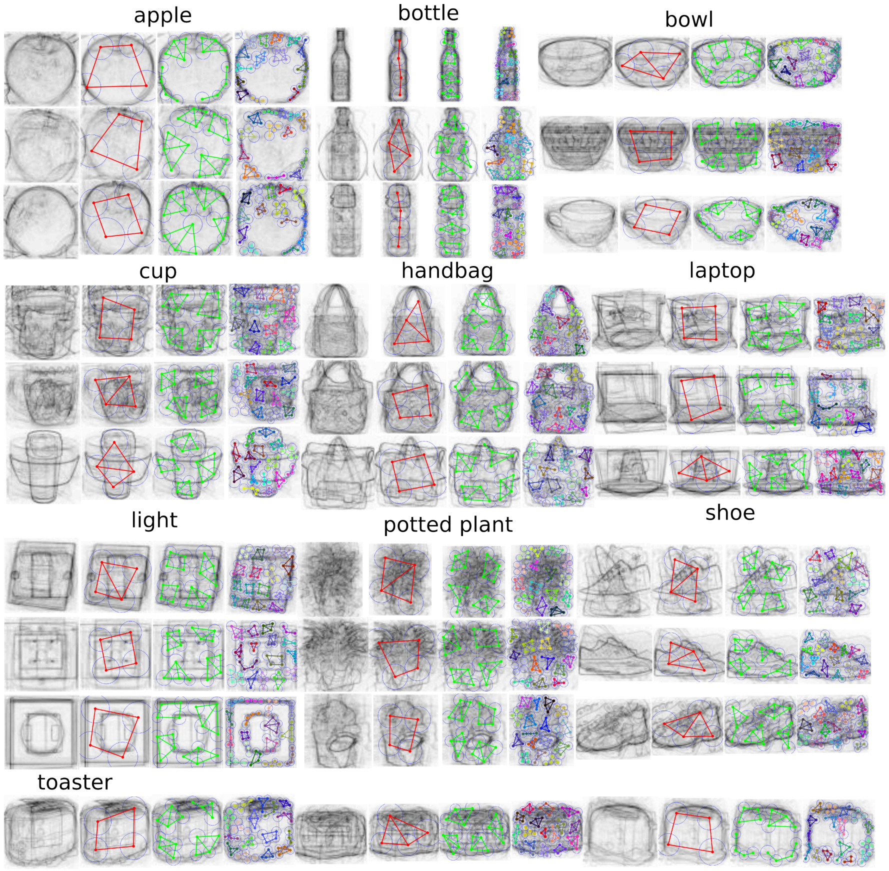


Figure 5.6: This figures shows the learned view-tuned kernel for the three principal views of every class.

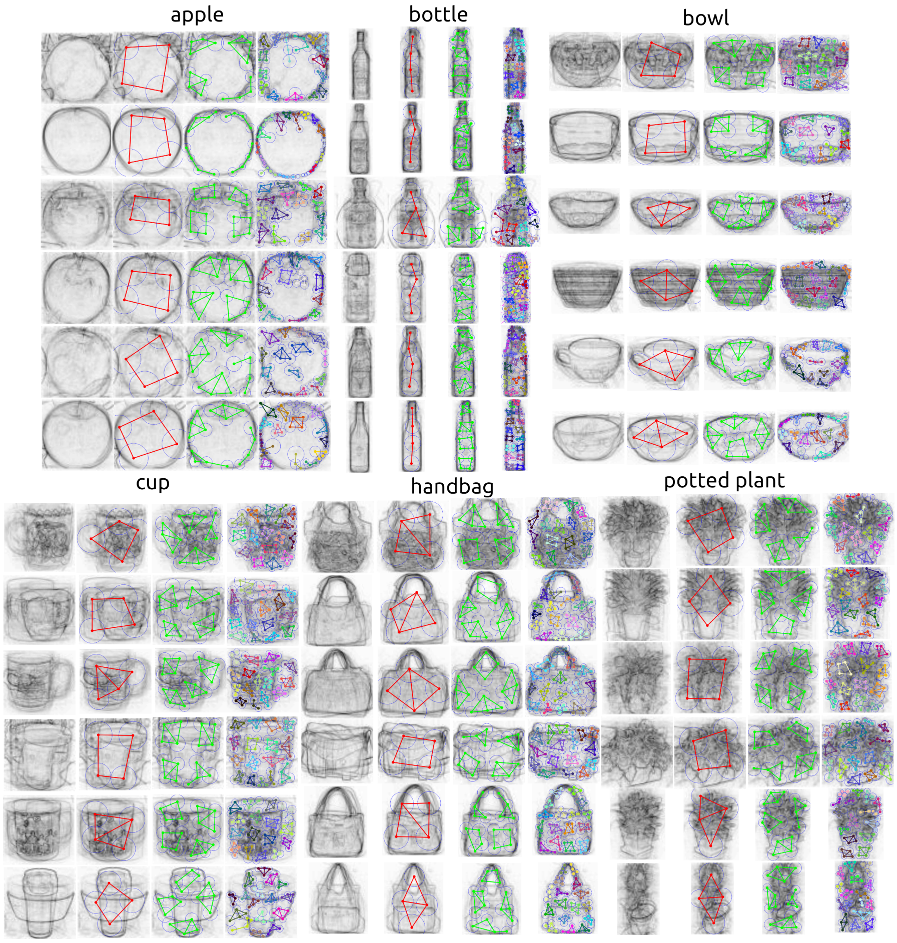


Figure 5.7: This figures shows the learned view-tuned kernel for the six principal views of the classes apple, bottle, bowl, cup, handbag and potted plant.

5.4 Results and Discussion

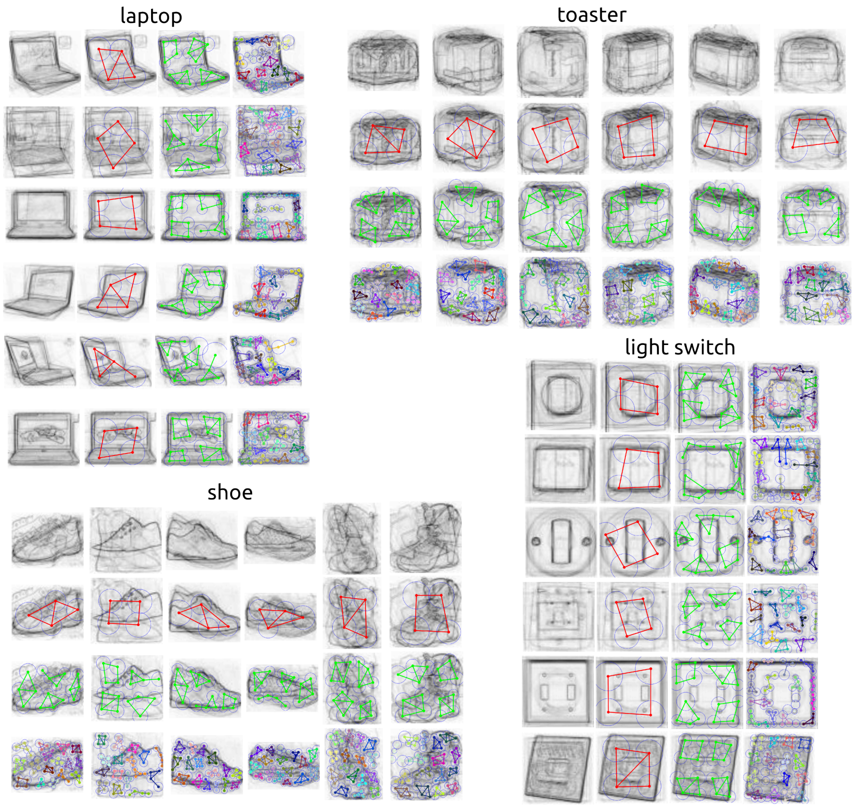


Figure 5.8: This figures shows the learned view-tuned kernel for the six principal views of the classes laptop, toaster, light switch and shoe.

As mentioned before, this chapter presents and discusses the learned view-tuned kernels for the ten domestic object categories of this thesis. For each principal view of each object class a multi-layered 2D codebook tree is learned, as presented in the last section. In this process the found optimal 2D lbTreeGNG parameters ($[b = 4; R = 3; \texttt{HOG\_grid}]$) are used for all classes, which produces comparable results. The results for three principal views can be found in figure 5.6. Moreover, the results for six principal views are shown in figure 5.7 (apple, bottle, bowl, cup, handbag and potted plant) and in figure 5.8 (laptop, toaster, shoe and light switch). Note that the results for nine principal views are not included, because they are structurally very similar to the results of six and three principal views and would require a lot of space ($9 \cdot 4 \cdot 10 = 360$ images). In the result figures, a principal view is presented at the beginning. Then the corresponding first and second levels of the learned 2D codebook are sketched in red and green respectively. Finally, the corresponding third level of the tree is painted in random colors.

Overall, the learned view-tuned kernels realize a hierarchal decomposition from rough to fine of relevant object parts. The first level of the tree decomposes the overall principal view into four regions. In doing so, the lbTreeGNG adapts to the underlying structure for the gradient and texture of the principal view, e.g. the first level for bottles and shoes look very different. But also for some views, the learned first levels look quite similar (e.g. compare the first levels of bottle). The second level refines this initial decomposition and introduces a tessellation of those regions, again into four cells each. For untextured objects, like apple, these regions roughly describe parts of the shape. But for more textured classes as well the second level introduces a rough notion of sub-parts', e.g. the four new subregions decompose a plant pot or a handle of handbags. Finally, the third levels make it possible to capture details of the sub-parts shape and texture during matching. This supports the discrimination of roughly similar looking parts. As stated before, the learned kernel only realises a view-tuning. For instance, take a look at the first view-tuned kernel for cup in figure 5.7. It is clear that the kernel is tuned towards a left-handed cup. However, due to the structure of the kernel it is also be possible to match a right-handed cup with this kernel. It may not be as good as the view-tuned kernel for right-handed cups, but – as turns out when used with an SVM classifier in the next chapter – the discrimination is often good enough to separate in-class and background images.

Chapter 6

View-tuned Support Vector Machines

This chapter presents the way in which view-tuned kernels can efficiently be used in the learning framework of Support Vector Machines (SVMs) [33] for the purpose of object detection. In this context the detection of objects is considered to be a two-class classification problem in a sliding window scheme (cf. chapter 2). One class represents the object category (using the acquired in-class images) and the other class subsumes negative sample images from the background class or other classes. When a window slides over the image, a trained SVM is used to classify whether or not an image region contains an object or not. First, the C-SVM formulation and the optimisation of unbalanced learning problems is presented in section 6.1. Here, unbalanced means that it is easy to get a large number of negative images, while having a fixed number of positive in-class images. For example, the background patches can easily be sampled from a database of images. This kind of learning problems, where the number of samples for one class is much higher than for the other class, need special training to achieve reasonable solutions and without giving preference to the bigger class. After that, section 6.2 discusses how to extract the local visual features efficiently, by using aligned key points and window offsets to avoid multiple computations of the local descriptors. Once there is a trained model, it can be applied to images as a detector. Thereby, the runtime of the detector is critically dependent on the number of support vectors in the model, i.e. the time complexity for classifying an input vector is $O(MD)$ where M is the number of support vectors and D is the dimensionality of the feature space. This problem is addressed in section 6.3. Using the results of [129], I show that the decision function for view-tuned kernels can be rewritten in such a way, that the classification complexity becomes independent from the number of support vectors (i.e. $O(D)$). Using a special caching scheme, it is possible to compute the exact value of the decision function very efficiently. On the one hand, this result supports the efficient application of view-tuned SVMs, and on the other it supports the application of large models. In a common object detection scenario, the majority of windows are non-class regions, and in order to classify them, a full kernel needs to be evaluated on each of those regions. Due to the introduced computation of the decision function (section 6.3) these kernel evaluations are efficient, but still not free of cost. To minimize the number of computations further, cascades are brought into play. The basic idea is to use a simple classifier first and to use more complex classifiers only on "interesting" image regions, that have been classified as in-class by the simple classifiers. In this process, large parts of the image are classified with only a cheap classifier and the evaluation of more complex classifiers is done only on few image regions. In section 6.4, I show how view-tuned kernels can be used in a cascaded scheme. In this scheme, deeper levels of the kernels are evaluated only on promising image regions which makes the detection with view-tuned kernel SVMs even more efficient. Further, as already mentioned, the learning problem is unbalanced and in the detection scenario negative images could be gathered. As has been shown (e.g. in [57]), to get a good detector it is very important to train on negative samples that are difficult to classify, i.e. near the decision boundary. However, initially it is not clear which negative samples from the giant pool of possible negative patches are hard negatives. Therefore, it has been suggested that a classifier should be trained on simple negative examples first and that the simple detector should be applied to find more difficult negative images (e.g. [57]). This procedure is then repeated several times in order to get very hard negatives. Section 6.5 explains the sequential discriminative optimisation that is employed for training strong view-tuned SVM detectors. So, a *view tuned Support Vector*

Machine is a C-SVM used with a view tuned kernel that is trained discriminatively as an efficient cascaded classifier for detection. Sample results are discussed in some of the relevant sections. However, note that the detection performance of the trained classifiers is evaluated as a part of the detection ensembles in the next chapter. Finally, section 6.6 concludes this chapter.

6.1 SVM Formulation and Optimisation

This section presents the formulation of two-class C-SVMs [15][33] and their optimisation in an unbalanced learning problem for training of view-tuned classifiers. For a more comprehensive introduction of Support Vector Machines e.g. confer [35].

Let $x_i \in \mathbb{R}^D$ denote the training vectors for the learning problem, with $i = 1, \ldots, N$. Further, let the class indication be denoted by $y \in \mathbb{R}^N$ with $y_i \in \{1, -1\}$. The Support Vector Machine maps an input vector $x \in \mathbb{R}^D$ with a non-linear function $\phi(x)$ into a high dimensional feature space and classifies the vectors there with a linear decision boundary $f(x) = w^T\phi(x) + b$. The resulting decision boundary in the input space can be highly non-linear and can be represented as a linear combination of the transformed training examples, i.e.:

$$f(x) = w^T\phi(x) + b = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)^T \phi(x) + b = \sum_{i=1}^{N} \alpha_i y_i k(x_i, x) + b. \tag{6.1}$$

As seen in the last term, it is possible to use a kernel function $k(.,.)$ to compute the dot product of the two transformed samples without explicitly computing the expensive transformations $\phi(x_i)$ and $\phi(x)$. The decision boundary is chosen such that the error on the training data and the generalisation error are optimised at the same time. Based on learning theory [211], these errors are minimised, when the margin between training examples and the class boundary is maximised. When the margin is optimising, the capacity of the classifier is tuned and overfitting is avoided. Formally, the C-SVM [15] [33] optimisation problem for two classes is given as follows.

$$
\begin{aligned}
\underset{w,b,\xi}{\text{minimise}} \quad & \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i \\
\text{subject to} \quad & y_i(w^T\phi(x_i)+b) \geq 1-\xi_i,\ i=1,\dots,N \\
& \xi_i \geq 0,\ i=1,\dots,N
\end{aligned}
\tag{6.2}
$$

Here, $C > 0$ is a regularisation parameter and the slack variables ξ_i are introduced to handle the optimisation for non-linear-separable classes. This minimisation problem is convex and has linear constraints. By using the Kuhn-Tucker-Theorem [69], the optimisation problem is easier to solve in the dual form, and can be written as follows

$$
\begin{aligned}
\underset{\alpha}{\text{maximise}} \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(x_i,x_j) \\
\text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& \alpha_i \geq 0,\ i=1,\dots,N.
\end{aligned}
\tag{6.3}
$$

An obvious issue for the optimisation of this problem is that the kernel matrix $K_{i,j} = k(x_i, x_i)$ is a dense matrix with N^2 elements, which may be too large to be stored in memory. Therefore, to solve the problem (6.3) for large kernel matrices, decomposition methods have been introduced. A decomposition method defines a working set B as a subset of all α variables, and modifies only this subset in one iteration. Thus, only some columns of the matrix are needed. The so called Sequential Minimal Optimisation (SMO) [164] considers minimal working sets of size two. In this thesis, I use the optimisation included in `LIBSVM` [71] [27], that employs a special SMO-like decomposition method (details in [47]). During the optimisation, the current working set α_i and α_j is chosen heuristically, while the resulting optimisation problem is solved analytically. After optimising problem (6.3), the optimal w satisfies $w = \sum_{i=1}^{N} y_i\alpha_i\phi(x_i)$ and the samples with $\alpha_i \neq 0$ are called support vectors. For a new sample x, the predicted class is given by the sign of the decision function, i.e. $sgn(f(x))$.

Using the presented scheme with view-tuned kernels is straight forward. For example, say a classifier using the first view-tuned kernel of apples (`apple_1`) should be learned. Here, I assume the IbTreeGNG in the `HOG` feature space has already been learned. The positive training set is composed of the apple images that have been assigned to the corresponding first principal view of apple, as described in chapter 4. At first, the negative images can simply be sampled as random patches from an image dataset, when it is quite certain that no apples are contained in the images. The training images of the apple class are scaled to the width and the height of the according principal view, and the training images of the other class are directly sampled with that size. For comparing two images $X_i^{\mathtt{IMG}}$ and $X_j^{\mathtt{IMG}}$, the key point grid associated to the `apple_1` principal view is used to generate the `HOG` descriptors sets X_i and X_j. By using the 2D IbTreeGNG and considering the key point locations, the `HOG` descriptors sets can be mapped to histogram vectors $\Xi^{\mathtt{apple\_1}}(X_i)$ and $\Xi^{\mathtt{apple\_1}}(X_j)$. So, in the presented SVM formulation the kernel function for comparing two samples is the view-tuned kernel for a specific class and view that matches the encoded descriptor sets of the sample images, i.e.:

$$x_i \equiv \Xi^{\omega}(X_i), \quad x_j \equiv \Xi^{\omega}(X_j), \quad k(x_i, x_j) \equiv \mathcal{T}(\Xi^{\omega}(X_i), \Xi^{\omega}(X_j)). \tag{6.4}$$

The optimal value for the meta-parameters C is found by using a 10-fold cross-validation [11]. That means, the dataset is split into 10 chunks, where nine of the chunks are used for training with a concrete parameter C and one chunk is used for testing. To estimate the performance under a concrete C, the performances of the 10 different test runs are averaged. The values for C are chosen in a two-step process. First, a number of $\mathtt{C}_1$ values is sampled in a broad range $[R_1, R_2]$. Let C^* denote the optimal C parameter in that range. In a second step, an additional number of $\mathtt{C}_2$ samples is drawn from $[C^* - R_3, C^* + R_3]$ to optimise the parameter further. As concrete values for the C-optimisation, the first interval is set to $[R_1 = .000001, R_2 = 1000]$ (because the maximum has always been found in this interval), and a total number of $\mathtt{C}_1 = 100$ samples are used. For each local sub interval, 10 points are sampled, e.g. in $[.000001, .00001]$, in $[.00001, .0001]$ and so forth. For the second interval a total number of $\mathtt{C}_2 = 25$ values is sampled in

$[\frac{C^*}{10}, C^* \cdot 10]$, where C^* is the optimal value found in the first interval. However, the parameters do not need to be chosen very carefully, since the score has not been observed to be peeked and good classifiers can be found for a broad range of C values.

What has not yet been addressed is what concrete performance measure is used for the optimisation of the C parameter. Training classifiers for object detection in the sliding window scheme is an unbalanced problem, i.e. there are many more negative examples available than positive examples. For example, let there be $1,000$ samples for the first principal view of apple and $9,000$ sampled non-apple patches. A common measure for optimising the C parameter for balanced problems is the precision $prec = \frac{tp}{tp+fp}$, where tp is the number of true positives and fp is the number of false positives. However, optimising C w.r.t. the precision value is very problematic for unbalanced problems. The optimisation will often choose a classifier that classifies all samples as belonging to the bigger class, because this model has a high precision, here $prec = \frac{9{,}000}{9{,}000+1{,}000} = .9$. In principle, there are two ways for tackling this problem. First, a balanced performance measure for optimising the C could be used. In the experiments in this thesis, the *balanced true positive ratio* of the classes $+$ and $-$ has been used, i.e.:

$$btpr = .5 \left[\frac{tp^+}{tp^+ + fn^+} + \frac{tp^-}{tp^- + fn^-} \right]. \tag{6.5}$$

In the case of the above example, the balanced true positive ration is $btpr = .5[\frac{0}{1{,}000} + \frac{9{,}000}{9{,}000}] = .5$. So an optimal score of 1 means that no errors are made in both classes. In the example, allowing nine errors for the bigger class counts equally to classifying one item of the smaller class correctly, i.e. $.5[\frac{1}{1{,}000} + \frac{8881}{9{,}000}] = .5[\frac{0}{1{,}000} + \frac{9{,}000}{9{,}000}]$. A second way to deal with unbalanced problems is to reformulate the SVM optimisation problem to have two regularisation parameters $C^+ > 0$ and $C^- > 0$, i.e. one for each class. For a C-SVM, the minimisation formulation can then be written as

$$\underset{w,b,\xi}{\text{minimise}} \quad \frac{1}{2} w^T w + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \tag{6.6}$$

subject to the same constraints as before in problem (6.2). The two C values are chosen in such a way that their values reflect the ratio of the training set sizes. For this thesis, I empirically compared both methods mentioned, in order to deal with unbalanced problems by using the balanced true positive ratio of the optimal parameters. As it turns out, for learning view-tuned kernels, the first method – i.e. using a C-SVM with a balanced performance measure – provided better results and is therefore used in all subsequent experiments. During the optimisation of the dual problem, the shrinking and caching options of `LIBSVM` [71] [27] were enabled. The caching stores some amount of the already computed kernel values for faster reuse. The shrinking technique tries to remove bounded elements from the optimisation problem to allow faster solving of smaller problems.

Up to this point the discussion has concerned the way how C-SVM with view-tuned kernels can be optimised. In the example, a randomly sampled negative image set was used, which results in a simple initial classifier. By using a better negative training set, a better classifier can be trained. Section 6.5 describes a method for sequentially training better classifiers by using hard negative examples (i.e. negative examples near the decision boundary). Using a simple classifier, the negative examples are harvested from a dataset of images. However, to harvest a negative training set of sufficient size, many image patches need to be processed. This procedure – and of course the context of a robotic application – demand both an efficient feature extraction and efficient evaluation of the decision function. Both aspects are addressed in the next sections.

6.2 Efficient Feature Extraction

This section discusses how to extract the local visual features efficiently, because it is a crucial aspect for efficient detection. Up to now, the key point extraction has only been performed on in-class and non-class images for building the representations used in the training phase of the SVMs. This allowed the grid of key points from the principal view to be applied directly to the in-class and non-class images, because they are of the same size. Then, for each key point a HOG descriptor is computed. The naive application of this scheme in the sliding window approach can be inefficient. The naive way to extract the key points for a current

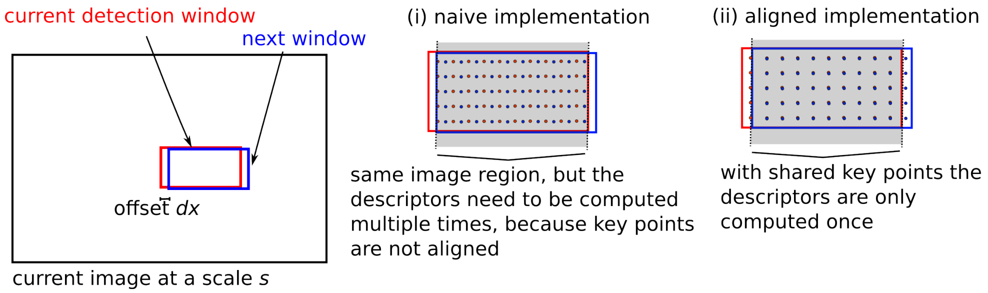

current detection window
next window
offset dx
current image at a scale s
(i) naive implementation
same image region, but the descriptors need to be computed multiple times, because key points are not aligned
(ii) aligned implementation
with shared key points the descriptors are only computed once

Figure 6.1: This figure shows two ways for implementing the feature extraction: (i) a naive solution, that causes multiple re-computations of very similar descriptors and (ii) an aligned version, where the descriptors need to be computed only once.

window is to compute the key point locations in the current window and then compute the local descriptors at each key point location. The key point locations in the current window can simply be calculated by using the current window origin as an offset for the key point grid defined on the principal view. Figure 6.1 shows the basic problem. The red window depicts the current detection window, and the blue box is the next detection window after shifting the red window by an offset dx. Usually, dx is quite small and both boxes do have a large overlapping region. However, in the naive approach the descriptors in that region need to be computed multiple times, because the key points are not aligned and have distinct locations. A better way to do this is shown in figure 6.1 (ii). Here, the key points are aligned, i.e. the offset in the x direction is chosen, in such a way that the key points after shifting lie above the old key point positions. Thus, all descriptors in the grey region need to be computed only once. Of course, the same principal can be applied in the y-direction to the offset dy. Taking both directions and the size of the view-tuned kernel into account, all key point locations can be specified on a grid for the entire image and be computed only once. Hence, no feature in any image region is computed more than once.

On the implementation side, the aligned key points are realised as follows. As stated before, the view-tuned kernel uses 900 key points for a view tuned kernel of size 128×128. That means a grid of 30×30 key points is employed and the key point have a grid distance of $\lfloor \frac{30}{128} \rfloor = 4px$ pixels in x- and y-direction. The offset parameters dx and dy for the sliding window search are not specified directly, but they are determined by two other parameters xs and ys, which state by what percentage of the width bw and height bh of the view-tuned kernel the box is moved, i.e.:

$$dx = xs \cdot bw \qquad xs \in (0, 1] \tag{6.7}$$

$$dy = ys \cdot bh \qquad ys \in (0, 1]. \tag{6.8}$$

The concrete offsets are then transformed such that that the key points are aligned. That means they are set to the nearest integer that is divisible by 4 without a remainder, i.e.

$$dx \longleftarrow \operatorname*{argmin}_{i \in \mathbb{N}^+, i \bmod 4 = 0} |dx - i| \tag{6.9}$$

$$dy \longleftarrow \operatorname*{argmin}_{i \in \mathbb{N}^+, i \bmod 4 = 0} |dy - i|. \tag{6.10}$$

In particular for small values of xs and ys, this alignment drastically reduces the complexity. For example, with $xs = .1$ and $ys = .1$ the aligned implementation is up to 10× faster. The presented feature extraction scheme is used implicitly from now on in this thesis. With an efficient feature extraction, another issue is how the decision function can be computed efficiently. This is discussed in the next section.

6.3 Efficient Computation of the Decision Function

Given a trained view-tuned SVM model, it can be used in a sliding window framework as a detector. In doing so, the runtime of the detector is dependent on the number of support vectors in the model. This means that each patch in the image must be compared with each of the support vectors and the kernel distance must be computed. Thus the time complexity for classifying a single pattern is $O(MD)$ where M is the number of support vectors and D is the dimensionality of the representation. Here, this calculation is made more efficient by using the results of [129] as a basis. The work of [129] shows, that (i) a histogram intersection kernel can be exactly computed in logarithmic time (as opposed to linear) and (ii) that an approximation can be computed in constant time (with negligible loss in accuracy). I extend this result to view-tuned kernels, and show that – due to the special design of view-tuned kernels – it is possible to compute the exact solution of the decision function in constant time. This means that the complexity of the decision function becomes *independent* from the number of support vectors and only depends on the dimensionality of the representation, i.e. $O(D)$.

First, some of the notation of the variables is introduced. Let M denote the number of support vectors, let R denote the number of levels of the 2D IbTreeGNG and let b denote the branching factor of the 2D IbTreeGNG. Further, let L denote the number of levels in the d-dimensional IbTreeGNG and let k denote the branching factor of that dD codebook tree. The used view-tuned kernel for a specific class and view ω is written as $k(x, y) \equiv \mathcal{T}(\Xi^{\omega}(X), \Xi^{\omega}(Y))$. Moreover, for a node q in the level p of the 2D IbTreeGNG the count according to a node v in the level u of the dD IbTreeGNG is denoted as $x_{pquv} \equiv n_{uv}(X^{\omega}_{pq})$. Also, let $x^1, \ldots, x^M$ denote the support vectors. Using the support vectors, the decision function can be written as follows:

$$f(x) \quad = \sum_{i=1}^{M} \alpha_i y_i k(x, x^i) + \beta. \tag{6.11}$$

Note that only the support vectors are used in this equation. This represents a slight difference from the notation used before, where all training examples were present and the non support vectors had a coefficient $\alpha_i = 0$. Plugging in the view-tuned kernel the function is written as:

$$f(x) \quad = \sum_{i=1}^{M} \alpha_i y_i \left[\sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \tilde{w}_{pq} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} w_{uv} min(x_{pquv}, x^i_{pquv}) \right] + \beta. \tag{6.12}$$

The first two sums in the squared brackets iterate over the indexes of the hierarchical codebook nodes in 2D and the second two sums iterate over the nodes of the hierarchical codebook in the d-dimensional feature space. By moving the coefficients inside, the equation can be written as follows:

$$f(x) = \sum_{i=1}^{M} \sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} \alpha_i y_i \tilde{w}_{pq} w_{uv} min(x_{pquv}, x^i_{pquv}) + \beta. \quad (6.13)$$

Now, the outer sum with the iterator i can be pulled inside and the decision function is written:

$$f(x) = \sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} \sum_{i=1}^{M} \alpha_i y_i \tilde{w}_{pq} w_{uv} min(x_{pquv}, x^i_{pquv}) + \beta \quad (6.14)$$

The inner sum iterates over the support vectors indexes i and provides a weighted minimum intersection for a specific bin, between the input vector and the i-th support vector. This sum can be expressed as a function of the bin indexes and the values of the input vector at that bin index, i.e.:

$$f(x) = \sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} h^{pquv}(x_{pquv}) + \beta. \quad (6.15)$$

The counts for the bins are integers, i.e. $s \in \mathbb{N}^0$, and the $h^{\cdot}(s)$-function in is given thus:

$$h^{pquv}(s) \;=\; \sum_{i=1}^{M} \alpha_i y_i \tilde{w}_{pq} w_{uv} min(s, x^i_{pquv}). \quad (6.16)$$

Obviously, the value of the $h^{\cdot}(s)$-function for a specific s is independent from a concrete input vector of the kernel and can be pre-computed. For view-tuned kernels, the maximal count for a specific bin x_{pquv} is bounded by the total number of key points G and bounded even much tighter by the number of key points G_{pq} that fall into the 2D bin pq, i.e.:

$$\text{(i)} \quad 0 \leq x_{pquv} \leq G \quad (6.17)$$

$$\text{(ii)} \quad 0 \leq x_{pquv} \leq G_{pq} \quad (6.18)$$

This fact makes it possible to pre-compute tables of a feasible size for a trained model once, and to store the exact solutions for a specific bin.

Technically, the cached $h^{\cdot}(.)$-functions are implemented to work on models that have been learned via `LIBSVM` [71] [27]. The worst-case space complexity of the cache depends on the size of the 2D and dD codebook trees. The total number of bins is obtained by multiplying the number of 2D nodes by the number of dD nodes, i.e.:

$O(R \cdot \sum_{i=1}^{R} b^i \cdot L \cdot \sum_{i=1}^{L} k^i)$. The total number of bins of the representation is quite high, e.g. a few hundred thousand dimensions. However, note that these representations are very sparse and actually need much less storage than this upper bound. The representations are implemented via sparse vectors of `OpenCV` [216] and store only the count for non-zero bins. Beyond that, when computing the decision function, a sparse iterator is used, such that the $h^{\cdot}(s)$-function lookup is performed only for non-zero bin counts ($s > 0$). The speed-up from using the cached $h^{\cdot}(.)$-function for computing the decision function is directly proportional to the number of support vectors. In the prototypical implementation of this thesis, the speeded up version was up to 20× faster. By using the scheme presented here, the efficiency of classification with view-tuned kernels is significantly improved. Moreover, the results render view-tuned kernel SVMs suitable for efficient application of very large models, since the runtime is independent of the number of supports patterns in the model.

6.4 View-tuned SVM Cascade

For an object detector in the sliding window approach, it can often be observed that the vast majority of tested windows do not contain a class instance. As a result, in the case of an view-tuned SVM, all of the windows are evaluated with a full kernel in order to classify them. The computation of the decision function has already been improved to a large extent (see section 6.3), however, it is still not free of cost. In order to minimise the residual amount of processing, I introduce cascaded classification for view-tuned SVMs in this chapter. Classifier cascades are a widely used technique (e.g. [215] [195]) for boosting the performance of detection systems. The basic idea is to define a sequence of classifiers $(C^1, \ldots, C^W)$, starting with simple classifiers that can be evaluated efficiently. On the other end of the sequence there

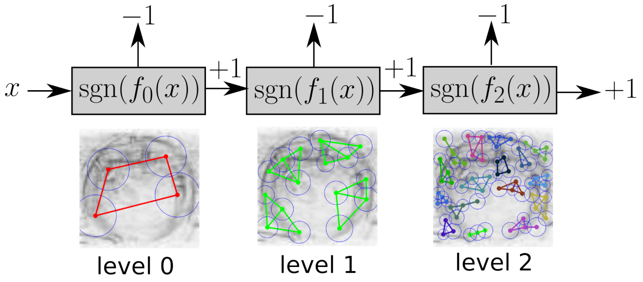


Figure 6.2: This figure sketches the basic idea of the cascaded view-tuned SVM. The input vector gets passed to the next classifier only if it is classified as in-class. The decision function $f_0(x)$ only uses the level 0 of the associated 2D codebook. The decision function $f_1(x)$ uses the result from level 0 and uses the level 1 of the codebook. The decision function $f_2(x)$ uses both results of the previous levels and additionally employs the level 2 of the hierarchical codebook.

are more precise, but also more complex classifiers that need more time for evaluation. Given an input x, the simple classifier $C^1(x)$ is used first. If the input is classified as in-class, i.e. $sgn(C^1(x)) = +1$, the input gets passed to the next classifier in the sequence, in this case C^2. If the input is classified as non-class, i.e. $sgn(C^1(x)) = -1$, this is taken as a final result and the input is not passed to any further classifier. The complex classification operations are computed only for "promising" samples, which have been classified by simpler models as in-class. A sample is only classified as in-class if $sgn(C^1(x)) = \ldots = sgn(C^W(x)) = 1$. If a sample is kicked out of the classification pipeline early, a large number of computations could be omitted for that sample. This is particularly useful in a detection scenario with a sliding window, because the vast majority of image patches do not even look close to an in-class sample. To that end, large parts of the image could only be classified with a cheap classifier and the evaluation of more complex classifiers is performed only for a few interesting image regions.

The basic idea for cascaded view-tuned SVMs is to the evaluate decision function piecewise by computing a decision function for each level. An input is passed to the next level only if it is classified as in-class. This scheme is sketched in figure 6.2. The cascade implements a rough to fine matching, and only promising image regions are processed further. In the example in figure 6.2, the first level has 4 bins, the second level has 16 bins and the third level has 61 bins. Obviously the cost of constructing a decision value from 61 nodes is much higher than for 4 nodes. For a trained cascade, the goal is to let all background image patches be kicked out of the pipeline as early as possible, while letting all in-class image patches pass through the entire pipeline.

Repeating the result of the last chapter, the cached version of the decision function has been written as:

$$f(x) = \sum_{p=0}^{R-1} \sum_{q=1}^{b^{p+1}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} h^{pquv}(x_{pquv}) + \beta. \tag{6.19}$$

In this equation, the first sum iterates over levels of the 2D codebook tree. The goal is to compute each level separately, and to reuse the already computed sums. Therefore, the sum over the p variables is is decomposed into R single $\underline{f}$-functions as follows.

$$\begin{aligned}
\underline{f}_0(x) &= \sum_{q=1}^{b} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} h^{0quv}(x_{0quv}) && (6.20)\\
&\dots \\
\underline{f}_i(x) &= \sum_{q=1}^{b^{i+1}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} h^{iquv}(x_{iquv}) + \underline{f}_{i-1}(x) && (6.21)\\
&\dots \\
\underline{f}_{R-1}(x) &= \sum_{q=1}^{b^{R}} \sum_{u=0}^{L-1} \sum_{v=1}^{k^{u+1}} h^{R-1,quv}(x_{R-1,quv}) + \underline{f}_{R-2}(x) && (6.22)
\end{aligned}$$

The cascade can now be defined by R classifiers, one for each level. In this process the $\underline{f}$-functions are employed as follows:

$$\begin{aligned}
f_0(x) &= \underline{f}_0(x) + \beta_0 && (6.23)\\
&\dots \\
f_i(x) &= \underline{f}_i(x) + \beta_i && (6.24)\\
&\dots \\
f_{R-1}(x) &= \underline{f}_{R-1}(x) + \beta_{R-1}. && (6.25)
\end{aligned}$$

Obviously, the original decision function $f(x)$ for the overall view-tuned SVM is given by the equation (6.25), i.e. $f(x) = \sum_{p=0}^{R-1} \underline{f}_p(x) + \beta_{R-1}$.

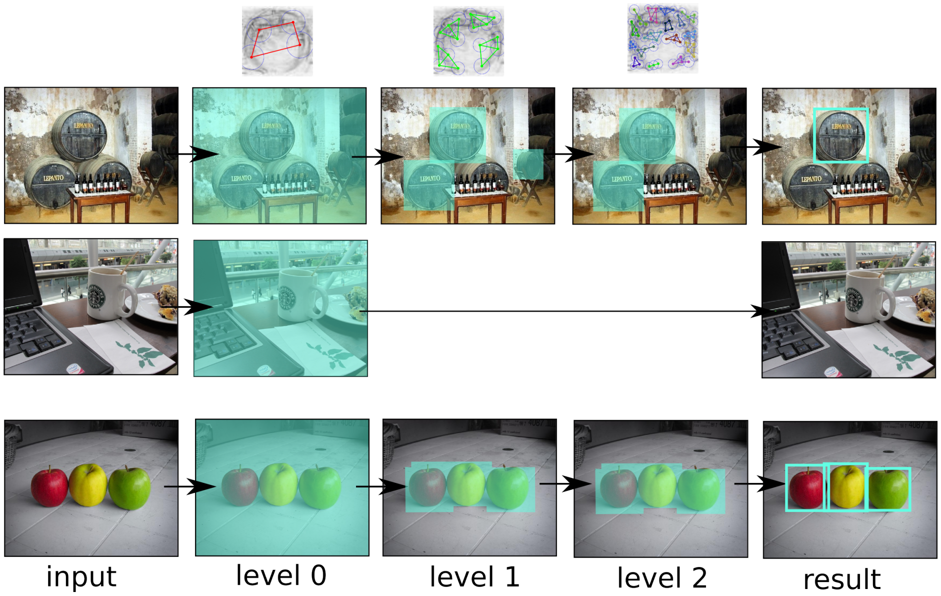


Figure 6.3: This figure shows the image regions (turquoise overlay) that are processed by the different classifier in view-tuned SVM cascade for the category apple on three input images.

Further, it is visible that the learning problem has been decomposed into R learning problems, which are themselves trainable in the view-tuned SVM framework of the last sections.

Although I have not yet explained the sequential discriminative optimisation, I want to show some exemplary results of a cascaded SVM that was trained with this technique, because it fits this section best. Figure 6.3 shows a view-tuned SVM cascade in as a sliding window detector for apples. The turquoise overlay shows what image regions are processed by the classifiers corresponding to the different levels of the cascade. In level 0 the entire image is processed. However, in the subsequent levels, the number of processed image regions could largely be reduced, because only patches that look somehow similar to the contour of an apple are passed through the cascade. Practically, because the decision function is already computed very efficiently, for the prototypical implementation "only" a speed-up factor $3-6\times$ has been observed. One question that is still open, is how to train the cascaded view-tuned

SVMs with “good” negative examples. This topic is discussed in the next section.

6.5 Sequential Discriminative Optimisation

As stated before, learning a view-tuned SVM classifier for object detection in a sliding window approach is an unbalanced learning problem. Given an image database, there is a very large number of image patches that can in principle be employed as negative examples for the training of a classifier. On the other hand, there is only a relatively small and fixed number of positive examples that is available for the training. As has been shown by other authors (e.g. [54] [195]) for training a good classifier it is important to use “good negative examples”. Good negatives are often considered to be negative samples that are hard to classify, i.e. the samples are close to the decision boundary. Since it is not initially clear what negative samples from the large pool of possible image patches are good negative samples, bootstrapping approaches have been proposed (e.g. [57]). Bootstrapping in this case works as follows: Initially a model is trained using the positive samples and a random set of negative images. Then this model is applied as to a dataset of negative images, and samples that are falsely classified as in-class are collected. After that, a new classifier is trained using the positive samples and the newly collected negative examples. This sequence of training a classifier and collecting negative samples is usually repeated several times. The practice is also called data-mining hard negative examples and is related to methods for working set selection (e.g. [98] [47]).

To train a level of a view-tuned SVM cascade in this thesis, bootstrapping is applied as well. Algorithmically, the bootstrapping approach is formalised as follows. Let S denote the overall set of negative samples and let $S_1 \cup \ldots \cup S_{T-1} = S$ denote a decomposition into $T-1$ chunks of data. One chunk of data S_i is used for the i-th training of the classifier. Further, let P denote the fixed set of positive samples and let N_i denote the set of samples, that is harvested in the iteration i. The algorithm for sequential discriminative optimisation is stated in

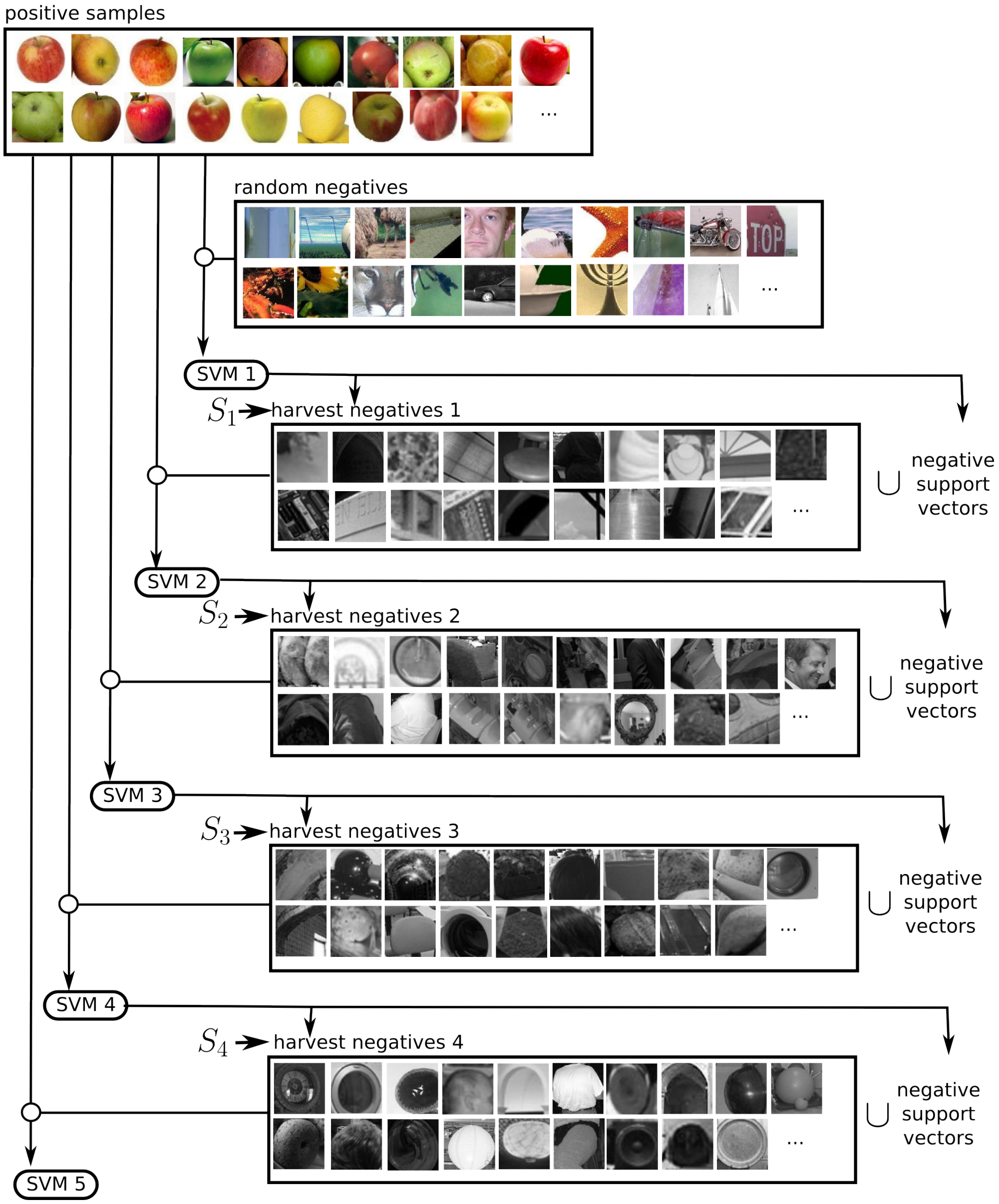


Figure 6.4: This figure shows an exemplary run of sequential discriminative optimisation (SDO) algorithm for the first cascade level of an view-tuned SVM and the category apple.

Algorithm 1 SDO Algorithm

```
Require: $T > 0$;  $P$;  $S = S_1 \cup \ldots \cup S_T$
1: init $N_0$ with random negative samples from $S$
2: for $i = 1$ to $T$ do
3:    $\texttt{model}_i \longleftarrow \texttt{train}(P, N_{i-1})$
4:    if $i < T$ then
5:       $N_i \longleftarrow \texttt{harvest}(\texttt{model}_i, S_i)$
6:       $N_i \longleftarrow \texttt{negative\_support\_vectors}(\texttt{model}_i) \cup N_i$
7:    end if
8: end for
```

pseudo-code in algorithm (1).

The harvesting process uses a current model $\texttt{model}_i$ on a chunk S_i and collects samples that are falsely classified, i.e. $\{x \in S_i : sgn(f^{\texttt{model}_i}(x)) = +1\}$. The function $\texttt{negative\_support\_vectors}(.)$ returns the set of support vectors for a given model. In line 3 of algorithm (1), the training is performed as stated in section 6.1, i.e. including the 10-fold cross-validation-based optimisation of the parameter C under the balanced true positive ratio. Because the set of positive examples is fixed and because the support vectors are kept in the working set, a solution can only improve if new training negative samples end up as support vectors.

Since the initial models are trained with random negative samples, they are usually quite weak and classify a lot of samples of S_i as in-class. Therefore, it is reasonable to choose the chucks to be of increasing size, i.e. $|S_1| < \ldots < |S_{T-1}|$, and to apply a well trained model to the largest amount of data, where $|.|$ denotes the cardinality of the set. In this thesis, I use $10,000$ negative images from the indoor scene database [168] for sampling the negative image patches. Another $5,000$ images from the indoor scene database are used later on for the evaluation of the detector. The indoor scene database is employed because it provides training and testing samples from the domain of application, i.e. images of domestic environments like kitchens, offices, garages etc. During learning, images are excluded from the database if they obviously contain target objects, e.g. the indoor scenes of a shoe shop are excluded from the dataset when harvesting negative images for the

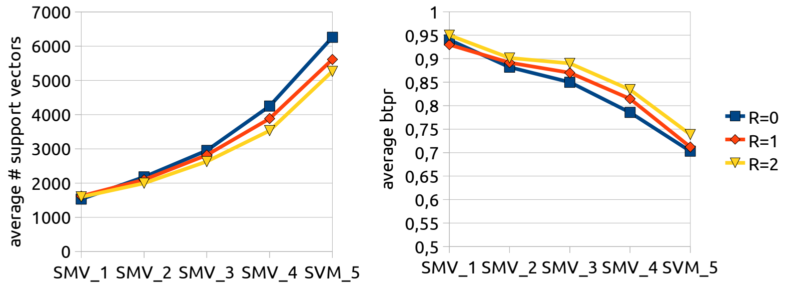


Figure 6.5: These plots show the average number of support vectors for the different SVMs of the sequential discriminative optimisation after training, and also the average balanced true positive rate (btpr) of SVMs after optimising C via cross-validation. The averages are calculated over the categories and over six extracted principal views.

shoe model. Beyond that, the database is quite large and contains a rich pool of relevant patterns. The images have been scaled in order to have approx. $640 \times 480 = 307200$ pixels. Empirically, a sequence length of $T = 5$ worked out quite well. Besides, it has been found to work well to double chunk size each time, because a simple model produces a lot of false positives and does not need much data, while a model that is already quite good needs much more data to find a sufficient number of negative samples. Formally, this is captured as

$$|S_2| = 2\cdot|S_1|, \quad |S_3| = 2\cdot|S_2| = 4\cdot|S_1|, \quad , |S_4| = 2\cdot|S_3| = 8\cdot|S_1|, \tag{6.26}$$

where the initial size $S_1 = \lfloor \frac{10{,}000}{15} \rfloor = 666$. Figure 6.4 sketches an exemplary run of the sequential discriminative optimisation for the first cascade level of a view-tuned SVM and the category apple. As can be seen, from a visual point of view, the harvested negative images make sense, because the sets of negative images in later iterations contain various patterns that are round in shape. All of those hard negative were harvested from the indoor scene database. Beyond that, figure 6.5 shows the average number of support vectors and the average balanced true positive ratio for the different SVMs in sequential discriminative optimisation. The values are averaged over six extracted principal views and over the categories. The number of support vectors can be observed to increase for SVMs that occur later in the sequence. This can be explained by the fact that the SVMs need to enhance the model complexity, as novel hard negative images are found in the training set. Further, it is clear that the average balanced true positive ratio (from the optimal C parameter of the cross validation) decreases for later SVMs in the pipeline. This may also be explained by the presence of the very hard negatives in the training set: the positive and the negative samples just become more difficult to separate and a trade-off between precision and recall is optimised. However, although the btpr of a single view-tuned SVM decreases when learned via SDO, the ensembling of different experts increases both the precision and the recall of the system since results from diverse experts are coupled. In a prototypical implementation, the training time for a complete run of the sequential discriminative optimisation for a single view-tuned SVM takes approx. 5-10 hours on a modern multi-core CPU like the Intel Core i7. An interesting opportunity for future work would be the integration of the harvesting process into the working set selection approach of [47], as it could improve optimization and the time needed for the harvesting procedure.

After considering the efficiency of the classifier, sequential discriminative optimisation is the last element in the training of view-tuned Support Vector Machines. The next chapter finally applies the view-tuned SVM for detection and presents a method for ensembling the results of several view-tuned SVMs.

6.6 Summary

As seen in this chapter, using SVMs with non-standard kernels and for the application in the sliding-window scheme is not a plug-and-play process. The essential questions are: How to do the processing efficiently? And, how to train good classifiers? In this chapter I have proposed a methodology for addressing both objectives. I have proposed an efficient approach for feature extraction, which avoids multiple computations of local descriptors by aligning the key points. Further, I have presented a caching scheme that makes the evaluation of a kernel independent from the number of support vectors and allows very efficient computation of the decision function of a view-tuned kernel. Beyond that, I have shown how to optimize a unbalanced problems in a C-SVM formulation with view-tuned kernels. Finally, I have presented a method for sequential discriminative optimization, which allows strong classifiers to be trained successively. The next chapter provides the methods for ensembling the results of different view-tuned SVMs and evaluates the overall system performance.

Chapter 7

Detection with Ensembles of View-tuned SVM Cascades

This chapter applies view-tuned SVMs for detecting domestic objects in images. In doing so, it is an important aspect, how to fuse the results of different view-tuned SVMs into a final detection result. In section 7.1, the concept of classifier combination is introduced. After that, section 7.2 formally introduces the problem of combining different bounding box detection results and discusses a set of different combination methods. Further, the section 7.3 introduces the methodology used for evaluation of the proposed detection system, and discusses results of set-ups with different parameter settings. Thereby, view-tuned models with a different number of principal views are compared, and also different ensemble methods. As a baseline, the results are compared with the performance of a state-of-the-art detection system trained on the same data.

7.1 Introduction

First of all, the output of the latest chapters is recapped. Initially, there is a data acquisition and annotation step, where category-related images are downloaded from the Web and class instances are annotated with bounding boxes in the images. This results in a pool of annotated training images for each of the selected domestic object categories. After that, the quite complex and large training sets are decomposed into several subsets, by employing a small number of extracted principal views. This associates the training images with principal views, which results in the association of one training set with each principal view. Subsequently, each principal view and its associated training set is used to train a view-tuned matching kernel, that structurally adapts to the training samples in the set and allows a tuned rough to fine matching of the class instances. These specific kernels are then used to train cascades of view-tuned SVMs via SDO. So, for a specific category, there is a set of view-tuned SVM cascade classifiers, which are employable for detection in a sliding window framework. However, a crucial question is how to combine the results of the cascades into a coherent detection result for an image.

Basically, after providing an input to a set of classifiers, an ensemble aggregates the different outputs of the classifiers into a final result. First of all, let us shortly look at the motivation for doing such a combination of classifiers. For a comprehensive view on classifier ensembles see e.g. [108]. There is a shared belief that, that for matters of great importance in our life, the aggregation of answers from different experts should allow a better and more informed decision. According to [108], there are three types of reason for combining single classifiers. The first reason is statistical. Assume there is a set of good classifiers with different generalisation performance on a data set. If a single classifier should be picked, there is the risk of picking a bad one. “Averaging” the results of all classifiers might not provide a better performance, but mitigates or eliminates the risk of picking an inadequate classifier. The second reason is computational. If a training algorithm uses approximative or heuristic components, the optimisation might lead to different local optima. When results are aggregated the combined classifier can end up closer to the optimal classifier. The third reason is representational. The optimal classifier for a problem may lie out of the function space that can be represented by a classifier. A combination of classifiers can allow learning to occur in a more suitable space. For example, a combination of linear classifiers can produce a non-linear decision boundary. Here, it is considered to be easier to learn a set of simpler classifiers, rather than to learn a single classifier of very high complexity directly. Another reason for classifier combination is stated in [165]. In the case of large datasets, the training of a single classifier with a large amount of data is commonly not feasible or scalable. A decomposition of the training set into smaller subsets of data, followed by a fusion of several classifiers follows the divide-and-conquer principle, that is usually more efficient and scalable. Beyond that, there is particular evidence in the field of object detection, showing that the combination of different detectors is beneficial for multi-view detection and provides better performance than single detectors (e.g. [199] [54]).

An important requirement to the single classifiers of an ensemble is that the classifiers make different errors. If the single classifiers produce the same errors, then a fusion of the results cannot reduce the total error. This notion is also called *diversity of the classifier outputs*. Therefore, the goal during ensemble design is to produce classifiers that are individual as possible and provide different decision boundaries [108] [165]. This principle of diversity is also applied in other places. For example, in safety-critical applications, different independent computers and implementations are used to compute a result and an action is performed only when all results agree. Since the individual software versions produce independent errors, the overall error can be largely reduced for the combination. In this thesis, the diversity, that is necessary to make the combination of individual classifiers work is ensured through two elements.

1. *Different Training Sets.* The class-associated image pools are decomposed into several different training sets, one for each principal view. By design, the training sets are intended to contain similar patterns and have only little overlay.

2. *Different Kernels.* For each subset, a view-tuned kernel is learned that defines a specific way to compare two samples. These kernels are used by the SVM classifiers. This adds another degree of diversity, since the different kernels compare training examples in their own ways.

Both aspects facilitate the diversity of the classifiers' outputs and creates different decision boundaries. The next section explains the ensembling of classification results for a detection framework in more detail.

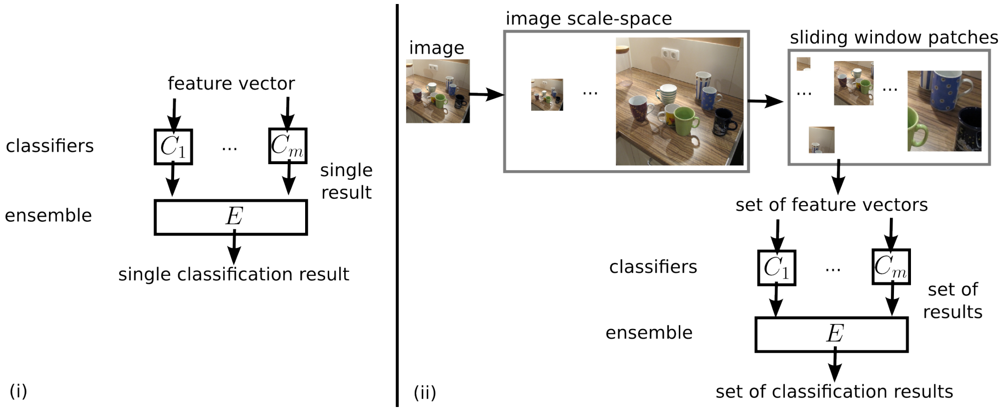


Figure 7.1: This figure sketches a (i) sample-based ensemble (ii) and a set-based ensemble for combining the classifiers in a sliding-window-based object detection approach.

7.2 Classifier Ensembles for Detection

Depending on the input and output specification of the classifiers and the combination, different types of ensembles could be distinguished. In the classical view, sketched in figure 7.1 (i), a single feature vector is passed to a set of classifiers $\{C_1, \ldots, C_m\}$, which all produce a single result. These results are the input of the ensemble E, which aggregates the results into a final classification result (cf. e.g. [108]). The outputs of the single classifiers are either considered to be class labels or continuous values. Therefore, methods for the *fusion of labels* and the *fusion of continuous values* are distinguished (cf. e.g. [165]). For two reasons, this scheme can not be applied directly for ensembling classification results in a sliding window detection framework. First of all, the scheme implicitly assumes that the samples are independent i.e., that knowing the predicted class of one image region does not provide any information for classifying another. This is obviously not true, for neighboring bounding boxes in an image that are overlapping. Knowing the predicted class of the one box could confirm or weaken the prediction of the neighboring window. The second reason is that, technically, there are multiple concurrent ensmbling problems: the fusion of the bounding boxes, the fusion of the class labels and the fusion of reliability scores.

This is because the final result of a detection ensemble has to be a set of pairs $\{(b_i, r_i)\}$, where b_i represents a bounding box, r_i represents a reliability score of the prediction. Here the bounding boxes that are elements of the result set are implicitly in-class, because windows that are classified with non-class are not to be contained in the final result set. The reliability score is mandatory, as it is needed for the evaluation of the detection system (cf. section 7.3). In the following, I present a set based ensembling scheme and a generic ensembling framework for dealing with the two stated aspects. This lays the ground for the coherent definition of several ensembles for detection results.

The first issue can be tackled, by considering a set-based ensembling problem w.r.t. the entire image, as outlined in 7.1 (ii). Here, a sliding window process on an image scale space defines a set of feature vectors (one for each window) and the feature sets are passed to the classifiers. Thus, the output of a classifier is a set of results. These sets are then combined by the ensemble in order to produce a final set of classification results. Using this scheme makes it possible to consider a merging of boxes, e.g. when one bounding box is contained in another box. In the classic scheme, this kind of information is not represented for the ensembling and can only be determined in post-processing (but some boxes may have already been rejected at that stage). Mathematically, the set-based formulation is introduced as follows. The decision functions of the view-tuned cascaded SVM from the last chapter are again denoted as $f_i(x)$. Here, an additional superscript j is introduced in $f_i^j(x)$, which represents the index of the corresponding principal view. For example, $f_1^0(x)$ is the decision function in level 0 of the cascade that belongs to the principal view number 1. Further, let Z denote the overall number of principal views (usually three, six or nine in this thesis). Further, let Γ be the multi-set of all features vectors, belonging to all windows at all scales, with a total size of $|\Gamma|$. When processing the set of all features vectors Γ with the cascades, the sets of results can be written for each view-tuned cascade as follows.

$$\begin{aligned} f_0^0(x), \dots, f_{R-1}^0(x), x \in \Gamma \longmapsto \quad & G^0 = \Big\{(b_i^0, q_i^0, r_i^0)\Big\}_{i=1,\dots,|\Gamma|} \\ & \vdots \\ f_0^{Z-1}(x), \dots, f_{R-1}^{Z-1}(x), x \in \Gamma \longmapsto \quad & G^{Z-1} = \Big\{(b_i^{Z-1}, q_i^{Z-1}, r_i^{Z-1})\Big\}_{i=1,\dots,|\Gamma|} \end{aligned} \tag{7.1}$$

Here, $b_i^j \in \mathbb{N}^4$ represents the bounding box coordinates, $q_i^j \in \{-1, +1\}$ represents the predicted class label and $r_i^j \in \mathbb{R}$ is a reliability value, given by the distance of the sample to the decision boundary. As an example, the set G^1 is the set of results when processing the set of all features Γ with the view-tuned SVM cascade of the principal view 1. Using the vector notation $\vec{q}_i = [q_i^0, \dots, q_i^{Z-1}]^T$ and $\vec{r}_i = [r_i^0, \dots, r_i^{Z-1}]^T$ these result sets can summarised across the different cascades, and be written as the joined set

$$G^{'} = \Big\{(b_i, \vec{q}_i, \vec{r}_i)\Big\}_{i=1\dots,|\Gamma|}. \tag{7.2}$$

This set does still contain all bounding boxes, even those in which all classifier cascades agreed that the corresponding image patch is non-class. The set $G^{'}$ is now condensed to a set that only contains bounding boxes that are predicted to be in-class by at least one classifier cascade, i.e.:

$$G = \left\{ s = (b_i, \vec{q}_i, \vec{r}_i) \quad : \quad s \in G^{'} \wedge \sum_{j=0}^{Z-1} q_i^j > -Z \right\}. \tag{7.3}$$

A classifier ensemble can now be written as a function E that transforms a set of predictions G into a final set $\mathcal{D}$ of pairs, i.e.:

$$E(G) \longmapsto \mathcal{D} = \big\{(b_1, r_1), \dots, (b_{|\mathcal{D}|}, r_{|\mathcal{D}|})\big\}. \tag{7.4}$$

Here, again $b_i \in \mathbb{N}^4$ represents the bounding box coordinates and $r_i \in \mathbb{R}$ represents a reliability score for the prediction. As was pointed out before, the reliability value is mandatory for each bounding box, because it is needed for the evaluation as part of the precision-recall curve computation. For Support Vector Machines, the reliability can be interpreted as being proportional to the distance $r_i^j \in \mathbb{R}$ of a sample to the decision

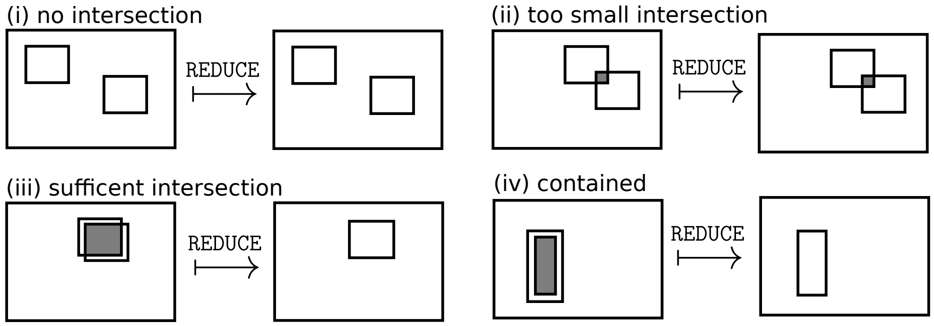


Figure 7.2: This figure sketches the results of the `REDUCE` method on pairs of bounding boxes. If two boxes are (i) not overlapping or (ii) have too little overlay, the tuple $(b_i, \vec{q}_i, \underline{\vec{r}}_i)$ remains untouched. Only if (iii) two boxes have a sufficient intersection or (iv) one box is contained within the other the respective tuples become merged.

hyperplane, given by $w^T w$. However, for different classifiers with different kernels the distance values may be in different ranges. Therefore it is better to consider a transformed distance value $\underline{r}_i^j \in [0, 1]$. The values $\underline{r}_i^j$ can then be interpreted as the "level of support" from the classifier j to the class q_i^j of the sample i. The new support scores can be computed with the softmax model [43], i.e.:

$$\underline{r}_i^j = \frac{\exp(r_i^j)}{\sum_{k=0}^{Z-1} \exp(r_i^k)}. \tag{7.5}$$

The vector of normalised reliability scores is written as $\underline{\vec{r}}_i = [\underline{r}_i^0, \dots, \underline{r}_i^{Z-1}]^T$.

The second issue is that there are multiple ensembling problems, when fusing a set G with an ensemble. Hence, the ensembling can be considered as a two step process:

1. *Reduction.* In the reduction step, overlapping bounding boxes are merged. At the same time, the corresponding class labels and the reliability scores need to be merged. Formally, the reduction can be written as a function that transforms the set of results G into another set of results G^r:

$$\texttt{REDUCE}(G; (B, Q, \underline{R})) \longmapsto G^r = \left\{(b_i, \vec{q}_i, \underline{\vec{r}}_i)\right\}_{i=1\dots,|G^r|}. \tag{7.6}$$

   The second parameter of the method is a tuple of functions. The function $B : \mathbb{N}^4 \times \mathbb{N}^4 \times \{-1, 1\}^Z \times \{-1, 1\}^Z \times [0, 1]^Z \times [0, 1]^Z \longrightarrow \mathbb{N}^4$ implements the merging of two bounding boxes b_1 and b_2 into a single box. The function $Q : \{-1, 1\}^Z \times \{-1, 1\}^Z \times [0, 1]^Z \times [0, 1]^Z \longrightarrow \{-1, 1\}^Z$ implements the merging of two class label vectors $\vec{q}_1$ and $\vec{q}_2$ into a single class label vector. Further, the function $\underline{R} : [0, 1]^Z \times [0, 1]^Z \times \{-1, 1\}^Z \times \{-1, 1\}^Z \longrightarrow [0, 1]^Z$ implements a merging of two reliability vectors $\underline{\vec{r}}_1$ and $\underline{\vec{r}}_2$ into one. The basic results of applying the `REDUCE` method to pairs of bounding boxes is sketched in figure 7.2. A triple $(b_i, \vec{q}_i, \underline{\vec{r}}_i)$ boxes is not altered, if the bounding box does not intersect any other box, or if the intersection is too small. Two boxes b_i and b_j (and their corresponding $\vec{q}_i$, $\vec{q}_j$ and $\underline{\vec{r}}_i$, $\underline{\vec{r}}_j$ vectors) are merged only if the boxes have a have a sufficient overlap or one box is contained in the other box. The pseudo code of the `REDUCE` method can be found in algorithm 2 in the appendix. Technically, the area of overlap threshold, which determines whether two boxes are considered as overlapping, is a parameter as well. However, in this thesis the threshold has been statically set as to the same value as in [46] (see algorithm 3 in the appendix).

2. *Fusion.* As the name suggests, this step fuses the elements of the reduced set G^r into a final set of detection results. In effect, the fusion operation acts like a filter, because weak hypothesis should be rejected by the ensemble and not be part of the final result set. The fusion can be formulated as follows:

$$\mathtt{FUSE}(G^r; F) \longmapsto G^f = \Big\{(b_i, \underline{r}_i)\Big\}_{i=1\ldots,|G^f|}. \tag{7.7}$$

   The second parameter of the method is a function $F : \mathbb{N}^4 \times \{-1, 1\}^Z \times [0, 1]^Z \longrightarrow \mathbb{N}^4 \times [0, 1]$, that fuses the class label vector and the reliability vector into a single reliability score. By convention, boxes with a fused reliability score equal to 0 are not added to the final result set G^f. The pseudo-code for the `FUSE` function can be found in algorithm 4 in the appendix.

By using the presented notation, a general ensemble for sets of detec-

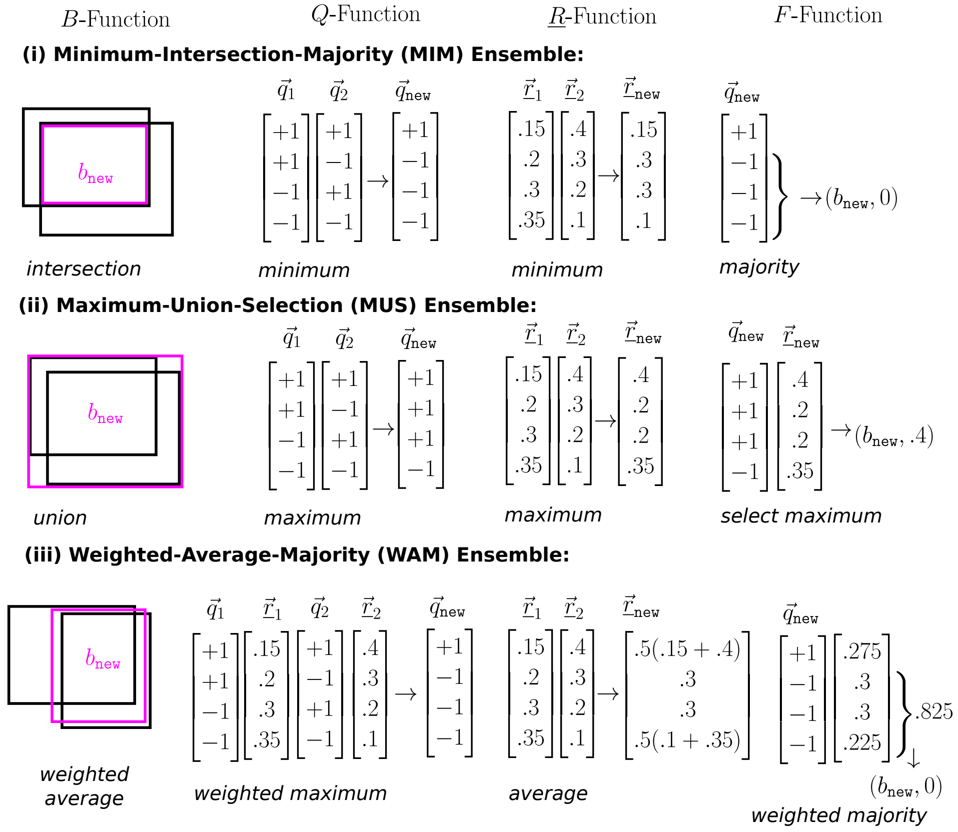

B-Function
Q-Function
R-Function
F-Function
(i) Minimum-Intersection-Majority (MIM) Ensemble:
b_{new}
$\vec{q}_1$ $\vec{q}_2$ $\vec{q}_{\text{new}}$
$\vec{r}_1$ $\vec{r}_2$ $\vec{r}_{\text{new}}$
$\to (b_{\text{new}}, 0)$
intersection
minimum
minimum
majority
(ii) Maximum-Union-Selection (MUS) Ensemble:
$\to (b_{\text{new}}, .4)$
union
maximum
maximum
select maximum
(iii) Weighted-Average-Majority (WAM) Ensemble:
$.5(.15+.4)$
$.5(.1+.35)$
$.825$
$(b_{\text{new}}, 0)$
weighted average
weighted maximum
average
weighted majority

Figure 7.3: This figure shows three different ensembles of detection results (details in the text).

tion results can be written as

$$E(G;(B,Q,\underline{R},F)) \longmapsto \mathcal{D} = \texttt{FUSE}(\texttt{REDUCE}(G;(B,Q,\underline{R}));F). \quad (7.8)$$

The benefit here is that different approaches for fusing the detection results can easily be defined by a tuple of functions $(B, Q, \underline{R}, F)$. Further, for implementing the single functions in the tuple, common ensembling approaches for fusion of labels and fusion of continuous values can be realised. In the following, I introduce three different ensembling methods, which are compared for their overall performance later on.

The Minimum-Intersection-Majority (MIM) Ensemble

The MIM-Ensemble is associated with the function tuple $(B_{\texttt{MIM}}, Q_{\texttt{MIM}}, \underline{R}_{\texttt{MIM}}, F_{\texttt{MIM}})$. These functions implement a "cautious fusion" and are seen in figure 7.3 (i). First, if the intersection of two bounding boxes is big enough, $B_{\texttt{MIM}}$ computes the new bounding box as the intersection of both boxes. That means that the function trusts only the image region that both boxes have in common. The $Q_{\texttt{MIM}}$ functions computes a new vector of labels component-wise via the minimum operation. This means that the agreed label is chosen only if both labels agree, and in case of any disagreement, the negative class is chosen. The behavior of an $\underline{R}$ function has to play hand in with the results of a Q function, to produce semantically correct results. Here, $\underline{R}_{\texttt{MIM}}$ is realised via a component-wise minimum, but the minimum is only taken for the components where the corresponding labels in $\vec{q}_1$ and $\vec{q}_2$ agree. Since $Q_{\texttt{MIM}}$ maps non-agreeing components to the minimum -1, the reliability score in the resulting component is set to the corresponding reliability value of the minimum (.3 and .3 in the example). Finally, the $F_{\texttt{MIM}}$ is realised with a majority vote. That means the box is classified as non-class (by setting the reliability to 0), if more than $50\% + 1$ of the predicted class labels in $\vec{q}_{\texttt{new}}$ are -1. If the majority of labels is positive, the function returns $(\vec{q}_{\texttt{new}}, \underline{r}_{\texttt{new}})$, where $\underline{r}_{\texttt{new}}$ is the minimum of the reliabilities in $\underline{\vec{r}}_{\texttt{new}}$ that belong to the positive components of $\vec{q}_{\texttt{new}}$. In the result, the MIM ensemble behaves rather pessimistic and tends to classify bounding boxes as in-class only if the ensemble shares a unified prediction.

The Maximum-Union-Selection (MUS) Ensemble

The MUS ensemble can be considered the opposite of the MIM ensemble. It implements an optimistic behaviour and classifies bounding boxes as in-class, even if they have little unified support. The ensemble is represented by the function tuple $(B_{\texttt{MUS}}, Q_{\texttt{MUS}}, \underline{R}_{\texttt{MUS}}, F_{\texttt{MUS}})$. The functions are seen in figure 7.3 (ii). The function $B_{\texttt{MUS}}$ merges two bounding boxes airly by using their union. The merging of two label vectors $\vec{q}_1$ and $\vec{q}_2$ is implemented with a component-wise maximum operation. That means, if both labels agree, the result component is set to that label. If the labels disagree, it is set to in-class. The $\underline{R}_{\texttt{MUS}}$ is implemented as a maximum function, if both labels agree. If the labels do not agree, the reliability score of the component is set to those of the respective positive label (.2 and .2 in the example). Finally, $F_{\mathtt{MUS}}$ is implemented as a classifier selection mechanism and selects the class with maximal reliability. So the result is set to $(b_{\mathtt{new}}, 0)$ if a negative class has the highest reliability value and to $(b_{\mathtt{new}}, \underline{r}_{\mathtt{new}})$ otherwise, where $\underline{r}_{\mathtt{new}}$ is the maximal positive reliability value. In this context, even if a positive class has only one vote, it can be picked if it has the highest reliability score.

The Weighted-Average-Majority (WAM) Ensemble

The WAM ensemble implements a behaviour between the MIM and the MUS ensemble. The combination is represented by means of a function tuple $(B_{\mathtt{WAM}}, Q_{\mathtt{WAM}}, \underline{R}_{\mathtt{WAM}}, F_{\mathtt{WA}M})$, and the functions are shown in 7.3 (iii). The $B_{\mathtt{WAM}}$ function is implemented as a weighted average, where the weight of each box is assigned the average of the reliabilities belonging to the positive labels. This kind of merging reduces the risk of moving parts of the box from reliable to unreliable image regions. Further, the function $Q_{\mathtt{WAM}}$ is realised as component-wise weighted maximum. In each component, the functions assigns the label, which has the highest corresponding reliability score. The function $\underline{R}_{\mathtt{WAM}}$ implements an averaging of the reliability values, if the two corresponding labels agree. If the two labels don't agree, the component is set to the maximum of the two reliability values (.3 and .3 in the example). Finally, the $F_{\mathtt{WAM}}$ function is computed as follows. If the sum of the reliability values belonging to the negative components is bigger than the sum of the reliability values belonging to the positive components, then the result is $(b_{\mathtt{new}}, 0)$. Otherwise, the result is $(b_{\mathtt{new}}, \underline{r}_{\mathtt{new}})$, where $\underline{r}_{\mathtt{new}}$ is the average of the reliability scores of the positive components. In the next section the three different detection ensembles are compared, as used with view-tuned SVMs in a sliding window approach.

7.3 Results & Discussion

This section evaluates the proposed system for the detection of domestic objects, i.e. ensembles of view-tuned SVM cascades. Ensembles with a different number of components are compared. Further, the performance of several combination methods is compared. A state-of-the-art detection system is employed to produce an experimental baseline.

Data

For the training and testing of the ensembles of view-tuned SVM cascades, the following positive and negative samples are employed.

- The *positive training samples* for a category are given by the annotated objects in the Web images. The data is processed into feature vectors and used for learning view-tuned SVM cascades as stated in the respective sections.
- The *negative training samples* are harvested in the sequential discriminative optimisation for a view-tuned SVM cascade from a set of $10,000$ images of the indoor scene database [168], as discussed before.
- The *positive testing samples* are also given from the annotation phase. A special `@HOME` tag was used to label all images, which show class instances in a domestic environment (like a toaster on a kitchen table). All `@HOME` images are employed as positive test images.
- As *negative testing samples*, a set of $5,000$ images from the indoor scene database is used [168]. The set is distinct from the image set used for training, and again the images are scaled to approx. $640 \times 480 = 307,200$ pixels. This image set is used to determine to what extent a detection system confuses background patterns with the patterns of objects.

Using the `@HOME` images as positive test samples and the indoor scene images as negative test samples is important, because both aims at the same thing: The applicability and generalisation ability of the proposed detection system should be assessed as realistic as possible. Therefore, the testing data should explicitly come from the domain of application, i.e. domestic home environments or more general, common indoor

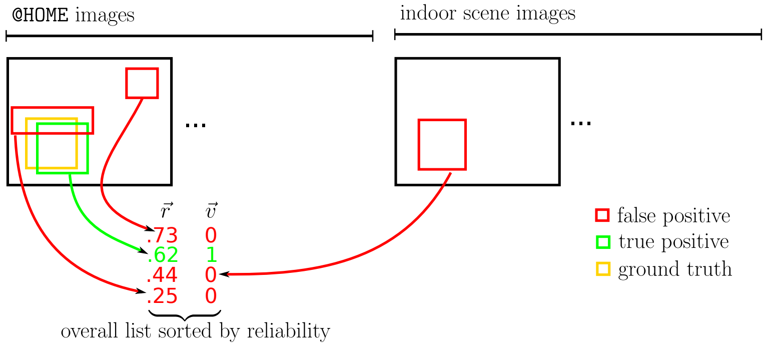


Figure 7.4: This figure presents a sketch of the evaluation scheme.

places. The indoor scene database is very large (67 categories; 15,620 images), and contains virtually every indoor scene category a domestic service robot may appear in, in the near future. Therefore, the indoor scene database provides a rich pool of patterns that can be used for training, but also for testing false positives in realistic environments.

Methodology

For the evaluation of a detection system, this thesis employs the same methodology as used in the evaluation of the well-established PASCAL detection challenge [46]. Each domestic object in the test images is annotated with a ground truth bounding box B_{gt}. Given a predicted bounding box B_p, the *area of overlap* a_o is computed as follows:

$$a_o = \frac{\texttt{area}(B_p \cap B_{gt})}{\texttt{area}(B_p \cup B_{gt})}. \tag{7.9}$$

If the area of overlap a_o is greater than .5, the predicted bounding box is counted as a true positive. Since each ground truth box can be counted only once as a true positive, in the case of multiple boxes overlapping with the ground truth box, only one of them is counted as a true positive and the other boxes are counted as false positives. For each predicted box, this results in a tuple (r_i, v_i), where $r_i \in \mathbb{R}$ is a reliability score belonging to the bounding box and $v_i \in \{0, 1\}$ is 1, if the box is counted as a true positive and 0 otherwise. The basic scheme is sketched in figure 7.4. The list of pairs is sorted according to the reliability values. From the sorted list of length $\mathcal{Y}$, a vector of sorted reliability scores $\vec{r}$ and a binary vector $\vec{v}$ of the corresponding v_i's can be defined. These sorted vectors are now used to compute the precision-recall curve as follows. First, lets recap the general definition of precision and recall. The precision score is defined as $prec = \frac{tp}{tp+fp}$ and the recall score is obtained through the formula $rec = \frac{tp}{tp+fn}$, where tp is the number of true positives, fp is the number of false positives and fn is the number of false negatives. Here, the precision and recall values for the i-th component of the vector is computed as:

$$prec_i = \frac{tp_i}{tp_i + fp_i} \tag{7.10}$$

$$rec_i = \frac{tp_i}{\mathcal{X}}. \tag{7.11}$$

Here, $i > 0$ and $tp_i = \sum_{j=0}^{i-1} v_j$ is the number of true positives until component i. Further, $fp_i = \sum_{j=0}^{i-1} \neg(v_j)$ is the number of false positives until component i, where $\neg(.)$ denotes the logical negation. Further $\mathcal{X}$ denotes the total number of ground truth bounding boxes. Obviously, if for a ground truth bounding box there is no predicted bounding box with $a_o > .5$, then this is counted as a false negative. So, the precision can be interpreted as a measure of exactness, where the recall can be seen as a measure of completeness. Then, the precision-recall curve is given by computing the precision-recall values for each predicted component, i.e. by the graph $\{(rec_i, prec_i) \ : \ i = 1, \ldots, \mathcal{Y}\}$. However, to get a principal quantitative measure of the performance, the *average precision* (AP) is then employed, as in the PASCAL detection challenge [46]. Given a precision-recall curve with monotonically decreasing precision, the AP is computed by numerical integration of the area under the curve. The AP value boils the shape of a single precision-recall curve down into a comparable quantity between 0 and 1. This value is employed when comparing two detection systems for a single category. For a set of categories, an averaged AP (APP) provides the average score over different categories and makes detection systems comparable globally.

	#principal views		
	three	six	nine
MIM ensemble	.381	.362	.110
MUS ensemble	.354	.303	.223
WAM ensemble	*.409*	**.477**	.134

Table 7.1: This table shows the overall system performance of cascaded view-tuned SVM ensembles – measured by the AAP score – for a different number of principal views and with different ensembling procedures. The best score is written in bold, the second best is written in italics.

Baseline

As mentioned in chapter 2, the Deformable Part Model (DPM) [54] has been among the best performing object detection systems at the PASCAL object detection challenge [44] [46]. Beyond that, is has also been successfully applied for object detection by the mobile robot Curious George [135], which has been among the top performing systems of the Semantic Robot Vision Challenge [197]. Further, the Deformable Part Model has a publically available implementation, and as [46] states, the DPM detector "*would form a reasonable state-of-the-art baseline for future challenges*". For all of this reasons, the DPM is employed as baseline detection system for the evaluation in this thesis. Specifically, the Deformable Part Model Release 4 [58] is employed with a total number of six components (there are three models, which are mirrored along the x-direction and which results in a total number of six components). Comparing the proposed object detection system of this thesis with DPM also provides a relation to various other detection systems, because the DPM has been compared with them. For a visualisation of the learned DPMs, please see figures B.2, B.3, B.4 and B.5 in the appendix. As visible in the figures – and also indicated by the AP scores later on – the DPM learns very reasonable detectors out of the box, when used with the large datasets described in this thesis. Note that the cascaded version of the DPMs were published in 2010 so, roughly stated, their development was carried at the same time as the development of the method in this thesis.

	apple	bottle	bowl	cup	handbag	laptop	light switch	potted plant	shoe	toaster	AAP
DPM	**.451**	**.589**	.513	.638	**.686**	**.561**	**.32**	.397	.274	**.391**	**.482**
WA_6	.346	.569	**.535**	**.709**	.657	.556	.235	**.451**	**.351**	.367	.478

Table 7.2: This table compares the six-component DPM model with the best-performing cascaded view-tuned SVM ensemble. The average precision (AP) and is shown for each category, as well as the averaged AP (APP) over all categories.

Evaluation

The evaluation compares ensembles of view-tuned SVM cascades with three, six and nine extracted principal views. In addition, for each of those view-tuned cascades, the three different ensembling strategies of section 7.2 are compared. The overall system performance is compared by means of the AAP score. This means that if the APP score of one detection system is higher than the score of another system, it implements a better detector in total. The APP results of the detectors can be found in figure 7.1. The best performing set-up is given by a model with six principal views and with the WAM ensemble. The second best score is achieved by the model with three principal views and the WAM-ensemble. It can be observed that the performance for systems with nine principal views are considerably lower for all ensembling methods, compared with the residual scores. One explanation for MIM_9 and WAM_9 is that the majority and weighted majority voting does not work properly for larger ensembles, because as it turns out, the majority of classifiers predict a non-class label most of the time. Only in a few cases do most experts seem to agree when predicting an in-class label. Further, for the MUS_9 model (which selects the class with the maximum reliability score) when more classifiers are present the maximum can more often be found for a classifier that predicts non-class. In contrast to that, the (weighted) majority vote does work, when the number of principal views is lower, i.e. equal to six or three. For three and six principal views, the WAM ensemble works best. An explanation for this is given by the following observation. In practice, the MIM ensemble reduces the recall of the system

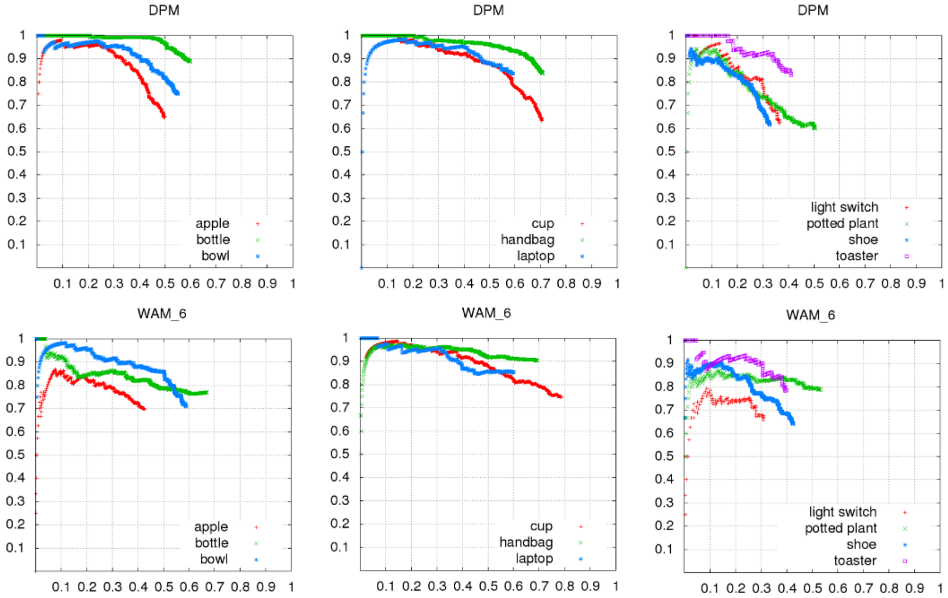


Figure 7.5: This figure shows the precision-recall curves of the DPM baseline and the best performing cascaded view-tuned SVM ensemble (WA_6) for each domestic object category.

by rejecting the boxes, which have inadequate support. This reduces the false-positive-rate (which is good), but at the same this process reduces the precision of the system. The MUS ensemble behaves the other way round in practice. It increases the recall of the system by classifying in-class more often than the WAM-ensemble. However, this also increases the false-positive rate of the system, what then reduces the precision. Thus, resulting from the experiments, the WAM ensemble shows the best trade-off between precision and recall for cascaded view-tuned SVM ensembles. Note, that although the table looks quite "inconspicuous", it took a long time to be computed, since over $3 \cdot 3 \cdot 3 \cdot 5 \cdot 10 + 6 \cdot 3 \cdot 3 \cdot 5 \cdot 10 + 9 \cdot 3 \cdot 3 \cdot 5 \cdot 10 = 8100$ classifiers need to be optimised via 10-fold cross-validation. Above, a single summand is given by $\#views \cdot \#ensembles \cdot \#cascades \cdot \#SDO\_seequences \cdot \#classes$.

Table 7.2 shows the AP score of each category for the optimal cascaded view-tuned SVM ensemble (WAM_6). Also the scores of the DPM baseline models are stated. As can be seen, the overall system

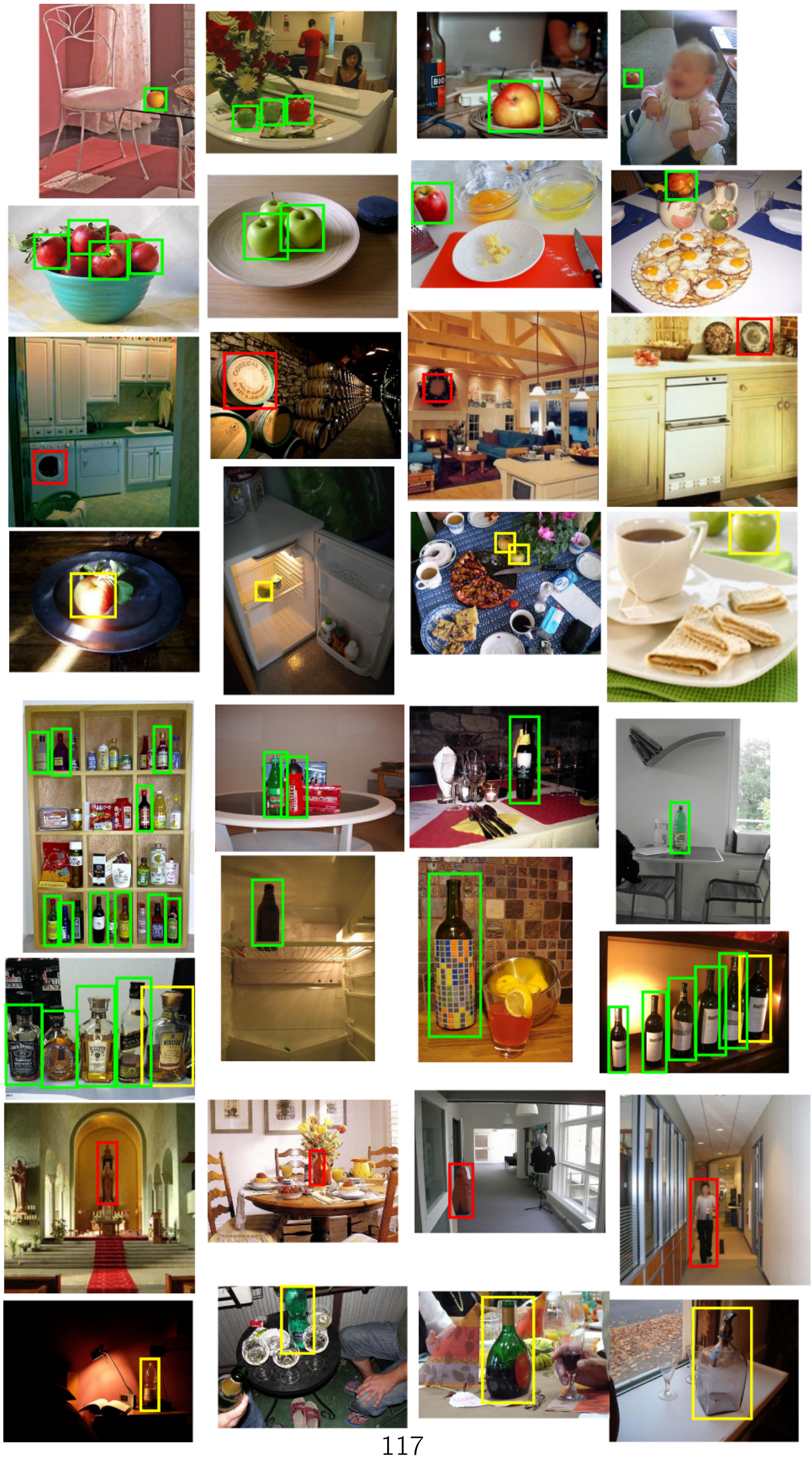


Figure 7.6: Example results of `apple` and `bottle` detections (green≙true positive; red≙false positive; yellow≙false negative).

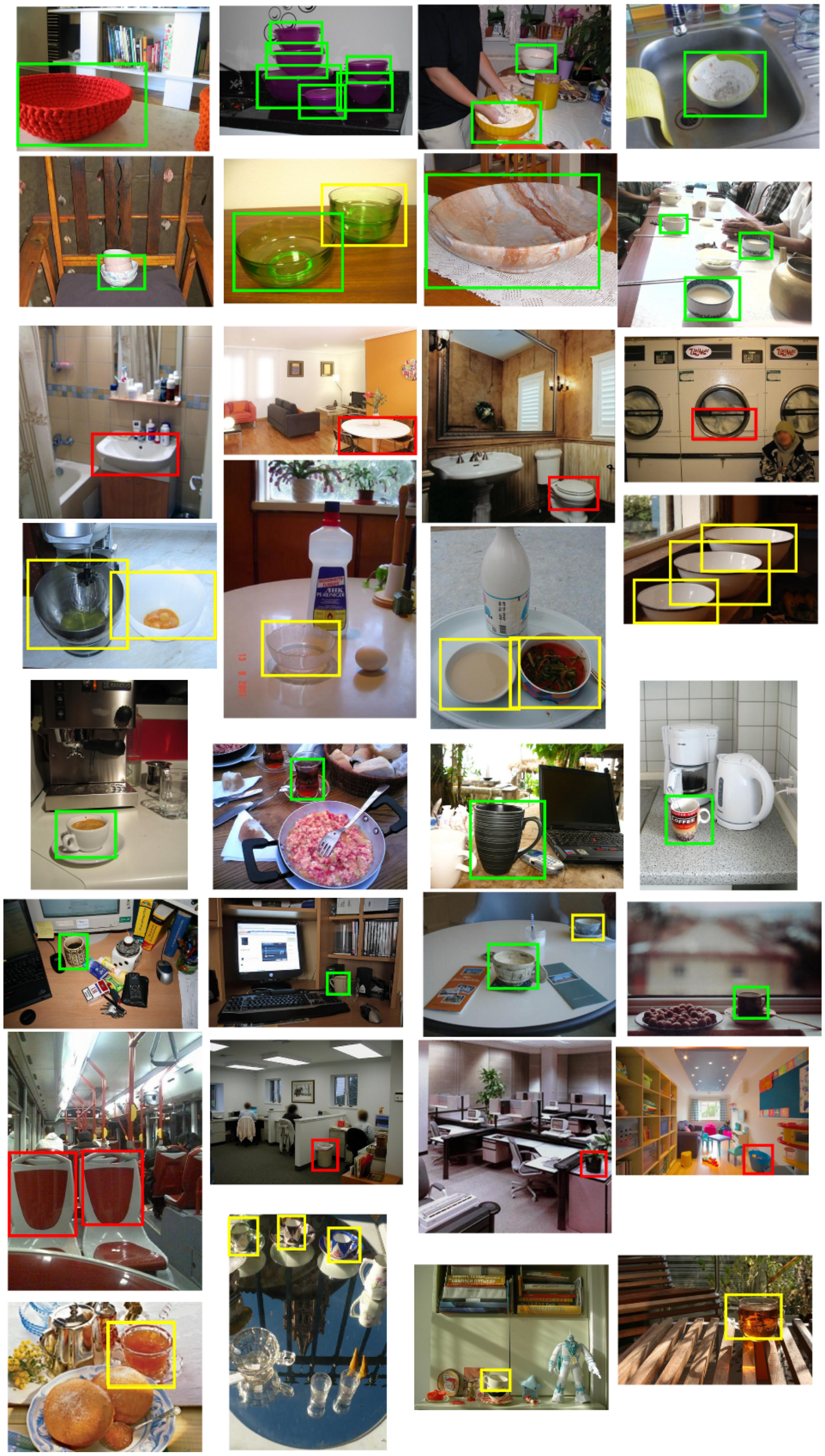


Figure 7.7: Example results of `bowl` and `cup` detections (green≙true positive; red≙false positive; yellow≙false negative).

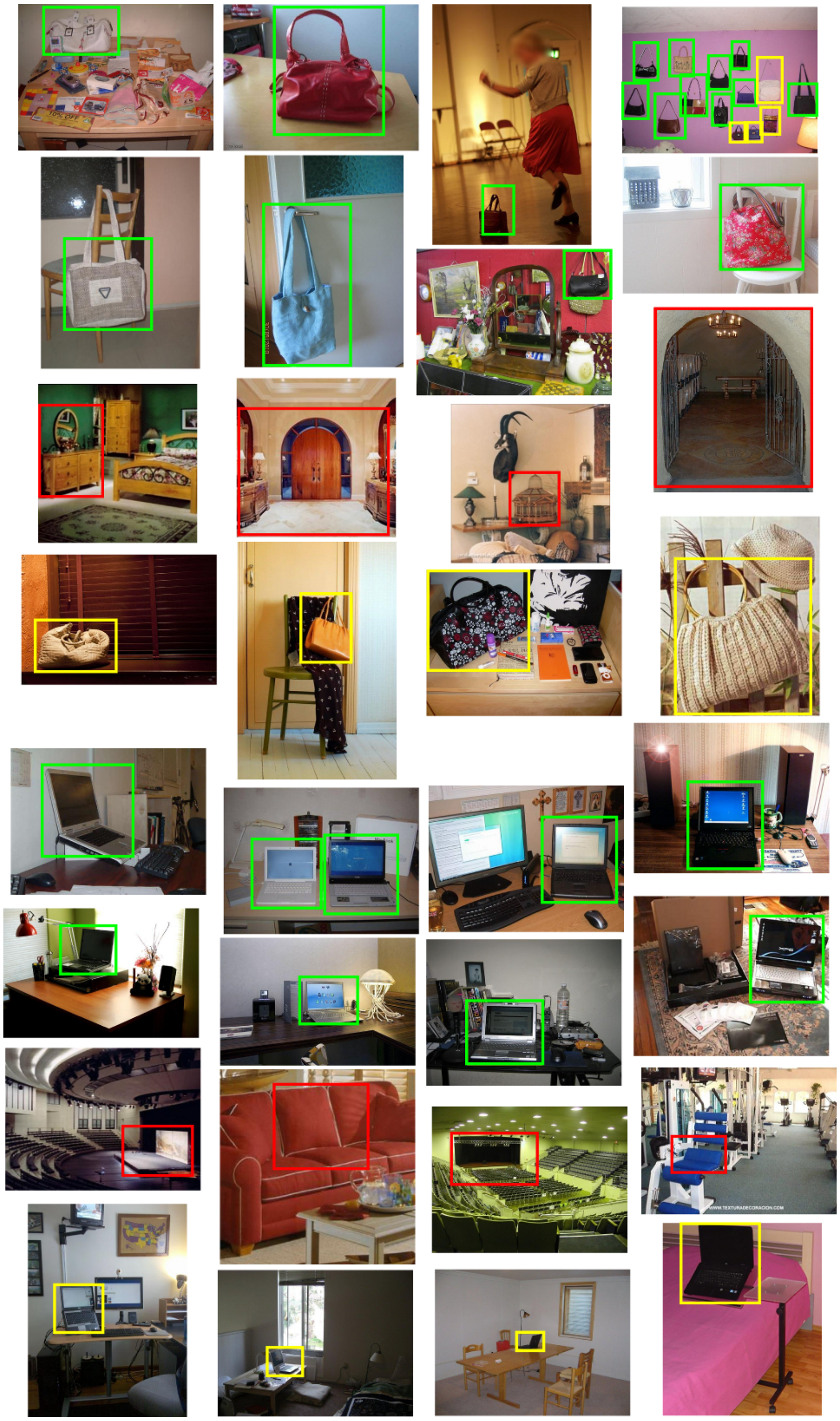


Figure 7.8: Example results of `handbag` and `laptop` detections (green≙true positive; red≙false positive; yellow≙false negative).

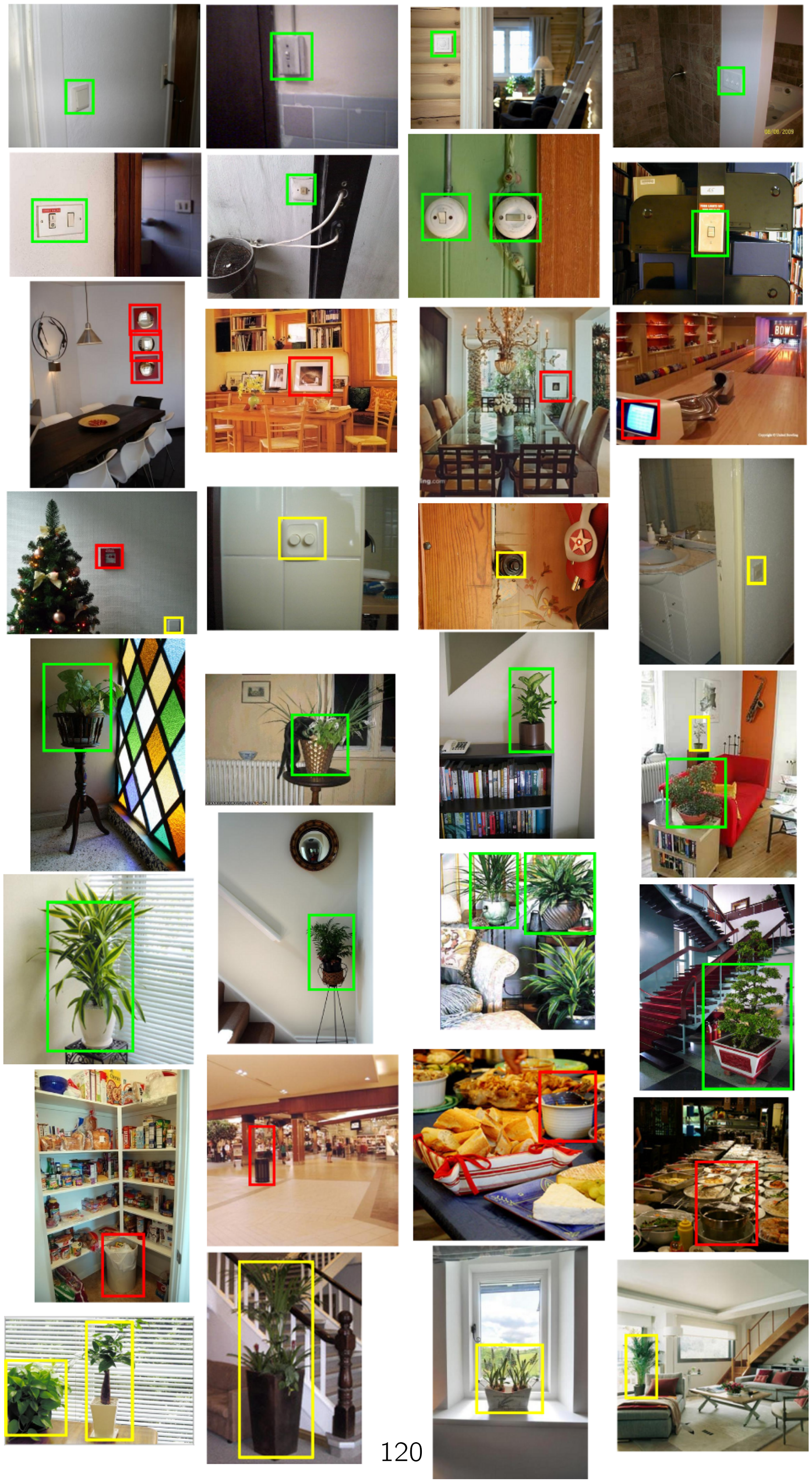


Figure 7.9: Example results of `light switch` and `potted plant` detections (green≙true positive; red≙false positive; yellow≙false negative).

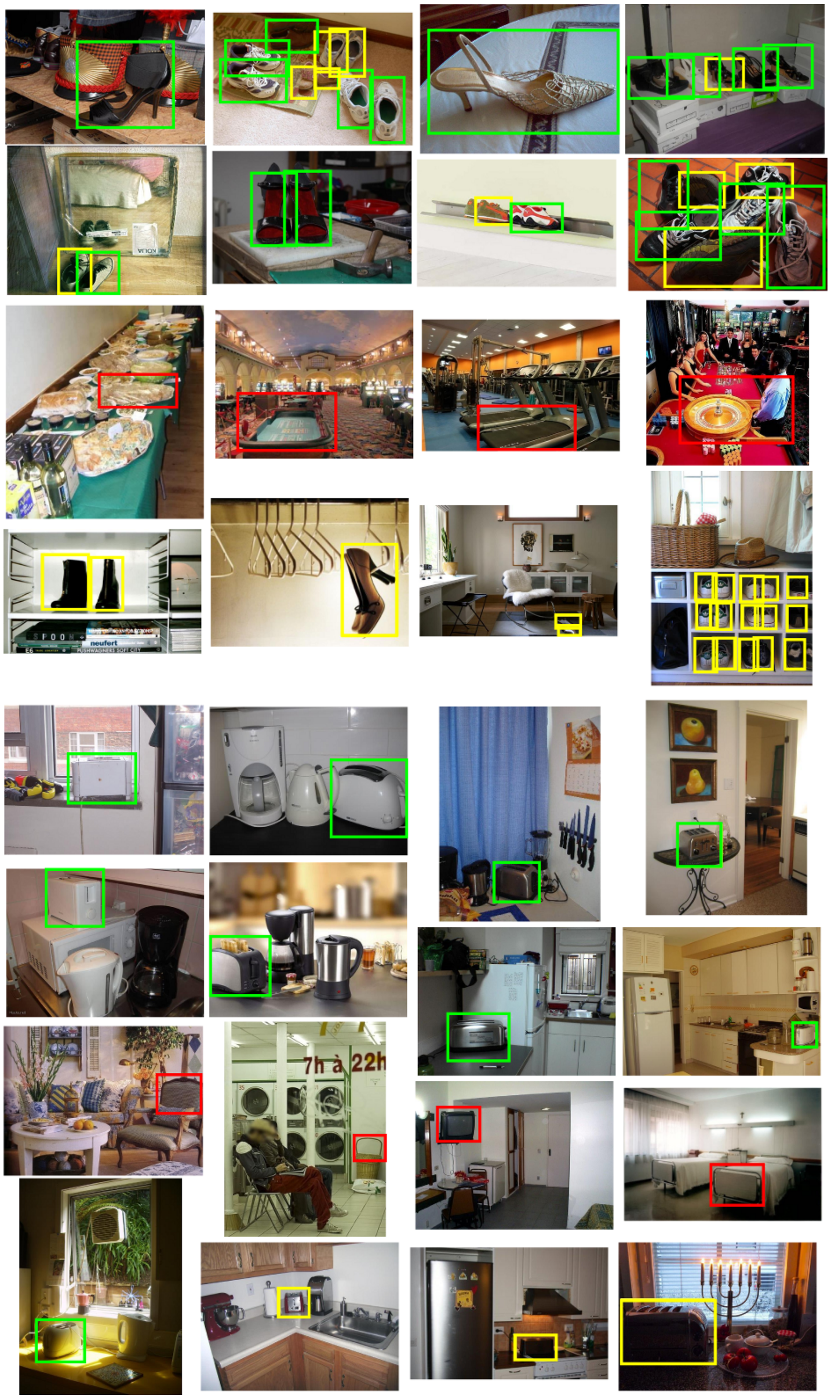


Figure 7.10: Example results of `shoe` and `toaster` detections (green≙true positive; red≙false positive; yellow≙false negative).

performance – measured by the average AP (AAP) – of the optimal WAM_6 model is comparable to those of the DPMs. This result shows that cascaded view-tuned SVM ensembles can achieve state-of-the-art detection performance. Looking at the scores for the different classes, it is clear that the DPM show the better performance for apple, bottle, handbag, laptop light switch and toaster. The WAM_6 model performs better for the categories bowl, cup, potted plant and shoe. Roughly speaking, it seems to be the case that the WAM_6 model works better for classes in which instances are cluttered or texture-rich (e.g. potted plant or shoe). On the other hand, the DPM seems to work better for classes with a well constrained edge-structure (e.g. apple or bottle). The categories that are most difficult to detect are light switches and shoes. One reason that light switches fall into this category might be that light switches are usually just white and do not provide may distinctive features, when viewed from a certain distance. Shoes may be difficult to detect because they can appear in quite unconstrained poses (e.g. when thrown on the floor), and the category has a huge in-class variability as well. The best detection performance can be found for the classes cup and handbag.

In figure 7.5, the precision-recall curves are provided for the DPM and WAM_6 model, for each of the domestic object categories. One interesting aspect, that can be observed for both models is that precision is usually quite high for the point with the maximal recall. In the PASCAL challenge [46], for some models precision drops significantly, when the recall increases. I suspect that this is the effect from using the large positive and in particular very large negative training sets during learning, because the system is forced to optimise the recall and precision at the same. This seems to lead to more cautious detectors, as opposed to detectors trained on smaller datasets. A similar effect can usually be achieved by increasing the detection threshold of the system (e.g. the confidence must be $> .8$), but it is obviously a good property of both detection systems that they optimise this objective implicitly. As expected from the results in table 7.2, the most different precision-recall curves can be found for apple, cup, shoe, potted plant and light switch. For the DPM, the lowest precision value at the highest recall value can be found for apple, cup, potted plant, shoe and light switch.

However, the precision values are always $\geq .6$. For the WAM_6 model, the lowest precision values at the highest recall can be observed for apple, bowl, light switch and shoe. Also here, the precision values are always $\geq .6$.

Examples of true positive, false positive and false negative detection results of the WAM_6 model for all categories are shown in the figure 7.6 (apple and bottle), 7.7 (bowl and cup), 7.8 (handbag and laptop), 7.9 (light switch and potted plant) and in figure 7.10 (shoe and toaster). The true positive detection results illustrate that the detection of novel object instances is possible in novel environments. The trained detectors are able to detect object instances, even at different scales, despite different appearances, poses, lighting conditions. On the one hand, for some false negatives it is not really clear, why the detector has not localised them, because they look quite similar to some of the true positive samples. On the other hand, there are also false negatives that obviously seem harder to detect, because of bad lighting, occlusion or small size. The false positives give an impression of what is learned. A rule of thumb is that a reasonable detector should produce false negatives that make sense visually. That means, the errors should occur on patterns that are similar to those of the samples. In the case of the cascaded view-tuned SVM ensembles, most of the negatives make sense that way. One can imagine that the miss-classified pattern provided feature vectors that are similar to the feature vectors of some samples. To a certain extent the false positives shown in the figures indicate where the limit of purely local detection methods might be. A rubbish bin have a shape just like that of a cup. However, to be able to distinguish between the two, additional features or information are needed. An interesting next step of future work is to integrate additional cues, e.g. of the object size and of the visual context, to be able to significantly enhance the results.

Remarks on the Systems Parameters and Runtime

Of course, the overall system has various parameters. However, only very few parameters are critical for the performance of the system. The choice of the concrete parameters is stated in the respective sections.

Often it was possible to use default parameters, because similar performances can be achieved with a range of different parameter settings. As seen in this section, an important parameter is the error variable m of the vector quantisation method in the principal view extraction, because it directly influences the number of extracted principal views. When comparing results for different numbers of principal views, the best and second best performance is found for six and three principal views respectivly. So, for new categories I suppose the usage of three or six principal views will produces reasonable results. Beyond that, as seen in this chapter, the choice of the ensembling method can be seen as an important parameter for the detetion with view-tuned SVM cascades. In this thesis, three different ensembling methods for the fusion of detection results have been compared, and the best performance has been achieved with the WAM ensemble. Thus, this ensembling scheme seems to be the best choice for novel categories. However, the study of other ensembling mehods is also an interesting topic for future research. The parameters for the sliding window search in this chapter are set as follows. The offset in x- and y-direction is set as .2 of the bounding box width and height respectivly, i.e. $xs = .2 = ys$. The scale space parameters are choosen heuristically and the following set of scales is employed: $\sigma = \{.25, .3, .35, .45, .6, .8, 1, .1.2, 1.4, 1.6, 2\}$. Also here, an interesting question for future research is how to choose the sliding window parameters such that the computation time is minimised, while retaining a reasonable performance at the same time.

The implementation of the view-tuned SVM cascade ensembles, as created as part of this thesis, is a prototype, that has been devloped as part of ongoing research. Although efficient algorithmic schemes are implemented (e.g. the efficient computation of the decision function), the code is not optimised by any means. This is because code optimisation is quite time consuming, and it should be done only after finishing the design of an algorithm (not as part of ongoing studies). As an example, the parallelisation of the prototype is only implemented sparsely at the process level. Therefore, all statements with respect to the computation time and the gained speed-up factors need to be considered with caution. Depending on the content of the image, the detection on a 640×480 image with a single view-tuned SVM cascade takes approx

$.5 - 1s$ on a single Intel Core i7 processor with 3.2 GHz. However, by employing multiple threads and an optimised implementation, I think it would be possible to achieve a (semi-)real-time performance. The next chapter concludes this thesis and provides an outlook on future reasearch.

Chapter 8

Final Conclusion and Outlook

This thesis deals with the problem of localizing domestic objects in images. The detection of domestic objects is motivated by the fact that various complex tasks that service robots may be engineered to perform in the future, require category detection abilities. For instance, domestic object detection allows a robot to perform category-level commands, to infer knowledge about novel objects, or to initiate interaction with humans in order to clarify things, in selective situations. Thus, the detection of domestic objects in images is a cornerstone of future research in the field of service robotics. This thesis contributes a novel system for the detection of domestic objects in images, which achieves state-of-the-art detection performance. The system demonstrates successful object detections, despite different scales, appearances, poses and lighting conditions. There are several aspects, by which the invariance of the model is implemented. First, the scale invariance is given by using the sliding-window approach at different scales of an input image. Second, there are invariances w.r.t. little occlusions, little pose changes and deformations, that are put forth by the coarse-to-fine matching of feature sets in the view-tuned kernels. For a single single SVM, the invariance is produced by storing support patterns of training examples with different appearances, poses and lighting conditions. Finally, the esembling of several view-tuned SVM experts adds invariance w.r.t. to different view-points of the objects. The system is trained on a newly created database of annotated domestic objects, which uses images from the Web. A distinct feature of this work is that the training and testing of the detection system concentrates exclusively on domestic objects. For example, a special `@HOME` tag is used for collecting tailored test image sets that show object instances in real home environments. Besides, an unsupervised method for decomposing complex training sets according to principal views is proposed. In addition, a method for learning specific view-tuned matching kernels is presented. Also, the application of the view-tuned kernels, in a C-SVM formulation and a sliding-window-based cascade is detailed. Finally, the ensembling of the detection results of different view-tuned SVM cascades is presented. Major contributions are made by the proposed system in the principal view extraction method and also the learning of view-tuned kernels, which generalises the pyramid matching of features sets. Other practices, like cascades, efficient computation of the decision function, and bootstrapping are then specifically adapted to be applicable with these kernels. Further, an efficient scheme for the extraction of local features on a grid is presented. In particular, the efficient exact computation of the decision function is a distinct feature of the system I propose. In contrast to other approaches, the efficient computation scheme guarantees the scalability of the method, even in the case of a very large number of support patterns. By allowing a large number of support patterns to be stored in the model, the learner offers an enormous capacity in practice (in other methods an increased capacity often comes along with an increased computational complexity). Scalability and efficiency are important properties, in particular in context of mobile robots with limited resources. Another feature of the approach outlined in this thesis is, that it is independent of the choice of the local visual features. That is to say, it allows to use other local feature descriptors within the same view-tuned kernel framework, which is not directly possible with other models like the DPMs. For instance, color features are interesting for domestic object detection, as many false positives (e.g. for apple or potted plant) should easily be avoidable by considering color information.

There are many topics in this thesis that are interesting for further work – some of them are mentioned in the thesis. Here, I want to state some directions, which I consider to be particularly important. First of all, I think there should be a systematic effort to create datasets with a large number of domestic object categories for training and for testing domestic of object detection systems within the context of service robotics. In particular, the images used for testing should come from the domain of application (e.g. photographs in indoor environments that are taken from a certain distance), to be able to assess the generalisation performance of different category detection systems. As shown in this thesis, even by using Web images, reasonable test data for domestic objects is hard to get, so an interesting question is: How can this data acquisition and annotation process be crowdsourced? Beyond that, I think it is important to incorporate additional cues into the object detection system. On the one hand, the false positives detection results presented in this thesis made sense because they often occurred on patterns that are similar to in-class patterns. On the other hand, for humans it is very easy to see that these patterns are false positives, because they incorporate knowledge about the object size, about the scene and so forth. In order to significantly improve the result of state-of-the-art detectors, the notion of purely local detectors should be discarded and the incorporation of additional cues for recognition should be considered. There is already some promising work being done in this direction. As a last aspect, it may be important for future systems to consider category learning with a human in the loop. The proposed system has interesting interaction possibilities at several stages. For instance, the results of the unsupervised principal view extraction could be used for interaction with a human tutor. Also the sequential discriminative optimisation in this thesis shows that the learning of powerful detectors can be considered as a sequential learning problem. By using an active learning loop, where the learner asks a human to label difficult examples, the system should be able to learn more quickly and be able to adapt to future unknowns.

Appendix A

A Limited Branching Tree Growing Neural Gas

This chapter presents the limited branching tree Growing Neural Gas (lbTreeGNG). Main parts of this chapter have already been published in [104] and [207]. At first, section A.1 shortly introduces the problem and sketches the basic idea of the algorithm. Subsequently, A.2 presents the the proposed approach in more detail. Finally, section A.3 provides and discusses the results of the algorithm an a variety of synthetic and real-world data sets.

A.1 Introduction

The Growing Neural Gas [73] (GNG) is a method for unsupervised online one-shot vector quantization of samples from an unknown distribution. By employing a Hebbian learning scheme, the GNG is able to learn codebooks while preserving the topological structure of the input space. Thus, the approach seems attractive for unsupervised learning problems in the field of computer vision, e.g. the learning of visual vocabularies. However, w.r.t. the large quantity of data that usually needs to be processed in vision tasks, two drawbacks of the GNG can be identified. First, the GNG keeps growing as long as new input samples are presented, which can lead to over-fitting. Second, the mapping of an input vector to a reference vector is quite expensive and requires an iteration to the entire codebook. Here, an hierarchical extension of

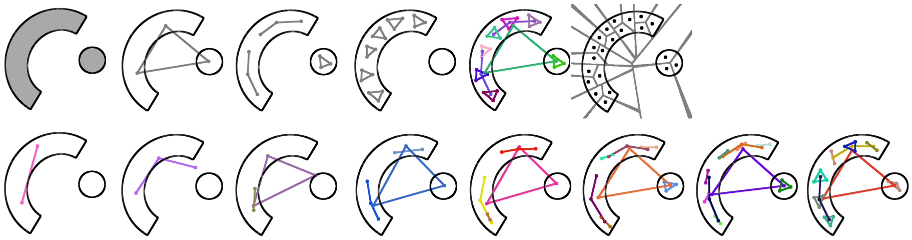


Figure A.1: *(First Row)*: In this example, the input vectors for the lbTreeGNG are uniformly sampled from the gray regions. After training with 20,000 samples, the first three levels of the lbTreeGNG are shown subsequently. Then the overall lbTreeGNG is presented, followed by the hierarchical Voronoi-Tessellation induced by the learned tree. *(Second Row)*: The second row shows the growth process of the lbTreeGNG during online learning. The corresponding iterations (from left to right) are 50, 125, 175, 250, 525, 750, 1975 and 3400. In the figure, colors of the GNGs are random. Further, points stand for nodes and lines between points stand for topological edges.

the GNG is proposed, called limited branching tree GNG (lbTreeGNG). The lbTreeGNG method allows to map input vectors to their reference vectors in sub-linear time and avoids over-fitting, while locally keeping track of the topology of the input space. Further, the lbTreeGNG contains the common GNG as a special case, and it keeps all of the normal GNG parameters. Within this chapter, it is shown that the algorithm works well on a variety of synthetic and real-world datasets, and even of non-stationary data. In doing so, input samples are mathematically represented by vectors ξ in a high dimensional *feature space* $\mathcal{F} = \mathbb{R}^d$. Any feature ξ that is processed is sampled from an unknown distribution $P(\xi)$, i.e. the overall distribution of all possible features in $\mathcal{F}$.

The Basic Idea

To get a first intuition of the outcome of the lbTreeGNG algorithm, consider the example in figure A.1. The first row shows a lbTreeGNG after processing 20,000 input signals uniformly sampled from the gray regions. Snapshots from the process of growth can be seen in the second row. Given a maximum branching factor $b = 3$, the method starts with a single GNG in the first level of the tree. This GNG grows until it has b nodes. Subsequently, each of these nodes initializes a new local GNG in it's Voronoi-Cell. Then the GNGs in the second level grow until they reach a size of b. The initialization and growing of GNGs in successive levels continue until a predefined node error m for the approximation is reached. Hence, GNGs in the third level are only created at positions that are not approximated well enough by the second level. The last two images in the first row show the overall lbTreeGNG and the hierarchical Voronoi-Tessellation induced by the codeword tree. In processing, the sampled input vectors are passed from the root GNG through the tree down to it's nearest leaf node. Thereby, an input vector induces a major adaption in it's corresponding leaf GNG and also propagates slight adaptions along the path up to the root GNG. As visible, the hierarchical space tessellation directly supports an efficient mapping from input vectors to the codewords. In this simple example with 21 leaf nodes, the number of distance computations is $O(b \cdot depth) = 9$ in worst-case.

A.2 Approach

In this chapter, the details of the lbTreeGNG are provided. In doing so, a notation similar to the the original one of [73] is used. For a node x, the associated reference vector is denoted with w_x. Note, that b and m are the newly introduced parameters. The parameter b limits the maximal branching factor of the learned codeword tree and thus controls the complexity of the method. The parameter m is the error threshold that effects the quantization error and guards the learner against over-fitting. The original GNG parameters remain untouched and could be used with the following default values as proposed in [73]: $\epsilon_b = .2$, $\epsilon_n = .006$, $a_{max} = 50$, $\alpha = .5$, $d = .995$ and $\lambda = 100$. All parameters in the method are used globally for all GNGs and do not change over time.

Main Loop

The lbTreeGNG is initialized with a single GNG. In the main loop of the algorithm, an input ξ is generated according to $P(\xi)$ and it is passed from the root level GNG $l = 0$ to a leaf GNG on level $l = L$ following the nearest reference vectors. For each level $l = 0, \cdots, L$ it is kept track of the winner nodes $s_{1,l}$ and second winner nodes $s_{2,L}$. Thereafter, adaptions are done on the winning leaf GNG of $s_{1,L}$ and the intermediate GNGs of $s_{1,l}$ with $l = 0, \ldots, L-1$. This strategy of passing vectors down a hierarchical tree, yields to very efficient (sub-linear) runtime complexity and is inspired from related work in computer vision [148] [31].

Adapt Winning Leaf GNG

Using $s_{1,L}$ and $s_{2,L}$ the algorithm does the same steps as the GNG algorithm [73], i.e. increment the edge age, increase the node error, move $s_{1,L}$ and $s_{2,L}$ towards the input, check the edge connectivity and the edge age. Then, every λ iteration some some extended processing is performed. First, to continue growth only in poorly approximated regions, a check is added whether the maximum node error of the leaf GNG is bigger than the parameter m. Only if this condition is true, the method continues to refine the current codeword tree. The strategy to stop growing, is inspired by the work of [131]. Going on in the process, a point is reached where one wants to create a new node and where the following three basic situations are discriminated. Note, that the maximal branching factor of the resulting codeword tree is controlled to keep track of the worst-case runtime. Again, this is inspired from works using hierarchical k-means with a fixed branching factor in computer vision [148]. The explicit restriction of the branching factor to impose the worst-case complexity is also a major feature of the presented method in comparison to other hierarchical GCS [21] [91] and GNG [42] approaches. In the following, let q denote the node with the highest error in the GNG of $s_{1,L}$ and f the node with the second highest error f.

Create a New Node

If the GNG of $s_{1,L}$ has less than b nodes, a new node r is created in the common way [73] halfway between the node with the highest error

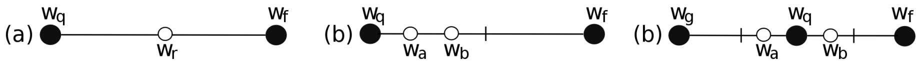


Figure A.2: Different cases in the node creation process. The variable q denotes the node with the highest error in a GNG, f is the neighbor node of q with the highest error and g the neighbor with the second highest error (if existing). Further, a and b are the initial nodes of a new child GNG that is inserted in the Voronoi-Cell of q. For any node x, w_x denotes the corresponding reference vector. (a) shows how a new node r is inserted in common GNGs. (b) and (c) visualize the two cases in which initial nodes a and b are created on q's topological edges within the Voronoi-Cell of q.

q and the node with the second highest error f. This case is visualized in figure A.2 (a). The corresponding equation is:

$$w_r = .5(w_q + w_f). \tag{A.1}$$

Create a New GNG

If the branching factor of the GNG is equal to the parameter b, it becomes necessary to insert a new child GNG for the node with the highest error. For a new GNG, two initial nodes must be created. Like the creation of new nodes in common GNGs [73], this follows the principle of accumulated error minimization and places these initial nodes on the topological edges of q inside the Voronoi-Cell of q. Thereby, q has either one neighbor f or more neighbors. Let $g \neq f$ denote the neighbor node of q with the second highest error, if it is existing. The two different cases that arise can be seen in figure A.2 (b) and (c). The corresponding equations are:

$$w_a = .875 w_q + .125 w_f \qquad w_b = .625 w_q + .375 w_f \tag{A.2}$$

$$w_a = .750 w_q + .250 w_f \qquad w_b = .750 w_q + .250 w_g. \tag{A.3}$$

Adapt Intermediate GNGs

Since the global tree should remain flexible at any time, it is necessary to adapt the GNGs of intermediate nodes as well. These nodes were traveled while passing the input vector down the tree and while walking through the nearest node in each GNG. The adaption is done in a winner-takes-all fashion and all winner nodes $s_{1,l}$ of intermediate levels $l = 0, \dots, L-1$ are moved towards the input ξ:

$$\Delta w_{s_{1,l}} = \beta(\xi - w_{s_{1,l}}). \tag{A.4}$$

For a reasonable choice of β the following aspects must be considered. On the one hand, β should be sufficiently smaller than ϵ_b (the factor for the adaption of a winning node in a leaf GNG) to avoid pulling nodes out of their enclosing Voronoi-Cells in higher levels. On the other hand, the number of adaptions that are received in upper layers grow exponentially with b, e.g. in a balanced tree a node at level l can receive up to b^{L-l} updates. Hence, β should incorporate this by allowing gradually less movements in higher levels of the tree. The following choice of β takes both discussed aspects into account:

$$\beta = \epsilon_n b^{-(L-l+1)}. \tag{A.5}$$

A.3 Results and Discussion

This subsection demonstrates the behavior of the algorithm on different artificial and real world datasets. The results are related to other methods and especially show the influence of the introduced parameters b and m.

Experiment 1: Stationary Artificial Data.

In this experiment the input distribution known from the example in figure A.1 is used. Also, the default values for the GNG parameters as stated in section A.2 are used and either b or m is varied to demonstrate the resulting effects. The visual summary can be found in figure A.3. The number of iterations is $20,000$. In a second experiment, the output of the algorithm is presented on two other simple input distributions in figure A.4, with $b = 3$ and $m = 1.33$.

In the left part of figure A.3 the effect of altering the maximum branching factor b can be seen, while keeping all other parameters constant. If $b = INF$, then the algorithm only has one GNG in the root

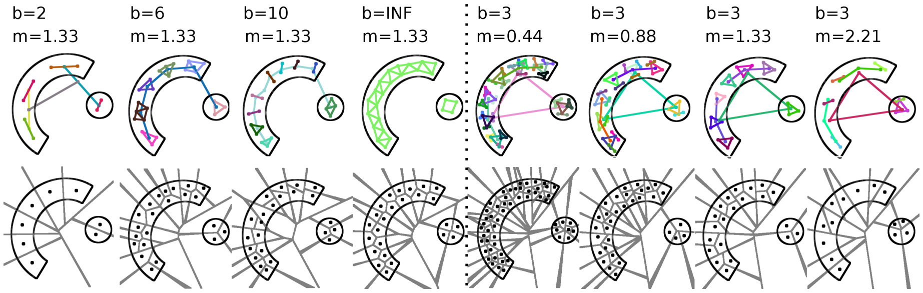


Figure A.3: On the new lbTreeGNG parameters. (*left*:) Shows the effect of changing the parameter *b* while keeping all other parameters constant. (*right*:) Shows the effect of changing the parameter *m* while keeping all other parameters constant.

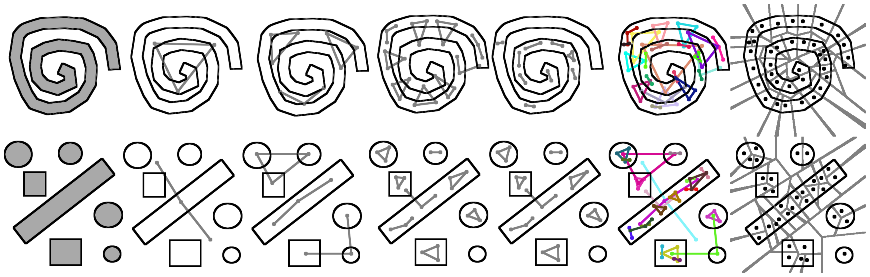


Figure A.4: Results of the lbTreeGNG algorithm on other simple distributions. (*row-wise*:) The Input examples are sampled from the gray regions. The four levels of the learned lbTreeGNG are shown subsequently. Then the complete lbTreeGNG is shown, followed by the hierarchical Voronoi-Partition.

level. Smaller values like $2, 6, 10$ induce different hierarchical Voronoi-Tessellations. It can be seen, that the parameter *b* effects the size of the local topologies and results in trees ranging from binary trees up to flat GNGs. The parameter is essential for the worst-case runtime complexity: Finding the nearest corresponding codeword for an input vector ξ takes at least $O(bL)$ distance calculations, where L is the maximum depth of the tree. The other parameter *m* controls the degree of approximation that is reached by the leaf nodes of the lbTreeGNG. The effects of varying *m* can be seen in the right part of Figure A.3. A low error threshold leads to a large number of leaf nodes which realize a dense approximation of the seen input data. A high error threshold leads to a more sparse approximation by few codewords.

Experiment 2: Stationary and Non Stationary Artificial Data.

As stationary artificial input distribution, a two-dimensional data distribution copying the one used for the evaluation of SOINN [74] was chosen. It comprises two Gaussian components (A and B), two ring-shaped components (C and D), and a sinusoidal component (E) composed from three subcomponents ($E1$, $E2$, and $E3$). Each component encompasses 18,000 individual samples. Additionally, the input distribution includes uniformly distributed random noise amounting to 10% of the total sample number (10,000 samples). This dataset was used to train four different types of networks: Fuzzy ART [26], IbTreeGNG, SOINN [74], and TopoART [206]. Figure A.5 depicts the applied data distribution and the respective results.

As Fuzzy ART constitutes the basis of TopoART, it was analysed first. For comparison reasons, β was set to 1. Therefore, the weights of the best-matching neurons are adapted in the same manner as with TopoART. ρ was selected in such a way as to roughly fit the thickness of the elliptic and inusoidal components of the input distribution. As this network does not possess any means to reduce the sensitivity to noise, virtually the whole input space was covered by categories.

In contrast to Fuzzy ART, both TopoART components created representations reflecting the relevant regions of the input distribution very well. This is remarkable since the value of ρ_a was equal to the value of the vigilance parameter ρ of the Fuzzy ART network. The representation of TopoART was refined from TopoART a to TopoART b: While TopoART a comprises one cluster, TopoART b distinguishes five clusters corresponding to the five components of the input distribution. By virtue of the filtering of samples by TopoART a and due to the fact that ρ_b is higher than ρ_a, the categories of TopoART b reflect the input distribution in more detail. This property is particularly useful if small areas of the input space have to be clustered with high accuracy. Here, TopoART a could filter input from other regions and TopoART b could create the desired detailed representation.

The IbTreeGNG network was trained with the default values for the

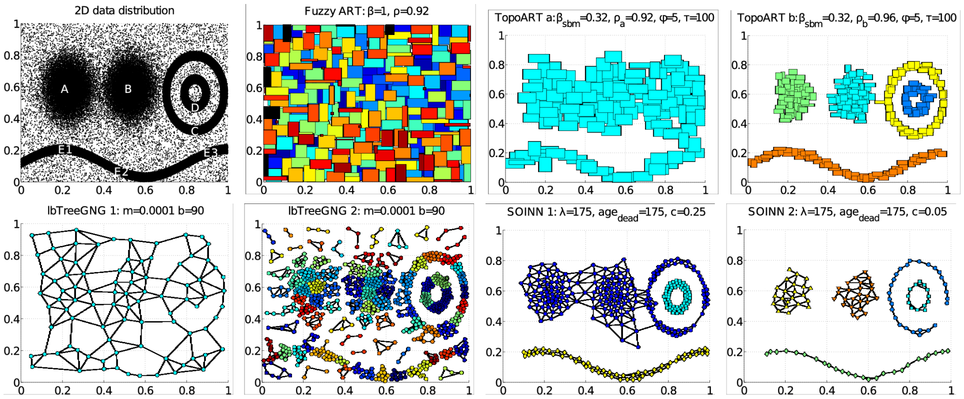


Figure A.5: Data distribution and results of several types of neural networks. Due to the noise contained in the data, Fuzzy ART covered virtually the complete input space with its rectangular categories. In contrast, TopoART learned a noise-insensitive representation in which the categories were summarised to arbitrarily shaped clusters. The representation of TopoART *a* was refined by TopoART *b*. Here, all categories of an individual cluster are painted with the same colour. The first level of the lbTreeGNG network represents the input space globally. At the second level, the representations are locally refined and the topological structure is locally maintained. Noise regions are represented by the lbTreeGNG network as well. But the node density is much lower than in the relevant regions of the input space. Finally, the data distribution was successfully clustered by SOINN. Here, the representation is refined from SOINN 1 (first layer) to SOINN 2 (second layer). Reference vectors belonging to the same cluster share a common colour and symbol.

parameters ϵ_b, ϵ_n, $a_{\max}$, α, d, and λ as stated in [104]. In addition, the parameter b for the limiting branching factor is set to 90 and the error threshold m is set to .0001. Here, the branching factor is chosen relatively large in order to produce a two-layered codeword tree providing results comparable to TopoART and SOINN. As can be seen in Fig. A.5, the levels 1 and 2 of the lbTreeGNG network, denoted by lbTreeGNG 1 and lbTreeGNG 2 respectively, reasonably capture the topological structure of the input space. Since noise is a significant part of the input distribution, the lbTreeGNG system learns codewords in those noisy regions as well. However, the learnt GNG networks show a much higher resolution in relevant parts of the input distribution. Due to the hierarchical space partitioning of the network, a single GNG network in the second layer only encodes local topological structures within a single Voronoi cell of the first layer. In contrast to SOINN and TopoART, lbTreeGNG does not directly provide a labeling of clusters. Rather the labels are implicitly represented by the hierarchical taxonomy.

For SOINN, the values of λ, age_{dead}, and c were selected in such a way that results comparable to those published in [74] were achieved. Here, individual parameter settings for both layers (SOINN 1 and SOINN 2) were allowed. Furthermore, the settings for α_1, α_2, α_3, β, and γ were directly adopted from [74] for both layers (1/6, 1/4, 1/4, 2/3, 3/4). Figure A.5 shows that SOINN, was able to create a hierarchical representation of the input distribution: The three clusters of SOINN 1 were refined by SOINN 2 which distinguishes five clusters. Similar to TopoART b, SOINN 2 exhibits a reduced sensitivity to noise.

In a second experiment, TopoART, lbTreeGNG and SOINN are compared regarding to their ability to represent changing data distributions. In doing so, the respective networks were successively trained with samples from the subdistributions $A+E3$, $B+E2$ and $C+D+E1$ (cf. Fig. A.5). As in the previous experiment, the subdistributions include 10% of uniformly distributed random noise as well. Each row in Fig. A.6 depicts snapshots of the different networks after training with the corresponding data.

In this experiment the lbTreeGNG uses the same parameters that has been used for the stationary input data. Figure A.6 shows that the lbTreeGNG system creates and maintains a reasonable codebook

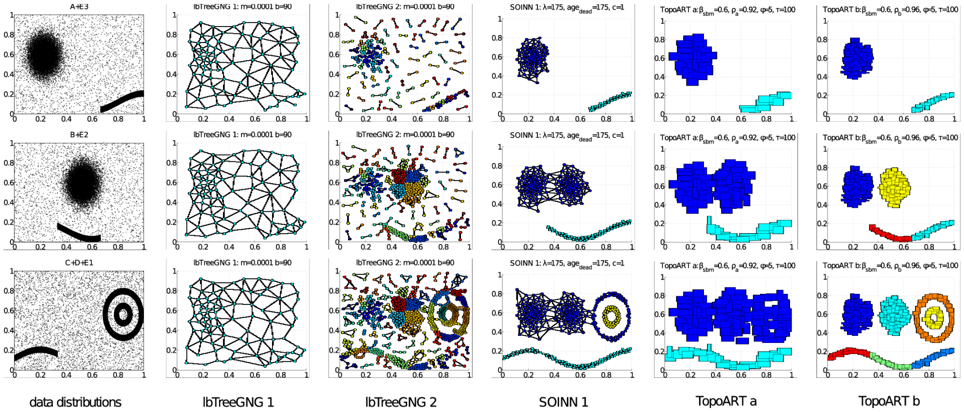


Figure A.6: Training results for a changing data distribution. In order to simulate non-stationary data, the networks were successively trained with samples from the subdistributions A+$E$3, B+$E$2, and C+D+$E$1, which are depicted in the leftmost column. Each row shows the formed representations of all considered networks after finishing the respective training period. Here, each cluster of SOINN and of TopoART as well as each connected component of lbTreeGNG has been drawn using an individual colour. All networks were able to incrementally incorporate the new data. The representations created by SOINN and by TopoART are stable; i.e., learned structures are not forgotten if the input distribution changes. In contrast, representations learned by lbTreeGNG is still altered during the learning process, for instance, the representation of subdistribution $E$3 created by lbTreeGNG 2.

over time. Similar to the results for stationary data, it can be observed that the topological structure of the input space is locally preserved and that relevant regions are represented with a much higher resolution than noise regions. As the data distribution changes over time, the node density is adapted accordingly. As a result, lbTreeGNG can learn novel or modified data distributions and already represented structures may be altered. This effect can be observed by comparing the different representations of $E$3 created by lbTreeGNG 2, for example. However, for deeper lbTreeGNGs the upper levels get more and more stable since the adaption rule of intermediate winning nodes allows gradually less plasticity in higher levels of the tree. In comparison to the results of the previous experiment, the size of the leaf GNGs in the regions A and $E$3 has decreased. The explanation for this is that the network tries to grow in breadth before it grows in depth. Thus, while learning $A+E3$ the network creates a higher resolution in the first layer since it has capacity left. As a consequence, smaller leaf GNGs are produced at the second level.

As the second layer of SOINN can only be trained after the first layer has finished learning, only the first layer (SOINN 1) could be applied to learn the non-stationary data. The results resemble the results obtained in the previous experiment (cf. Fig. A.5). But here, the respective clusters were incorporated subsequently, depending on the current data distribution. Learned representations remained virtually stable and were only slightly modified due to noise.

Finally, figure A.6 shows that both components of TopoART incrementally learned the presented input. Similar to SOINN, already created representations remained stable when the input distribution changed. As in the stationary case, TopoART b performed a refinement of the representation of TopoART a. But here, the sub-regions $E1$, $E2$, and $E3$ were separated, since the corresponding input samples were presented independently and could not be linked. TopoART a was able to compensate for this effect, as its lower vigilance parameter ρ_a allowed for larger categories which could form connections between the sub-regions.

Experiment 3: Real World Global Descriptors.

This experiment aims at two things. First of all, it tries to show that the effects of the parameters b and m (that was seen in experiment 1) also transfer to learning in high dimensional spaces. Second, the experiment tries to relate the outcome to a well known baseline method. The experiment employs a pseudo-streams of global descriptors from the following well known datasets.

The MNIST training data set [114] is used which contains 60,000 images of handwritten digits. For each gray-scale image of size $28\times28\times1$ a column vector $v \in \mathbb{R}^{784}$ is constructed with normalized intensities in $[0, 1]$. Further, the 1.5 mil. tiny images data set provided by [203] is used. The data set contains $1.5 \cdot 10^6$ color images of size $32 \times 32 \times 3$. Roughly speaking, the images have been gathered from the

Table A.1: Results on real data sets. `avRecErr`: average reconstruction error for a pixel; `avRecErrVar`: variance of avRecErr; `numOfLeafs`: number of leafs/ codebook size; `maxDepth`: maximum depth of codebook tree; `wcRT`: worst-case runtime for mapping a vector to a leaf codeword $O(maxDepth \cdot b)$; `numOfGNGs`: number of GNGs in the IbTreeGNG; `avBrachFac`: average branching factor of the codebook tree.

IbTreeGNG												
Data	MNIST						1.5 Mil. Tiny Images					
b	3	6	INF	3	3	3	3	6	INF	3	3	
m	10	10	10	30	50	70	150	150	150	70	80	120
`avRecErr`	.0066	.0065	.0062	.0069	.0076	.0084	.0031	.0031	.0030	.0030	.0030	.0031
`avRecErrVar`	.0022	.0021	.0015	.0019	.0015	.0014	.0021	.0021	.0021	.0020	.0020	.0021
`numOfLeafs`	478	505	602	316	120	26	2769	2802	3066	9509	8570	4673
`maxDepth`	12	22	1	12	9	7	18	28	1	19	18	17
`wcRT`	36	132	602	36	27	21	54	168	3066	57	54	51
`numOfGNGs`	297	180	1	193	72	15	1714	1004	1	5817	5286	2866
`avBrachFac`	2.6	3.8	602	2.6	2.7	2.7	2.6	3.8	3066	2.6	2.6	2.6
k	478	505	602	316	120	26	2769	2802	3066	9509	8750	4673
`avRecErr`	.006	.006	.0059	.0062	.0065	.0073	.0029	.0029	.0029	.0028	.0029	.0029
`avRecErrVar`	.0017	.0018	.0017	.0018	.0017	.0015	.0021	.0021	.0021	.0020	.0020	.0020
k-means												

Web by pumping a dictionary of English words into different image search engines. Thus, the tiny images contain a huge diversity of scenes and objects. For each image a column vector in $\mathbb{R}^{3072}$ is created with normalized intensities in $[0, 1]$.

As before, the default parameters for the GNGs are used and the values for parameter b and m are varied respectively. The quality of the structure of hierarchical codebook is assessed in terms of different key numbers, e.g. the average pixel-wise reconstruction error on a data set given a learned codebook (`avRecErr`). Further, in order to provide a baseline for the reconstruction error results are presented on the data sets using an offline learning k-means. For the overall view at the experiments and the results see table A.1. The results in each column of the table are averaged over 10 runs of the algorithms and rounded appropriately.

On the simulated data, it is clear that the maximum branching factor b directly effects the worst-case mapping runtime. For $b = INF$ the algorithm produces a flat non-hierarchical GNG. The real data supports the observations by the numerical values in table A.1. It can be seen, that for both data sets that the worst-case runtime `wcRT` for mapping a vector to a leaf codeword grows as *b* goes up. The computational benefit from the hierarchical structure gets bigger when the codebooks are larger, especially in comparison to flat GNGs. Also before, it has been shown on the simulated data that the *m* parameter effects the approximation of the input distribution. On the real data sets this is measured in terms of `avRecErr` and `avRecErrVar` in table A.1. Given a constant *b*, it can be seen for both data sets that `avRecErr` and `avRecErrVar` increase as *m* gets larger. The performance in terms of `avRecErr` and `avRecErrVar` is comparable to k-means. The difference between IbTreeGNG and k-means reconstruction error is usually not large in terms of image gray-scale values, e.g. less than a single gray-scale value per pixel for the first column of the table. Overall, the results show that the introduced parameters can be used to control the underlying trade-off between speed and accuracy, while maintaining reasonable codebooks. Figure A.7 shows some exemplary visual codewords.

Experiment 4: Real World Global Descriptors

This experiments aims at assessing the behavior of the IbTreeGNG algorithm on local image descriptors over time. Therefore, three different Web-image datasets together [44] [183] [82], resulting in a collection of approx. 90,000 images. The images were scaled to approx. $100,000$ pixels. From the overall image collection, we use 75,000 images as a training set and sample 15,000 images as a test set. Totally, a pseudo-stream of approx. 44 million SURF [7] descriptors is processed while learning. During training, we repeatedly measure the reconstruction error on a the test set and the current number of leaf nodes. The results are averaged over multiple runs.

In the plot in figure A.7 we see the characteristic progress of the test reconstruction error and the number of leaf nodes during learning on a stream of $4.4 \cdot 10^7$ SURF descriptors. In the beginning, by adding few nodes the learner yields a large drop down of the reconstruction error. After that, on a trained IbTreeGNG much more nodes need to be added to the codebook to realize relatively small error reductions. Finally, a saturation of the growth and of the reconstruction error can

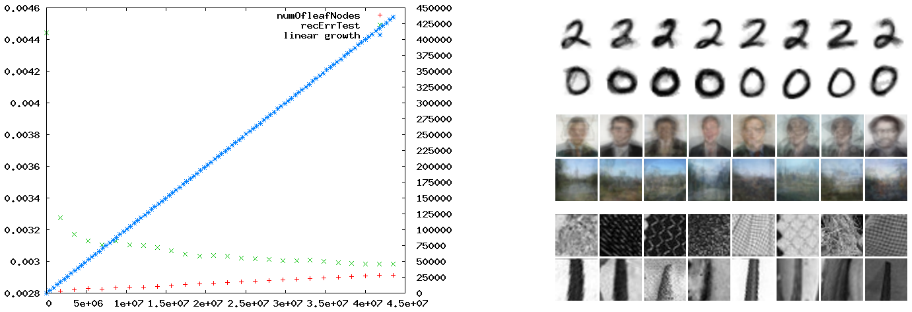


Figure A.7: (*left-side*) The left plot shows the characteristic progress of the test reconstruction error (*green*) and the number of leaf nodes (*red*) during learning on a stream of approx. $4.4 \cdot 10^7$ SURF descriptors. (*right-side*) The right side exemplary shows vizualisations of some topologically neighbored leaf codewords on the different datasets: MNIST (*top*), Tiny Images (*middle*), merged SURF (*bottom*).

be seen. Figure A.7 exemplary presents some visual codewords.

In this chapter, a hierarchical extension of the GNG [73] has been presented. The method added two additional parameters b and m, while preserving all of the original GNG parameters. As presented, the parameter b limits the branching factor codebook tree and allows an efficient assignment of input vectors to codewords. Further, as shown, the parameter m is an error threshold, that affects the quantization error and guards the learner against over-fitting. The influence of the parameters s been discussed using a variety of synthetic and real-world datasets. Overall, the approach locally preserves the topological structure of th input space and allows efficient classification of novel input signals.

Appendix B

Supplementary Material

Listing B.1: Example of a XML annotation for a retrieved image.

```
<tns:Annotation xsi:schemaLocation="OwnDataType.xsd">
<tns:Image-Info>
    <tns:Category>CUP</tns:Category>
    <tns:English-Search-Key>cup</tns:English-Search-Key>
    <tns:Translated-Search-Key>beker</tns:Translated-Search-Key>
    <tns:Language>DUTCH</tns:Language>

    <tns:Provider>P1</tns:Provider>
    <tns:Provider-URL>
        http://www.style-files.com/images/bekersikea500x333.jpg
    </tns:Provider-URL>
    <tns:Rank>93</tns:Rank>
    <tns:Retrieval-Date>2010-8-31T2:54:53</tns:Retrieval-Date>

    <tns:Filename>b89c0c9:12ac7888154:-7ec3.jpg</tns:Filename>
    <tns:Size Height="333" Width="500" Depth="3"/>
    <tns:Type>jpg</tns:Type>
</tns:Image-Info>

<tns:Annotation-Info>
    <tns:Metadata Name="CUP"/>
    <tns:Bounding-Box X-Min="281" X-Max="457" Y-Min="109" Y-Max="307"/>
</tns:Annotation-Info>

<tns:Annotation-Info>
    <tns:Metadata Name="CUP"/>
    <tns:Bounding-Box X-Min="153" X-Max="310" Y-Min="68" Y-Max="243"/>
</tns:Annotation-Info>
</tns::Annotation>
```

Algorithm 2 `REDUCE`$(G; (B, Q, \underline{R}))$

```
$G^r \leftarrow G$
intersect $\leftarrow$ **true**
**while** intersect **do**
   **for** $i = 1$ **to** $|G^r|$ **do**
      **for** $j = i + 1$ **to** $|G^r|$ **do**
         intersect $\leftarrow$ BOXES_INTERSECT$(b_i, b_j)$
         **if** intersect **then**
            $b_{\mathtt{new}} \leftarrow B(b_i, b_j, \vec{q}_i, \vec{q}_j, \underline{\vec{r}}_i, \underline{\vec{r}}_j)$
            $\underline{\vec{r}}_{\mathtt{new}} \leftarrow \underline{R}(\underline{\vec{r}}_i, \underline{\vec{r}}_j, \vec{q}_i, \vec{q}_j)$
            $\vec{q}_{\mathtt{new}} \leftarrow Q(\vec{q}_i, \vec{q}_j, \underline{\vec{r}}_i, \underline{\vec{r}}_j)$
            REMOVE$(G^r, (b_i, \vec{q}_i, \underline{\vec{r}}_i))$
            REMOVE$(G^r, (b_j, \vec{q}_j, \underline{\vec{r}}_j))$
            ADD$(G^r, (b_{\mathtt{new}}, \vec{q}_{\mathtt{new}}, \underline{\vec{r}}_{\mathtt{new}}))$
            **break**
            **break**
         **end if**
      **end for**
   **end for**
**end while**
**return** $G^r$
```

Algorithm 3 `BOXES_INTERSECT`$(b_1, b_2, \theta \leftarrow .5)$

```
$a_o \leftarrow \frac{\mathtt{AREA}(b_1 \cap b_2)}{\mathtt{AREA}(b_1 \cup b_2)}$
**if** $a_o > \theta$ **then**
   **return** **true**
**end if**
**return** **false**
```

Algorithm 4 `FUSE`$(G^r; F)$

```
$G^f \leftarrow \{\}$
**for** $i = 1$ **to** $|G^r|$ **do**
   $(b_{\mathtt{new}}, \underline{r}_{\mathtt{new}}) \leftarrow F(b_i, \vec{q}_i, \underline{\vec{r}}_i)$
   **if** $\underline{r}_{\mathtt{new}} > 0$ **then**
      ADD$(G^f, (b_{\mathtt{new}}, \underline{r}_{\mathtt{new}}))$
   **end if**
**end for**
**return** $G^f$
```

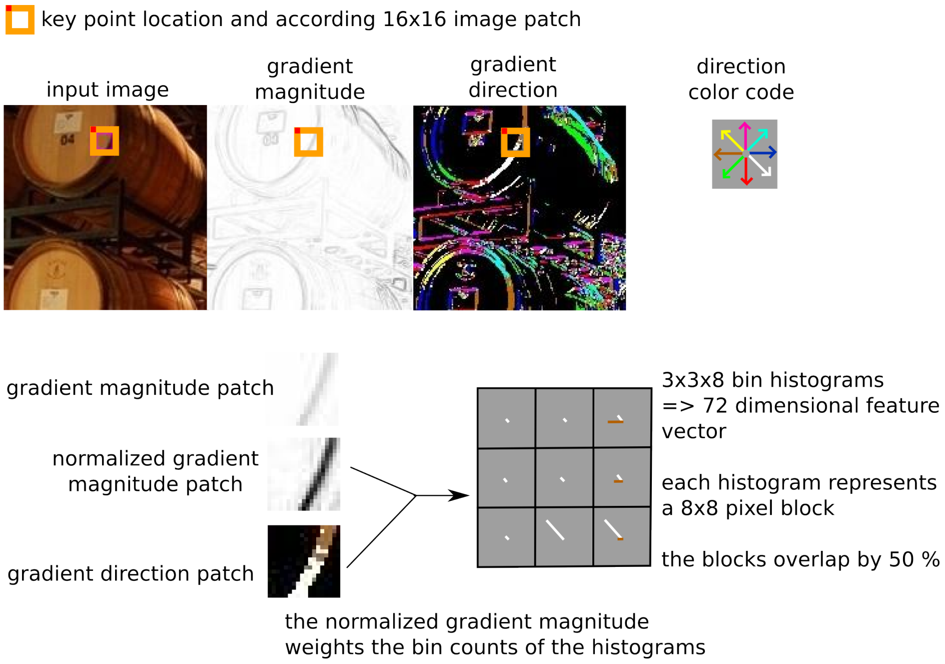

key point location and according 16x16 image patch
input image
gradient magnitude
gradient direction
direction color code
04
gradient magnitude patch
normalized gradient magnitude patch
gradient direction patch
3x3x8 bin histograms
=> 72 dimensional feature vector
each histogram represents a 8x8 pixel block
the blocks overlap by 50 %
the normalized gradient magnitude weights the bin counts of the histograms

Figure B.1: This figure sketches the 72-dimensinoal HOG features that are used as to describe the gradient structure in an image patch aroung a key point location.

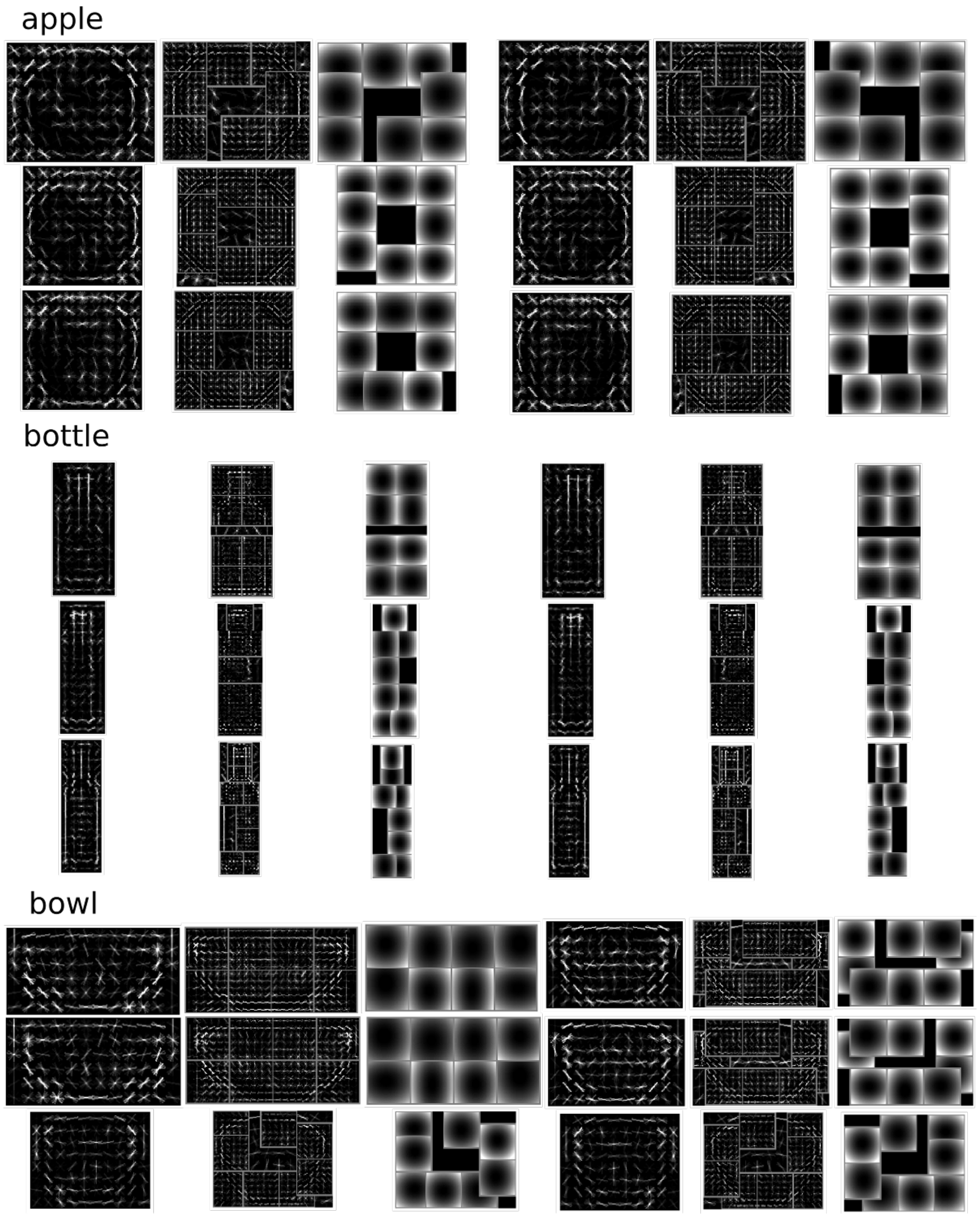


Figure B.2: This figure visualizes the DPMs of apple, bottle and bowl.

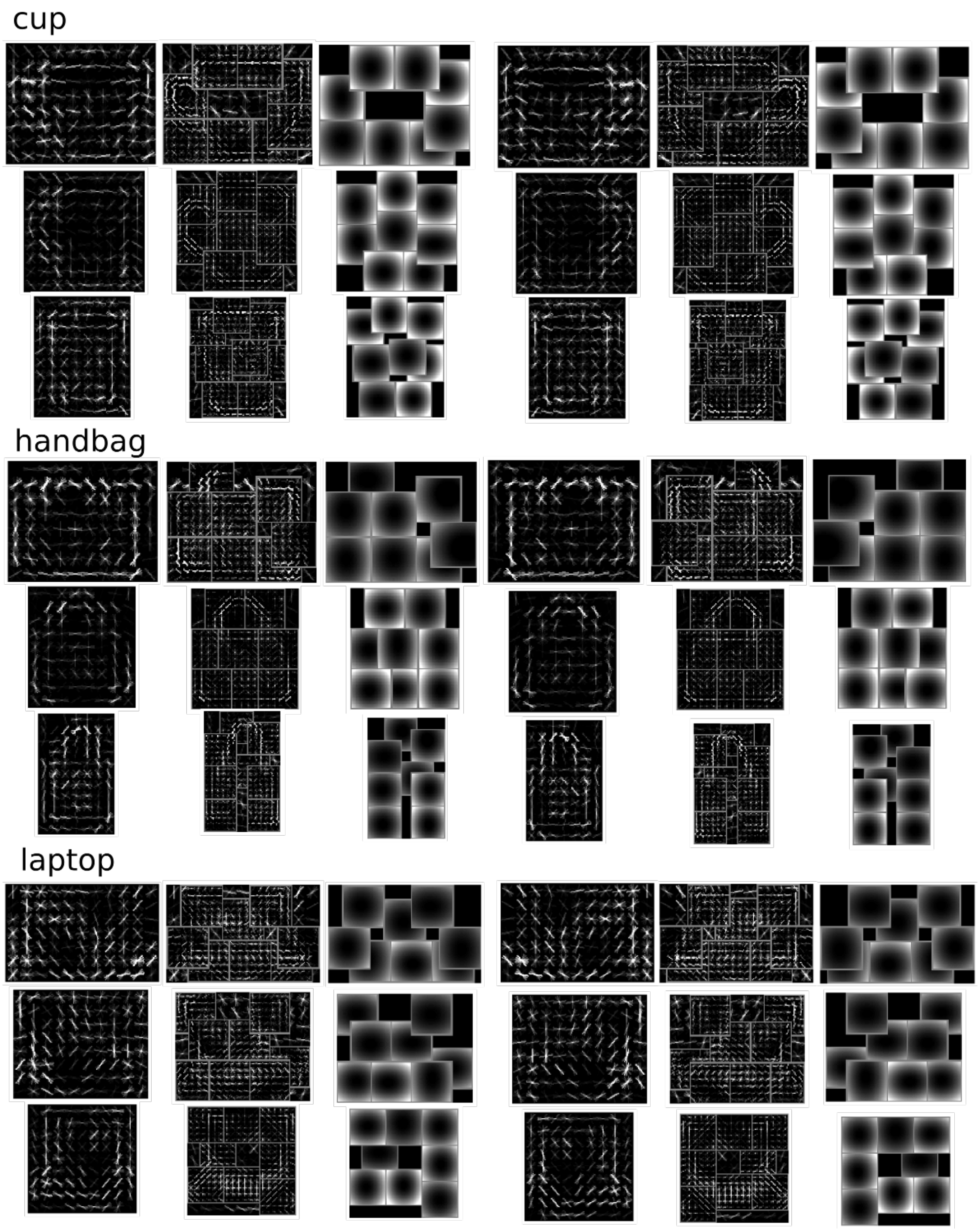


Figure B.3: This figure visualizes the DPMs of cup, handbag and laptop.

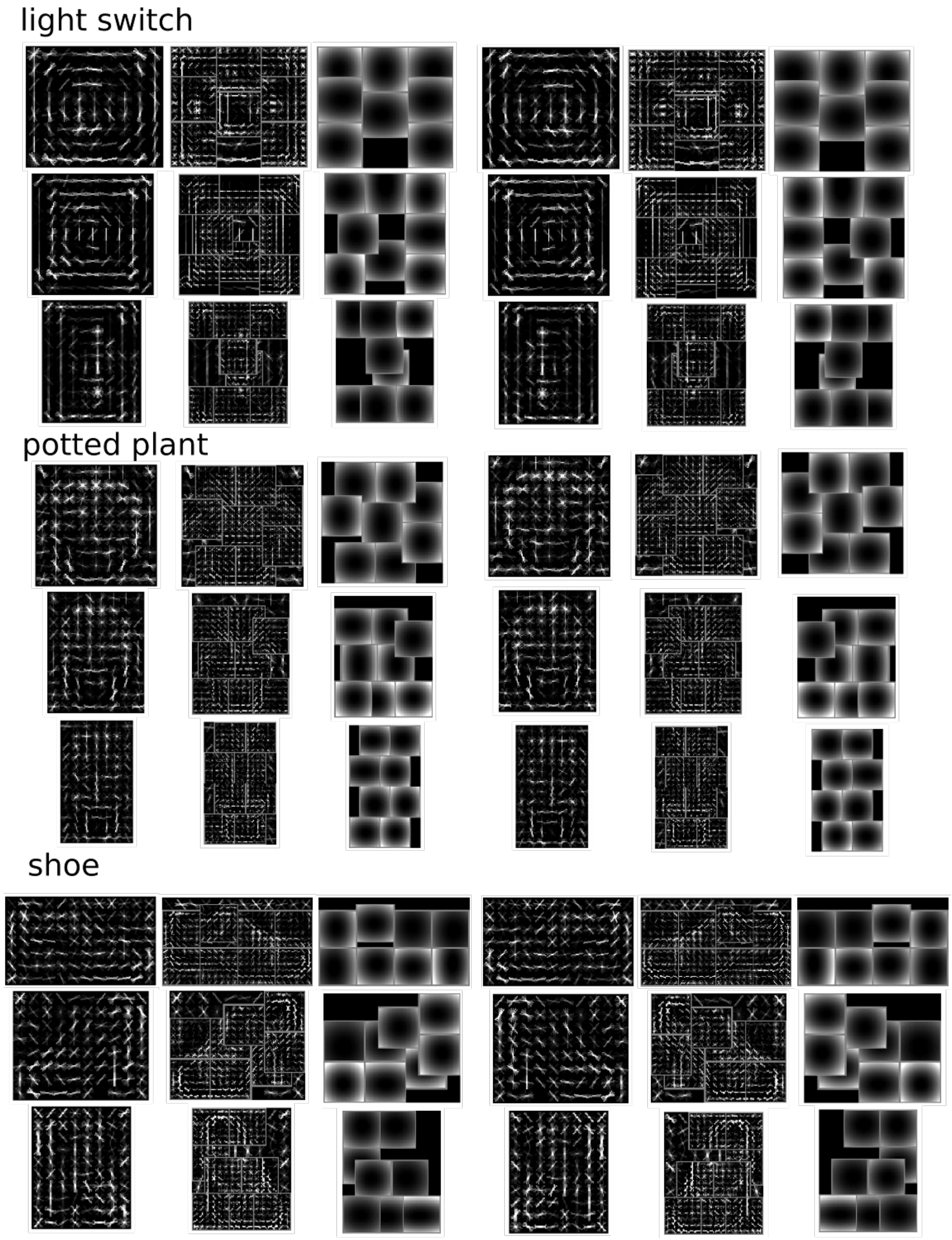


Figure B.4: This figure visualizes the DPMs of light switch, potted plant and shoe.

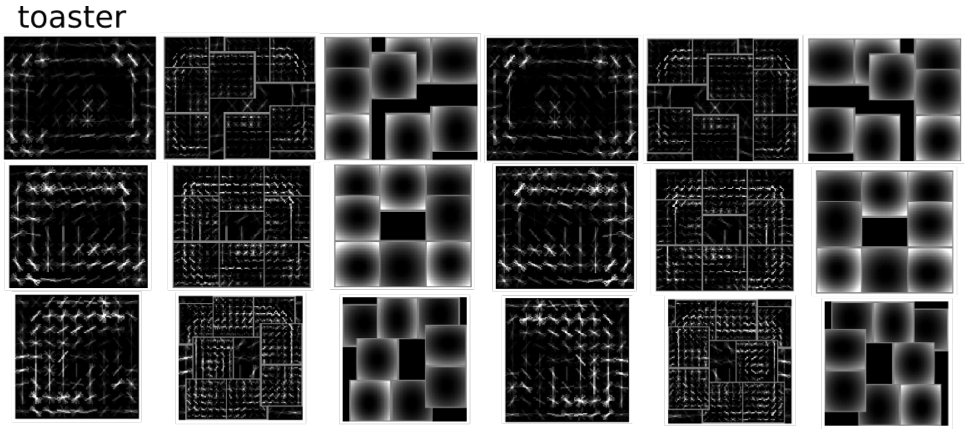


Figure B.5: This figure visualizes the DPMs of toaster.

Bibliography

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

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, 2002.

[3] Ho Seok Ahn. *Advances in Service Robotics*. InTech, 1st edition, 2008.

[4] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *Computer Vision and Pattern Recognition*, 2009.

[5] B. Babenko, S. Branson, and S. Belongie. Similarity metrics for categorization: from monolithic to category specific. In *International Conference on Computer Vision*, 2009.

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

[7] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[8] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *Computer Vision and Pattern Recognition*, 2005.

[9] T.L. Berg and D.A. Forsyth. Animals on the web. In *Computer Vision and Pattern Recognition*, 2006.

[10] N. Beuter, O. Lohmann, J. Schmidt, and F. Kummert. Directed attention - a cognitive vision system for a mobile robot. In *International Symposium on Robot and Human Interactive Communication*, 2009.

[11] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2nd edition, 2007.

[12] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4–5):993–1022, 2003.

[13] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *International Conference on Computer Vision*, 2007.

[14] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *International Conference on Image and Video Retrieval*, 2007.

[15] B.E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, 1992.

[16] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *Computer Vision and Pattern Recognition*, 2005.

[17] S.C. Brubaker, J. Wu, J. Sun, M.D. Mullin, and J.M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1–3):65–86, 2008.

[18] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[19] G.J. Burghouts and J.M. Geusebroek. Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113:48–62, 2009.

[20] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *European Conference on Computer Vision*, 1998.

[21] V. Burzevski and C. K. Mohan. Hierarchical growing cell structures. In *Neural Networks*, 1996.

[22] N.J. Butko and J.R. Movellan. Optimal scanning for faster object detection. In *Computer Vision and Pattern Recognition*, 2009.

[23] H. Cai, K. Mikolajczyk, and J. Matas. Learning linear discriminant projections for dimensionality reduction of image descriptors. In *British Machine Vision Conference*, 2008.

[24] L. Cao. Support vector machines experts for time series forecasting. *Neurocomputing*, 4(2003):321–339, 2003.

[25] G. Carneiro and D.G. Lowe. Sparse flexible models of local features. In *European Conference on Computer Vision*, 2006.

[26] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[27] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 3(2):27:1–27:27, 2011.

[28] T. Chang, T. Liu, and J. Chuang. Improving local learning for object categorization by exploring the effects of ranking. In *Computer Vision and Pattern Recognition*, 2008.

[29] Y. Chen, L.L. Zhu, A. Yuille, and H. Zhang. Unsupervised learning of probabilistic object models (POMs) for object classification, segmentation, and recognition propagation. In *Computer Vision and Pattern Recognition*, 2008.

[30] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Computer Vision and Pattern Recognition*, 2007.

[31] B. Collins, J. Deng, and L. Fei-Fei. Towards scalable dataset construction: An active learning approach. In *European Conference on Computer Vision*, 2008.

[32] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

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

[34] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Computer Vision and Pattern Recognition*, 2005.

[35] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1st edition, 2000.

[36] G. Csurka, C.R. Dance, L. Fan, j. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, 2004.

[37] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, 2005.

[38] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40:5:1–5:60, 2008.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, 2009.

[40] S.J. Dickinson, A. Leonardis, B. Schiele, and M.J. Tarr. *Object Categorization: Computer and Human Vision Perspective*. Cambridge University Press, 1 edition, 2009.

[41] S.K. Divvala, D. Hoiem, J. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition*, 2009.

[42] K. A. J. Doherty, R. G. Adams, and N. Davey. Hierarchical growing neural gas. In *Adaptive and Natural Computing Algorithms*, pages 140–143, 2005.

[43] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

[44] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–308, 2009.

[46] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[47] R.-E Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[48] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition: Workshop on Generative-Model Based Vision*, 2004.

[49] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[50] L. Fei-Fei, R. Fergus, and A. Torralba. Recognizing and learning object categories. Short Course at International Conference on Computer Vision, 2009.

[51] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*, 2005.

[52] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1st edition, 1998.

[53] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer Vision and Pattern Recognition*, 2010.

[54] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[55] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Computer Vision and Pattern Recognition*, 2000.

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

[57] P. Felzenszwalb, D. McAllester, and D. Ramaman. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition*, 2008.

[58] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://www.cs.brown.edu/ pff/latent-release4/.

[59] R. Fergus, Fei-Fei L., P. Perona, and A. Zisserman. Learning object categories from google's image search. In *International Conference on Computer Vision*, 2005.

[60] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition*, 2003.

[61] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *European Conference on Computer Vision*, 2004.

[62] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Computer Vision and Pattern Recognition*, 2005.

[63] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71(3):273–303, 2007.

[64] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36–51(30):1, 2008.

[65] V. Ferrari, T. Tuytelaars, and L. Van Gool. Integrating multiple model views for object recognition. In *Computer Vision and Pattern Recognition*, 2004.

[66] S. Fidler, M. Boben, and A. Leonardis. Similarity-based cross-layered hierarchical representation for object categorization. In *Computer Vision and Pattern Recognition*, 2008.

[67] S. Fidler, M. Boben, and A. Leonardis. Learning hierarchical compositional representations of object structure. In S. Dickinson, A. Leonardis, B Schiele, and M. Tarrm, editors, *Object Categorization: Computer and Human Vision Perspectives*, pages 196–215. Cambridge University Press, 2009.

[68] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22(1):67–92, 1973.

[69] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition, 2000.

[70] F. Fleuret and D. Ferman. Coarse-to-fine face detection. *International Journal of Computer Vision - Special Issue on Statistical and Computational Theories of Vision: Part II*, 41(1–2):85–107, 2001.

[71] LIBSVM A Library for Support Vector Machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[72] P.-E. Forssén. Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition*, 2007.

[73] B. Fritzke. A growing neural gas network learns topologies. In *Neural Information Processing Systems*, pages 625–632, 1995.

[74] S. Furao and O. Hasegawa. An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19:90–106, 2006.

[75] D.M. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *International Conference on Computer Vision*, 1999.

[76] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *International Conference on Computer Vision*, 2009.

[77] J.-M. Geusebroek, J.G. Burghouts, and A.W.M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[78] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, 2005.

[79] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *Computer Vision and Pattern Recognition*, 2006.

[80] K. Grauman and T. Darrell. Approximate correspondences in high dimensions. In *Neural Information Processing Systems*, 2007.

[81] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8(36):725–760, 2007.

[82] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[83] R. Gross. Face databases. In S. Li and A. Jain, editors, *Handbook of Face Recognition*. Springer, 2005.

[84] C. Gu, J.J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Computer Vision and Pattern Recognition*, 2009.

[85] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *European Conference on Computer Vision*, 2010.

[86] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, and G. Sagerer. Biron – the bielefeld robot companion. In *Workshop on Advances in Service Robotics*, 2004.

[87] B. Heisele, P. Ho, J. Wu, and T. Poggio. Face recognition: component-based versus global approaches. *Computer Vision and Image Understanding*, 91(1–2):6–21, 2003.

[88] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *European Conference on Computer Vision*, 2008.

[89] S. Helmer, D. Meger, P.E. Forssén, S. McCann, T. Southey, M. Baumann, K. Lai, B. Dow, J.J. Little, and D.G. Lowe. Curious George: The UBC semantic robot vision system. In *AAAI Technical Report Series (AAAI-WS-08-XX)*, 2007.

[90] S. Helmer, D. Meger, M. Muja, J.J. Little, and D.G. Lowe. Multiple viewpoint recognition and localization. In *Asian Conference on Computer Vision*, 2010.

[91] V. J. Hodge and J. Austin. Hierarchical growing cell structures: TreeGCS. *Knowledge and Data Engineering*, 13(2):207–218, 2001.

[92] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. In *Computer Vision and Pattern Recognition*, 2006.

[93] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for multi-view object class recognition and segmentation. In *Computer Vision and Pattern Recognition*, 2007.

[94] A. D. Holub, M. Welling, and P. Perona. Combining generative models and fisher kernels for object class recognition. In *International Conference on Computer Vision*, 2005.

[95] W. Hu and S.-C. Zhu. Learning a probabilistic model mixing 3D and 2D primitives for view invariant object recognition. In *Computer Vision and Pattern Recognition*, 2010.

[96] G. Hua, M. Brown, and S. Winder. Discriminant embedding for local image descriptors. In *International Conference on Computer Vision*, 2007.

[97] B. Jähne. *Digitale Bildverarbeitung*. Springer, 7. edition, 2010.

[98] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[99] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *International Conference on Computer Vision*, 2005.

[100] T. Kadir, A. Zisserman, and J.M. Brady. An affine invariant salient region detector. In *European Conference on Computer Vision*, 2004.

[101] T. Kanade. *Computer Recognition of Humans Faces*. Birkhauser, Basel, 1977.

[102] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Computer Vision and Pattern Recognition*, 2009.

[103] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition*, 2004.

[104] M. Kortkamp and S. Wachsmuth. Continuous visual codebooks with a limited branching tree growing neural gas. In *International Conference on Artificial Neural Networks*, 2010.

[105] M. Kortkamp and S. Wachsmuth. View-tuned approximate partial matching kernel from hierarchical growing neural gases. In *International Conference on Artificial Neural Networks*, 2011.

[106] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[107] M.P. Kumar, P.H.S. Torr, and A. Zisserman. OBJ CUT. In *Computer Vision and Pattern Recognition*, 2005.

[108] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 1st edition, 2004.

[109] C.H. Lampert. An efficient divide-and-conquer cascade for nonliear object detection. In *Computer Vision and Pattern Recognition*, 2010.

[110] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition*, 2008.

[111] S. Lazebnik, C. Schmid, and J. Ponce. Affine-invariant local descriptors and neighborhood statistics for texture recognition. In *Computer Vision and Pattern Recognition*, 2003.

[112] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *Computer Vision and Pattern Recognition*, 2003.

[113] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, 2006.

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

[115] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *European Conference on Computer Vision*, 2004.

[116] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[117] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Computer Vision and Pattern Recognition*, 2003.

[118] C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *Computer Vision and Pattern Recognition*, 2008.

[119] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Computer Vision and Pattern Recognition*, 2007.

[120] 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, 2001.

[121] T.K. Leung, M.C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *International Conference on Computer Vision*, 1995.

[122] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, 1998.

[123] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition*, 2009.

[124] L.-J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: Automatic online picture collection via incremental model learning. In *Computer Vision and Pattern Recognition*, 2007.

[125] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *Computer Vision and Pattern Recognition*, 2010.

[126] M. Linke. Retrieval of web images for computer vision research. Bachelor's Thesis, Bielefeld University, 2009.

[127] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[128] Z. Lu and H.H.S. Ip. Image categorization with spatial mismatch kernels. In *Computer Vision and Pattern Recognition*, 2009.

[129] S. Maji, A.C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition*, 2008.

[130] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Computer Vision and Pattern Recognition*, 2009.

[131] S. Marsland, J. Shapiro, and U. Nehmzow. A self-organising network that grows when required. *Neural Networks*, 25(8-9):1041–1058, 2002.

[132] M. Marszalek and C. Schmid. Spatial weighting for bag-of-features. In *Computer Vision and Pattern Recognition*, 2006.

[133] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, 2002.

[134] D. Meger, A. Gupta, and J.J. Little. Viewpoint detection models for sequential embodied object category recognition. In *International Conference on Robotics and Automation*, 2010.

[135] D. Meger, M. Muja, S. Helmer, A. Gupta, C. Gamroth, T. Hoffman, M. Baumann, T. Southey, P. Fazli, W. Wohlkinger, P. Viswanathan, J.J. Little, D.G. Lowe, and J. Orwell. Curious george: An integrated visual search platform. In *Canadian Robot Vision*, 2010.

[136] K. Mikolajczyk and J. Matas. Improving descriptors for fast tree matching by optimal linear projection. In *International Conference on Computer Vision*, 2007.

[137] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[138] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[139] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, 2004.

[140] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. J. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[141] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[142] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *International Journal of Computer Vision*, 23(4):349–361, 2001.

[143] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[144] S. Munder and D.M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006.

[145] J. Mutch and D.G. Lowe. Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80(1):45–57, 2008.

[146] T. Nakamura, T. Nagai, and N. Iwahashi. Multimodal object categorization by a robot. In *Computer Vision and Pattern Recognition*, 2007.

[147] S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (COIL-100). Technical report, CUCS-006-96, Columbia University, 1996.

[148] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, 2006.

[149] F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing*, 14(2):169–180, 2005.

[150] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[151] B. Ommer and J.M. Buhmann. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.

[152] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006.

[153] A. Opelt, A. Prinz, and A. Zisserman. Learning an alphabet of shape and appearance for multi-class object detection. *International Journal of Computer Vision*, 1(80):16–44, 2008.

[154] F. Orabona, L. Jie, and B. Caputo. Online-batch strongly convex multi kernel learning. In *Computer Vision and Pattern Recognition*, 2010.

[155] B. Schiele P. Dollár, C. Wojek and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition*, 2009.

[156] S.E. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1st edition, 1999.

[157] M. Pedersoli, A. Vedaldi, and J. Gonzàlez. A coarse-to-fine approach for fast deformable object detection. In *Computer Vision and Pattern Recognition*, 2011.

[158] J. Peltason, I. Lütkebohle, B. Wrede, and M Hanheide. Mixed initiative in interactive robotic learning. In *International Symposium on Robot and Human Interactive Communication*, 2009.

[159] J. Peltason and B. Wrede. Modeling human-robot interaction based on generic interaction patterns. In *AAAI Fall Symposium: Dialog with Robots*, 2010.

[160] F. Perronnin, G. Dance, C. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *European Conference on Computer Vision*, 2006.

[161] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.

[162] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1):e27, 2008.

[163] A. Pinz. Object categorization. *Foundations and Trends in Computer Graphics and Computer Vision*, 1(4):255–353, 2005.

[164] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods*, pages 185–208. MIT Press, 1999.

[165] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[166] J. Ponce, T.L. Berg, M. Everingham, D.A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B.C. Russell, A. Torralba, C.K.I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In *Toward Category-Level Object Recognition. Springer Lecture Notes in Computer Science*, 2006.

[167] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman. *Toward Category-Level Object Recognition*. Springer Lecture Notes in Computer Science, 2006.

[168] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition*, 2009.

[169] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *International Conference on Computer Vision*, 2007.

[170] D. Ramanan and S. Baker. Local distance functions: A taxonomy, new algorithms, and an evaluation. In *International Conference on Computer Vision*, 2009.

[171] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. In *Computer Vision and Pattern Recognition*, 2011.

[172] S. Ravishankar, A. Jain, and A. Mittal. Multi-stage contour based detection of deformable objects. In *European Conference on Computer Vision*, 2008.

[173] H. Riemenschneider, M. Donoser, and H. Bischof. Using partial edge contour matches for efficient object category localization. In *European Conference on Computer Vision*, 2010.

[174] Robocup@Home. `http://www.robocupathome.org`.

[175] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake. Computationally efficient face detection. In *International Conference on Computer Vision*, 2001.

[176] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[177] O. Russakovsky and A.Y. Ng. A steiner tree approach to efficient object detection. In *Computer Vision and Pattern Recognition*, 2010.

[178] B.C. Russell, W.T. Freeman, A.A. Efros, and J. Sivic. Using multiple segmentations to discover objects and their extent in image collections. In *Computer Vision and Pattern Recognition*, 2006.

[179] B.C. Russell, A. Torralba, C. Liu, R. Fergus, and W.T. Freeman. Object recognition by scene alignment. In *Neural Information Processing Systems*, 2007.

[180] S. Savarese, A. Criminisi, and J. Winn. Discriminative object class models of appearance and shape by correlatons. In *Computer Vision and Pattern Recognition*, 2006.

[181] C. Schmid, R. Mohr, and C. Bauckhave. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

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

[183] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *International Conference on Computer Vision*, 2007.

[184] E. Seemann, B. Leibe, and B. Schiele. Multi-aspect detection of articulated objects. In *Computer Vision and Pattern Recognition*, 2006.

[185] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition*, 2005.

[186] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[187] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision*, 2005.

[188] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition*, 2008.

[189] J. Sivic, B.C. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman. Discovering object categories in image collections. In *International Conference on Computer Vision*, 2005.

[190] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[191] H. Su, M. Sun, L. Fei-Fei, and S. Savarese. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *International Conference on Computer Vision*, 2009.

[192] E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Willsky. Describing visual scenes using transformed dirichlet processes. In *Neural Information Processing Systems*, 2005.

[193] E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *International Conference on Computer Vision*, 2005.

[194] E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Willsky. Depth from familiar objects: A hierarchical model for 3D scenes. In *Computer Vision and Pattern Recognition*, 2006.

[195] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 1st edition, 2010.

[196] CORGNIRON: The Cognitive Robot Companion. `http://www.cogniron.org`, 2004.

[197] SRVC: The Semantic Robot Vision Challenge. `http://http://www.semantic-robot-vision-challenge.org`.

[198] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Elsevier Academic Press, 4th edition, 2009.

[199] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Computer Vision and Pattern Recognition*, 2006.

[200] A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191, 2003.

[201] A. Torralba. Short course on recognizing and learning object categories: Classifier-based methods. Short Course at Computer Vision and Pattern Recognition, 2007.

[202] A. Torralba. How many pixels make an image? *Visual Neuroscience*, 26(1):123–131, 2009.

[203] A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[204] A. Torralba, B.C. Russell, and J. Yuen. LabelMe: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484, 2010.

[205] A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *Computer Vision and Pattern Recognition*, 2010.

[206] M. Tscherepanow. TopoART: A topology learning hierarchical ART network. In *International Conference on Artificial Neural Networks*, pages 157–167, 2010.

[207] M. Tscherepanow, M. Kortkamp, and M. Kammer. A hierarchical art network for the stable incremental learning of topological structures and associations from noisy data. *Neural Networks*, 24(8):906–916, 2011.

[208] T. Tuytelaars and Mikolajczyk. Local invariant feature detectors – A survey. *Foundations and Trends in Computer Graphics and Computer Vision*, 3(3):177–280, 2008.

[209] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[210] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *European Conference on Computer Vision*, 2006.

[211] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1st edition, 1998.

[212] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Computer Vision and Pattern Recognition*, 2010.

[213] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *International Conference on Computer Vision*, 2003.

[214] S. Vijayanarasimhan and K. Grauman. Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *Computer Vision and Pattern Recognition*, 2008.

[215] P. Viola and M.J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[216] OpenCV (Open Source Computer Vision). `http://opencv.willowgarage.com/wiki/`.

[217] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *ACM Conference on Human Factors in Computing Systems*, 2004.

[218] S. Wachsmuth, F. Siepmann, D. Schulze, and A. Swadzba. Tobi – Team of Bielefeld: The human-robot interaction system for robocup@home 2010. In *RoboCup Singapore*, 2010.

[219] G. Wang, Y. Zhang, and L. Fei-Fei. Using dependent regions for object categorization in a generative framework. In *Computer Vision and Pattern Recognition*, 2006.

[220] L. Wang and K.L. Chan. Learning kernel parameters by using class separability measure. In *Advances in Neural Information Processing Systems, Sixth workshop on Kernel Machines*, 2002.

[221] M. Weber, W. Einhaeuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *International Conference on Automatic Face and Gesture Recognition*, 2000.

[222] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, 2000.

[223] Website. `http://google-opensource.blogspot.com/2010/01/2009-semantic-robot-vision-challenge.html`.

[224] Website. `http://www.gnu.org/software/glpk/`.

[225] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Computer Vision and Pattern Recognition*, 2005.

[226] B. Yamauchi. A frontier-based approach for autonomous exploration. In *International Conference on Robotics and Automation*, 1997.

[227] J. Yang, Y.G. Jiang, A.G. Hauptmann, and C.W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Multimedia Information Retrieval*, 2007.

[228] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *Computer Vision and Pattern Recognition*, 2008.

[229] M.-H. Yang, D.J. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[230] T. Yeh and T. Darrell. Dynamic visual category learning. In *Computer Vision and Pattern Recognition*, 2008.

[231] T. Yeh, J. Lee, and T. Darrell. Adaptive vocabulary forests br dynamic indexing and category learning. In *International Conference on Computer Vision*, 2007.

[232] T. Yeh, J.J. Lee, and T. Darrell. Scalable classifiers for internet vision tasks. In *Computer Vision and Pattern Recognition, Workshop on Internet Vision*, 2008.

[233] F. Yuan, L. Twardon, and M. Hanheide. Dynamic path planning adopting human navigation strategies for a domestic mobile robot. In *International Conference on Intelligent Robots and Systems*, 2010.

[234] A.L. Yuille. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1):59–70, 1991.

[235] G. Zehang Sun Bebis and R. Miller. On-road vehicle detection: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):694–711, 2006.

[236] H. Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition*, 2006.

[237] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. 73(2):213–238, 2007.

[238] Q. Zhu, L. Wang, Y. Wu, and J. Shi. Contour context selection for object detection: A set-to-set contour matching approach. In *European Conference on Computer Vision*, 2008.

[239] S.C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006.

[240] L. Ziegler, F. Siepmann, M. Kortkamp, and S. Wachsmuth. Towards an informed search behavior for domestic robots. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots: Workshop on Domestic Service Robots in the Real World*, 2010.